工學博士學位論文

Efficient Key Management Schemes in Multicast Communication

멀티캐스트 통신에서 효율적인 키 관리 방법

2012年 8月

서울大學校 大學院

電氣・컴퓨터工學部

諸　東　炫

Dissertation for the Degree of Doctor

Efficient Key Management Schemes in Multicast Communication

DongHyun Je

Department of Electrical Engineering and Computer Science

Graduate School

Seoul National University

August 2012

Abstract

Efficient Key Management Schemes in Multicast Communication

DongHyun Je

Department of Electrical Engineering and Computer Science
Graduate School
Seoul National University

Advisor: Prof. SeungWoo Seo

As the demand of various applications such as entertainment, communication, and device control has been increased and the network technologies have been advanced, multicast communication becomes one of the promising solutions to reduce the communication complexity because it can deliver a message to a group of users at a single transmission simultaneously. However, as the information exposure becomes one of the main concerns, key management scheme is considered as an essential factor in successfully deploying commercialized applications in multicast communication. On the contrary of unicast, the scalability of multicast is a major obstacle in managing a secret key to provide data confidentiality. Therefore, in this dissertation, I focus on the relations between the security and the resource requirements in managing the secret key among multicast members, and propose the several efficient key management schemes in multicast communication: Computation-and-Storage efficient key tree management protocol, Optimal batch rekeying interval for secure group communication, Membership Dynamics based Key Management for secure vehicular multicast communication.

Keywords: Secure Multicast, Group Key Management, Key Tree Management, Batch Rekeying, Data Confidentiality, Vehicular Multicast Services, Access Control.

Student number: 2006-21289

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

As the demands of data communication extremely increases, it is required to develop new generation networks to support low latency, high mobility, and high speed transmission rate. To efficiently deal with the amount of data, multicast is a promising solution, since multicast can deliver a message to a number of users simultaneously in a single transmission. But, due to the transmission characteristic, the multicast data is exposed to the outside of the multicast members. Thus, group key management has been paid attention, because it guarantees the data confidentiality in transmission as well as supports the multicast communication.

Since the key management of multicast is considered as the same as that of unicast except for the number of users in communication, the key management is not fully investigated in an aspect of multicast communication. Furthermore, the key management schemes are not considered as an essential component in data communication due to the following reasons: First, the key management has no relation with throughput improvement. On the contrary, the key management lowers the data throughput since it consumes the network resources to establish and exchange secret information. Second, key management schemes require other kinds of resources such as memory space for storing secret information such as a key and computation power for encryption and decryption. Third, key management schemes increase end-to-end delay due to cryptographic operations.

However, as commercialized multicast services, in which information exposure is a main concern, have been emerging, the key management becomes important in transmitting multicast data as well. Compared to key management schemes of unicast, those of multicast have to be differently considered. Major different characteristic of multicast is the number of multicast members, which is relatively very larger than that of unicast. To provide the security in multicast communication, multicast members have to share the same key management scheme as well as the same secret key. In addition, the secret key must be hidden from non-multicast members. Since key management schemes must provide the confidentiality in delivering the shared secret key, the key management complexity is greatly increased, which is generally called the scalability problem. Therefore, this dissertation focuses on efficient key management schemes to alleviate the scalability problem as well as to provide the data confidentiality in multicast communication.

1.2 Organization of This Dissertation

1.2.1 Computation-and-Storage-Efficient Key Tree Management Protocol for Secure Multicast Communications

In secure multicast communication, group key management plays an essential role for the guarantee of data confidentiality and integrity. Because communication bandwidth is a limited resource, most group key management schemes for scalable secure multicast communications have focused on reducing the number of update messages, i.e., communication cost. To alleviate the scalability problem, a key tree structure was proposed and many group key management schemes have since adopted this approach. Though a key tree structure reduces communication cost, it often requires, as a tradeoff, a more powerful computing capability for executing several cryptography algorithms and having enough storage for various kinds of keys, i.e., computation cost and storage cost, respectively. However, in mobile devices with limited computation power and storage space, it is crucial to minimize simultaneously the overheads of computation and storage as well as that of communication. This paper proposes a computation-and-storage-efficient key tree structure, and a key tree management protocol for secure multicast communication. By considering the resource information of each group member's device, the proposed protocol manages the key tree structure to maximize the efficiency of the computation and storage costs, and to minimize the increment of the communication cost. Through analysis and simulations using three kinds of cost metrics, it demonstrated that the proposed protocol saves computation and storage costs at the expense of a very small increase in communication cost as a tradeoff when the number of total members and the ratio of members leaving in a batch update interval are moderately large.

1.2.2 Optimal Batch Rekeying Interval for Secure Group Communication

The design of a group key management scheme in secure group communication requires an efficient key updating method. Batch rekeying has been proposed as a way to significantly improve the rekeying efficiency for large-scale group communication. But this scheme is known to sacrifice the data confidentiality, which is a crucial requirement in a secure group communication. Therefore, as well as an efficiency of key updating, the security of the group key management has to be considered if the design is to be based on batch rekeying. This paper proposes a new batch rekeying scheme, which optimizes the long term average cost of the batch rekeying. We model and quantify the total cost per unit time, which consists of the communication cost from the rekeying messages, and the security damage cost which occurs from breaking the data confidentiality. The proposed model considers the various variables such as the communication cost weight, the security damage cost weight and the departure rate of group members. Simulation results show that the proposed batch rekeying scheme can adaptively control the optimal batch rekeying interval and efficiently minimize the total cost per unit time by more than 50%.

1.2.3 Membership Dynamics based Key Management (MDKM) for Secure Vehicular Multicast Communications

As many applications based on wireless communications are being embedded on a vehicular platform, multicast services have begun to be essential for efficient information delivery. Since multicast services are vulnerable to unauthorized access, group key management (GKM) is expected to play an essential role as access control. However, the conventional GKMs are inefficient and inadequate for use in the vehicular environment because the GKM schemes are not designed by considering such characteristics as vehicle mobility, transmission delay, and vehicle safety. This paper proposes a GKM scheme called Membership Dynamics based Key Management (MDKM) for efficient and secure vehicular multicast services. The proposed scheme can greatly reduce the communication cost as well as computation and storage costs. These cost reductions are achieved by decomposing the rekeying data into multicast part and unicast part, and by delivering the minimal number of keys generated from the Key Packing Algorithm (KPA) and the reduced Next Key Information (NKI) derived by the Next Key Information Algorithm (NKIA). As a design problem, this paper also propose an optimization method to minimize the key management cost. The simulation results demonstrate that the proposed scheme exceeds the conventional GKM schemes in terms of the key management cost.

Chapter 2

Computation-and-Storage-Efficient Key Tree Management Protocol for Secure Multicast Communications

2.1 Introduction

Multicast has been an essential communication technology for efficiently providing multimedia services such as IPTV, Video on Demand(VOD) and video conferencing. In multicast services, data are usually protected by employing some cryptographic techniques, thereby guaranteeing confidentiality and integrity. In practice, data protection is often employed as a way of an access control, which allows only subscribed customers to access a particular service. To enforce this, they often use a group key that is used to encrypt data, and must be shared among the legitimate group members [1], [2], so that only the members with the group key can successfully decrypt the encrypted data. Whenever new members join the group, or existing members leave the group, the group key has to be updated with a new one for forward and backward confidentiality, and distributed to all authorized group members [4], [17]. Here, forward confidentiality means that non-group members, who have left the group, should not be able to access to any future keys, and backward confidentiality means that group members who have newly joined the group should not be able to access to old keys.

For this process, a trusted third-party system, called a 'Key Distribution Center (KDC)' has to exist with an authority to manage the group key. The KDC is usually assumed to have exchanged the Individual Key (IK) in advance with each group member. Whenever a group key needs to be changed, the KDC encrypts the group key by using each member's IK to provide confidentiality during a key distribution, and distributes the encrypted group key to all group members, respectively. In this process, the number of messages which contain the encrypted group key is proportional to the number of group members. Therefore, as the number of group members increases, the process causes a scalability problem in distributing a new group key.

To alleviate the scalability problem, many group key management protocols have been proposed which efficiently manage the key update procedure in a secure group communication [5]-[10]. The protocols are mainly based on the idea that the KDC divides the group members into sub-groups, and assigns sub-group keys to each sub-group shared by only the sub-group members. Sub-group members can be further divided into smaller sub-groups which have their own sub-group key. The group key is managed with a key tree structure which is constructed by the layered sub-group keys. With the key tree structure, the KDC can deliver secure data (including a new group key or sub-group keys) to specific sub-group members by using the sub-group key, shared by the members, in a single transmission. Thus, the number of update messages increases by the logarithmic order of the number of group members.

While a key tree structure can alleviate the scalability problem of key update, it causes other problems. To maintain the key tree structure, group members have to store many kinds of keys, e.g., group key, sub-group keys and IK. Whenever the keys need to be updated, group members have to decrypt the encrypted messages which contain the new keys. The number of decryptions for each group member is proportional to the number of sub-group keys each group member has. Although modern mobile devices are equipped with powerful processors and have become more power efficient, they still lack major resources. Furthermore, it is known that the computation of many new complex algorithms for security consume a large amount of battery power. In the case of cryptography algorithms, the energy consumption of decryption performed by mobile devices increases as much as 30% more than that needed for encryption [11]. In addition, since the KDC delivers new keys in an encrypted manner, group members have to decrypt the encrypted messages several times to obtain the new keys. the KDC cannot adopt the new group key for data encryption until all devices have obtained the new group key after a series of computations. As the number of the decryptions increase, group communications incur more decryption delays. Thus, if the number of decryptions is reduced, this will lead to lower battery consumptions as well as decreases in the decryption delay in the group communications.

Furthermore, if a device lacks storage capabilities, it may be impossible within the device to implement a group key management protocol based on a key tree structure. For example, let us consider the Telos B sensor node [12] based on Texas Instruments' 16bit RISC-based MSP430x13x and MSP430x14x [13], which features ultralow-power consumption, and also assume a case of group communication that each group member has to store 20 keys in their secure memory. Since the key length is normally 128 bits, it requires 320 bytes ($128 \times 20 \div 8$ bits) to store 20 keys. Specifically, in case of MSP430F133, since it has only 256 byte RAM, it is impossible to store 20 keys. As another case, MSP430F147 with 1 Kbyte RAM consumes the $\frac{1}{3}$ amount of its memory only to save 20 keys. Furthermore, in addition to the key information, sensor nodes also have to store an operating system such as TinyOS [14] as well as data for a key setup module. Thus, it is important to reduce the number of group keys for mobile devices which lack communication resources.

For these reasons, the performance of group key management protocols is limited by the lowest performance end device with the least amount of resources within the group [15]. Thus, in ensuring effective group communication among the devices with limited resources, the computation and storage cost should be minimized as well as the communication cost. As a result, the overall performance of a group key management protocol has to be evaluated using three aspects, namely, communication, computation and storage costs. While these three aspects are all important, the communication cost has mostly been considered as the evaluation metric since this has been known to be the most influential factor when addressing the scalability problem in assessing group key management protocols [4]-[10].

In this chapter, we propose a new key tree called CSET (Computation-and-Storage-Efficient key Tree structure) and a key tree management protocol called the CSET management protocol, which incurs less computation and storage costs and alleviates the scalability problem in rekeying. The proposed key tree management protocol removes the redundant key tree levels which have little influence on the communication cost when the ratio of leaving members is moderately large. In addition, we provide a new analysis method called the balls and baskets model for the analysis of the communication cost when using a level-homogeneous key tree structure. By using this model, the communication cost of various key tree structures is analyzed, such as the A-ary key tree, level-homogeneous key tree structure and CSET. Moreover, key tree structures are analyzed from three different views of cost, while most group key management schemes have tended to consider only the communication cost. To verify the analysis, a full performance evaluation of the new protocol is conducted here to demonstrate the advantages.

The main contributions of this chapter can be summarized as follows: (a) We design CSET management protocol to maximize the efficiency of the computation and storage costs and minimize the increase in the communication cost; (b) We propose the new key tree structure, CSET, which enables a device to join a secure multicast group, even though it may have small memory and low processing power; (c) We define a more general key tree structure, called the level-homogeneous key tree structure to provide an expression and its analysis of more general key tree structures; (d) We provide a new analysis method referred to as the balls and baskets model, for the analysis of the communication cost of various key tree structures.

The rest of the chapter is organized as follows: Section 2 covers key tree structures and how such keys are updated and the new proposed protocol is described in Section 3. In Section 4, the proposed protocol is analyzed using the three different cost metrics of computation, storage and communication overheads. Section 5 presents the simulation results and the chapter is concluded in section 6.

2.2 Key Tree Structures and Key Update

In this section, we propose a new key tree structure, called the Computation-and-Storage-Efficient Key Tree (CSET), designed to reduce the computation and storage costs. To start the discussions, brief overview of a typical key tree structure is given first and a new one, called the level-homogeneous key tree structure, is then defined. The key tree structures can be classified as depicted in Fig. 2.1.

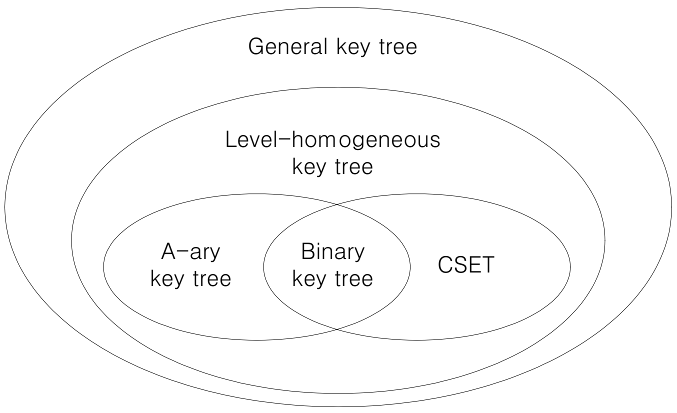

Figure 2.1: A classification of key tree structures.

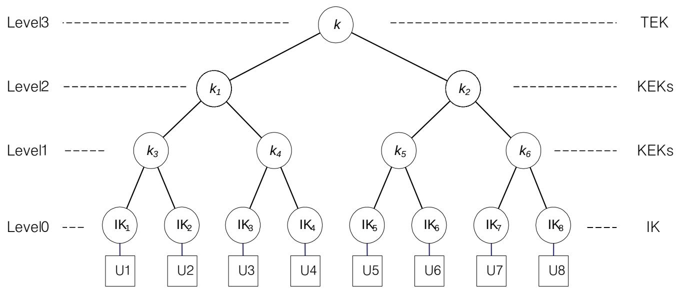

Figure 2.2: An example of a key tree.

2.2.1 Typical Key Tree Structure

In a typical key tree, the keys are classified into three types according to their specific purposes, namely, the Group Key, Sub-group Key, and Individual Key (IK). Since a group key on the top level of the key tree is used to encrypt multicast data, it is known as the Traffic Encryption Key (TEK). The sub-group key is on the levels between the top and the bottom of a key tree and is called the Key Encryption Key (KEK) because it is mainly used to encrypt

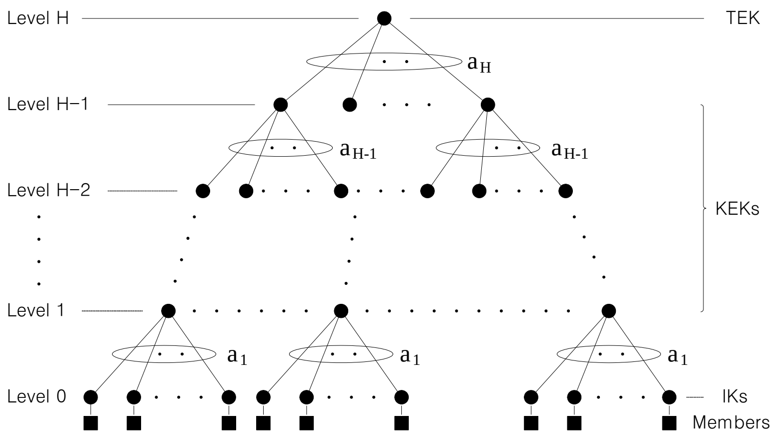

Figure 2.3: Level-homogeneous key tree structure, $T(a_1, a_2, \cdots, a_H)$.

a new key. The IK is on the bottom level of a key tree and is exchanged between the KDC and each group member. Fig. 2.2 shows an example of a key tree, including eight group members. The key tree is a binary tree, where each node has two child nodes. Here, k is the TEK and $k_1, k_2, \cdots, k_6$ are called the KEKs. In case the TEK is changed to a new one, and a key tree structure is not used, the KDC should transmit four encryption messages ($\{k'\}_{IK_i}$, $i = 5, 6, 7$ *and* 8) for specific group members(U5, U6, U7 and U8). Here, $\{k'\}_k$ is the encryption of key k' using key k, and k' is an updated one of key k. However, with a key tree structure, the KDC transmits only one encryption message ($\{k'\}_{k_2}$) for delivering a new TEK to the same sub-group members who already have k_2.

2.2.2 Level-Homogeneous Key Tree Structure

Since a key tree structure alleviates the scalability problem in escalating communication cost, many researchers have studied the structures in terms of those which reduce communication cost [4], [10], [18]. However, these papers provide an analysis of communication cost of only A-ary key tree structures in terms of the number of rekeying messages. The A-ary key tree structure is a one in which all intermediate nodes have A number of child nodes. Therefore, it is only possible to express A-ary key tree as far as, $N = A^H$. Here, H denotes the height of a key tree. For example, when the number of group members is 63 or 114, it is not possible to analyze the communication cost to date.

To express and analyze a more general key tree structure, we define the *level-homogeneous* key tree structure as, *a tree in which every node in the same level has the same degree* [19]. Let a tree structure, $\mathbf{T} = T(a_1, a_2, ..., a_H)$, denote a level-homogeneous tree of H-level such that every parent node on the j-th level from the bottom ($j = 1, 2, ..., H$) has a_j children as described in Fig. 2.3. Therefore, the A-ary tree is a subset of the *level-homogeneous* tree as depicted in Fig. 2.1, which is expressed as $T(A, A, \cdots, A)$.

For example, a key structure with $N = 63$ can be expressed as $T(3, 3, 7)$, $T(3, 7, 3)$, $T(7, 3, 3)$, $T(7, 9)$ and $T(9, 7)$. In particular, the key tree structure depicted in Fig. 2.2 can be expressed as $T(2, 2, 2)$. An accurate method

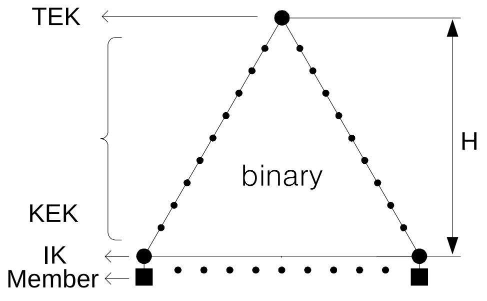

Figure 2.4: Binary tree

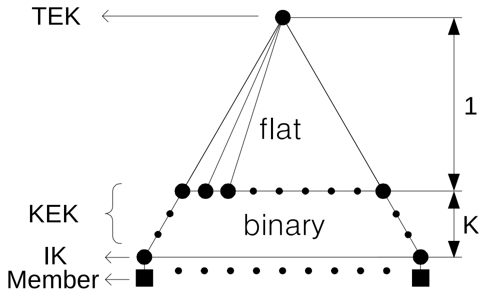

Figure 2.5: Computation-and-Storage-Efficient key Tree (CSET)

to analyze the communication cost of this *level-homogeneous* tree will be provided in a later section, using the balls and baskets model.

2.2.3 Computation-and-Storage-Efficient Key Tree (CSET)

In the key tree based group key management protocols, the KDC delivers new keys to group members after encrypting them with the old keys. Thus, to obtain the new group key and the KEKs, group members should decrypt the encrypted keys with their old keys, or new keys which are obtained after decryptions. Lots of computations, such as the decryption, force the KDC to delay in adopting the new TEK for the data encryption to a group communication system, and devices tend to consume much power to get the new TEK. Therefore, the number of computations should be minimized in a group key management protocol.

Moreover, keys including TEK, KEK and IK have to be stored in the memory of communication devices and group members have to store secure group keys, not in common memory but in secure memory such as a Universal Subscriber Identity Module (USIM), to protect them from various network and software attacks. In addition, communication devices have to store several programs in their secure memory space. Each program requires memory spaces to store their own secure keys as well as their secure data, respectively. Since secure memory space is limited, the number of keys to be stored must also be minimized.

By considering these constraints, we propose the CSET, considering the efficiencies of communication, computation and storage costs. The CSET consists of two hierarchical parts, where the lower part of the CSET consists of binary trees to reduce the communication cost, and the upper part of the CSET consists of a flat tree to minimize computation and storage costs. The TEK is directly connected with the KEKs on the top level of the lower part of the CSET so that it is a subset of the level-homogeneous tree, as shown in Fig. 2.1, and expressed as $T(2, 2, \cdots, 2, 2^{H-K})$. Here, $H = [\log_2(N)]^+$ is the height of a complete binary tree, and N is the total number of the group members, while K is the height of the lower part of the CSET. If $K = (H-1)$, then the CSET is the same as a binary tree.

Fig. 2.4 and Fig. 2.5 compares a typical binary tree with the new proposed CSET where the key on the top level means a group key, TEK, and the intermediate nodes mean auxiliary keys, KEKs. Each member has a unique key, IK.

2.2.4 Key Update

Whenever new members join, or old members leave a communication group, the keys shared by the new, or old members need to be updated with new ones. For this process, there are many key update algorithms such as Logical Key Hierarchy (LKH) [6], One-way Function Trees (OFT) [7] and One-way Key Derivation (OKD) [10] have been proposed. Among them, OKD has been focused upon for the following two reasons : (1) OKD is as secure as the LKH if one-way key derivation function provides strong confidentiality. Furthermore, OKD is more secure than OFT, because OFT is vulnerable to a collusion attack which means that any set of fraudulent members are able to deduce the current used key; (2) OKD is known to produce the smallest communication cost(i.e., the number of update messages) in a leaving scenario and requires no update message in a joining scenario. By using the same one-way key derivation function, some group members can generate new keys in the same manners as the KDC. The KDC does not, however, need to send rekeying messages for delivering the new keys to the group members. Thus, communication cost is greatly reduced. In view of these reasons the OKD is adopted as the key update algorithm in this chapter.

In a key tree as shown in Fig. 2.2, suppose that U_8 leaves the communication group. For forward secrecy, k, k_2 and k_6 should be updated. The KDC randomly selects a key among the child keys of a key to be updated and the selected key should be unknown to the leaving group members. Then, the selected key is used for a key generation process. Assume that the KDC selects k_1, k_5 and IK_7 to generate new keys. The KDC, and the group members who know k_1, k_5 or IK_7, derive the new keys for themselves as follows:

$$k^{'} = f(k_1 \oplus k),\ k_2^{'} = f(k_5 \oplus k_2),\ k_6^{'} = f(IK_7 \oplus k_6).$$

Here, $\oplus$ is a mixing operator, such as exclusive OR, and the derivation function $f(\cdot)$ is a one-way key derivation function, such as a hash. Then, for the other members who can not make the new keys, the following messages will be broadcast by the KDC.

$$KDC \to \{U5, U6\} : \{k^{'}\}_{k_5}$$

$$KDC \to \{U7\} : \{k^{'}\}_{IK_7}, \{k_2^{'}\}_{IK_7}$$

2.2.5 Batch Rekeying

Together with a key tree structure, and efficient group key management schemes, another way to reduce communication cost is by batch rekeying. Batch rekeying [3], [4], [21], [22], [23], [24] is a scheme that rekeys the group keys together simultaneously, after the KDC collects join or leave requests during the time duration called the batch rekeying interval. Some KEKs and TEK are shared among the leaving group members. So, the shared keys are updated once in the batch rekeying while the keys are updated at every joining and leaving event in an individual rekeying. In addition, it can greatly lower the number of rekeying events. For these reasons, batch rekeying is known to reduce the communication cost significantly and this is adopted here in the design of the new protocol.

2.3 Computation-and-Storage-Efficient Key Tree(CSET) Management Protocol

Algorithm 1 Algorithm for calculating, K, the height of the lower part of CSET

```
1: Given Parameter $SI[N]$, $CI[N]$
2: SET $K \Leftarrow infinity$
3: SET $MinSI \Leftarrow infinity$
4: SET $MinCI \Leftarrow infinity$
5: $i \Leftarrow 0$
6: while $i < N$ do
7:    if $SI[i] < MinSI$ then
8:       $MinSI = SI[i]$
9:    end if
10:   if $CI[i] < MinCI$ then
11:      $MinCI = CI[i]$
12:   end if
13:   $i \Leftarrow i + 1$
14: end while
15: if $MinSI \leq MinCI$ then
16:    $K = MinSI - 1$
17: else $\{MinCI < MinSI\}$
18:    $K = MinCI - 1$
19: end if
```

In section II, we proposed new key tree structures, the level-homogeneous key tree structure and the CSET structure. Now, we design a new key tree management protocol, CSET management protocol, which takes into account computation and storage efficiency as well as the communication overhead.

The CSET management protocol provides the KDC with a method that can control the CSET considering communication cost and the resource information such as the CPU power and the amount of free (secure) memory space of each member's device. So, when a non-group member wants to join the group communication, it sends a *'Join Request'* message which contains its resource information. Then, the KDC adjusts the key tree so as to accommodate the lowest-end device that has the least amount of resources by using Algorithm 1. $SI[i]$ and $CI[i]$ are the storage information and computation information of the i-th device, respectively. In this procedure, the non-group member simply adds additional information within the *'Join Request'* message and sends it to the KDC. Since the size of the *'Join Request'* message slightly increases but the CSET management protocol does not require additional message exchanges, the overhead of this procedure is negligible, therefore, its affect is not included in the cost analysis.

2.3.1 Join Operation

The join operation of the CSET is depicted in Fig. 2.6. When a new group member wants to join a communication group, the group member sends a *'Join Request'* message to the KDC. Using the information within the *'Join Request'* message, as well as the information of existing group members,

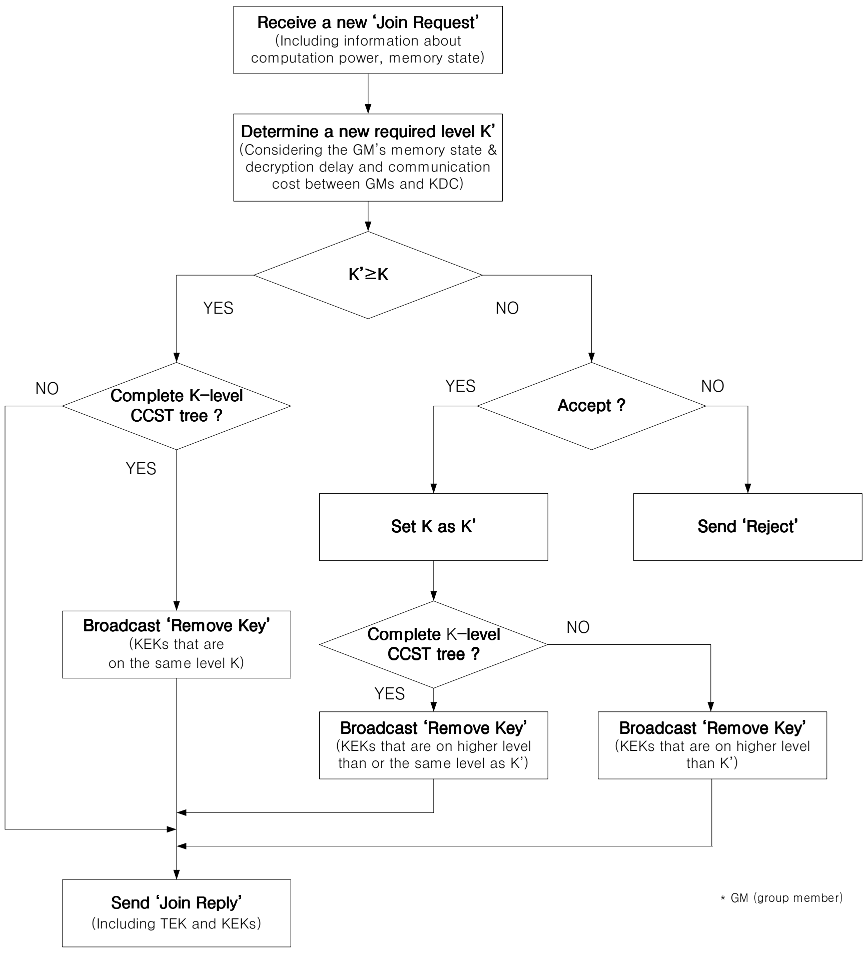
Receive a new 'Join Request'
(Including information about computation power, memory state)
Determine a new required level K'
(Considering the GM's memory state & decryption delay and communication cost between GMs and KDC)
K'≥K
YES
NO
NO
Complete K–level CCST tree ?
YES
YES
Accept ?
NO
Set K as K'
Send 'Reject'
Broadcast 'Remove Key'
(KEKs that are on the same level K)
Complete K–level CCST tree ?
NO
YES
Broadcast 'Remove Key'
(KEKs that are on higher level than or the same level as K')
Broadcast 'Remove Key'
(KEKs that are on higher level than K')
Send 'Join Reply'
(Including TEK and KEKs)
* GM (group member)

Figure 2.6: Flow chart of CSET management for join operation.

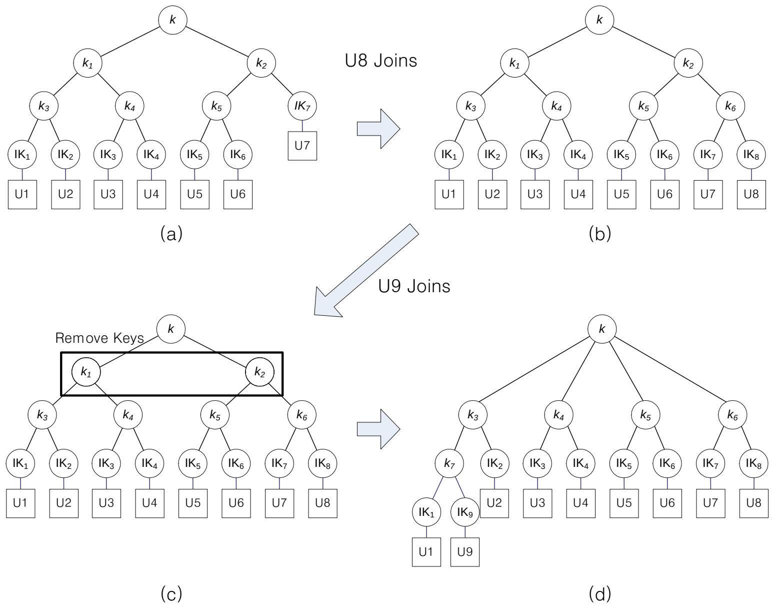

Figure 2.7: Illustration of join operation procedures of CSET management when $K = 2$.

the KDC determines the newly required level K' of the CSET. If the group member's device, which wants to join, has the worst performance, the new level K' must be smaller than the level of the present CSET, K.

If $K' \geq K$, then the KDC does not need to change the key tree level except for the one case when the present key tree structure is the complete K-level CSET. The meaning of *'complete'* is that all group members have the same number of KEKs. In the case of a complete K-level CSET, because the key tree structure requires one more level to accept the new member and to meet the requirement, KDC must broadcast the *'Remove key'* message to remove the KEKs in level K.

The relation, $K' < K$, indicates that the height of the CSET has to be lowered from $K+1$ to $K'+1$ in order to support the new device. Because lowering the height of the CSET increases the communication cost, the KDC has to determine whether to accept the *'Join Request'* message from the device. If the KDC decides that accepting the *'Join Request'* message can excessively increase the communication cost for key update, the KDC can reject the message. Otherwise, the KDC has to support the devices by lowering the height of the lower part of CSET from K to K'. Thus, the KDC broadcasts the *'Remove key'* message to remove KEKs that are on a higher level than K'. If the present tree is a complete K-level CSET, the KDC has to remove one more key level. In this case, the KDC broadcasts the *'Remove key'* message to remove KEKs on the same, or a higher, level than level K'.

Fig. 2.7 shows an illustration of the join operation procedures when new group members join. Assuming a given requirement, $K = 2$, newly joining devices have enough memory space and computation power to meet the requirement. There are seven group members, as seen in Fig. 2.7(a). U8 wants to join the communication group. Since the present key tree is not complete, the KDC can accept U8 as a group member without delivering the *'Remove key'* message and the key tree structure is changed from that seen in Fig. 2.7(a) to that in Fig. 2.7(b). Next, U9 wants to join the communication group. If U9 joins, the height of the lower part of the changed key tree will be three. So, to accept U9 and meet the requirement, the KDC broadcasts the *'Remove key'* message to clear KEKs on level K, as depicted in Fig.

2.7(c) so that U9 can then join the communication group. Finally, the key tree structure will be as shown in Fig. 2.7(d).

2.3.2 Leave Operation

The leave operation of the CSET is described in Fig. 2.8; when a group member wants to leave a communication group, it sends the *'Leave Request'* message to the KDC. After collecting the *'Leave Request'* messages during a batch rekeying interval, the KDC has to update the KEKs to keep confidentiality of the group communication. After group keys are updated, the height of a new key tree can sometimes be changed due to the empty positions of leaving group members in a key tree structure. Thus, the KDC has to recalculate the height of the lower part of the changed key tree structure, K''. Furthermore, if the group member who has the worst performance leaves the group, the KDC can increase the height of the key tree. So, the KDC also determines a newly required level K', considering the remaining group members' memory space and computation power, and computation cost.

It is self-evident that $K'' \leq K \leq K'$. If $K'' = K'$, the height of the changed key tree is the same as the height of the key tree which is required for the group communication. Therefore, in this case, the KDC does not need to heighten or lower the height of the key tree. If $K'' < K'$, then the KDC determines whether it should raise the height of the key tree structure, considering three kinds of costs. If the KDC does not want to raise the height, then it keeps the present key tree. Otherwise, the KDC heightens

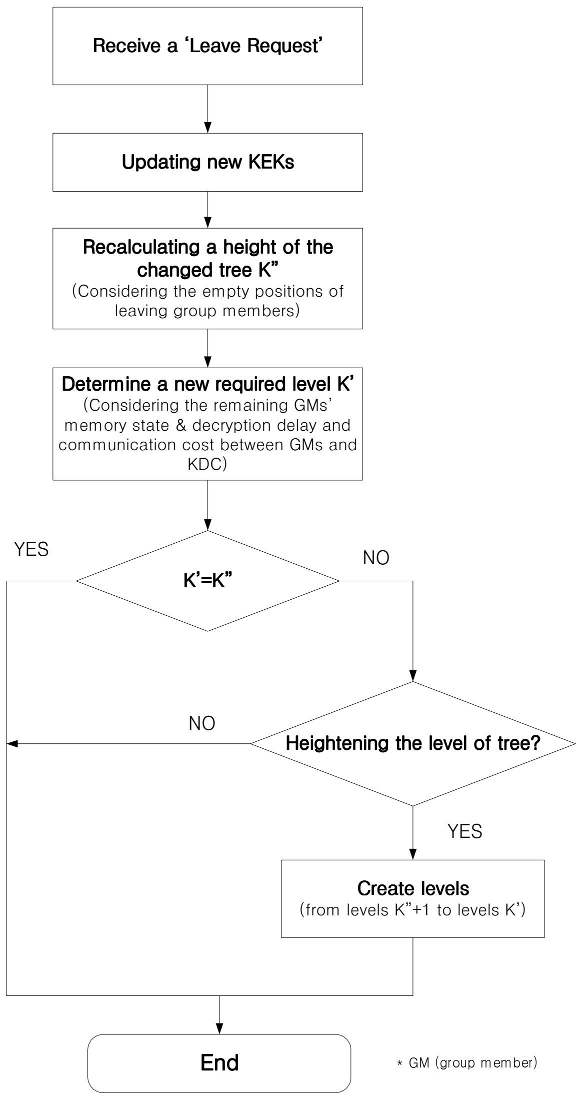
Receive a 'Leave Request'
Updating new KEKs
Recalculating a height of the changed tree K"
(Considering the empty positions of leaving group members)
Determine a new required level K'
(Considering the remaining GMs' memory state & decryption delay and communication cost between GMs and KDC)
YES
K'=K"
NO
NO
Heightening the level of tree?
YES
Create levels
(from levels K"+1 to levels K')
End
* GM (group member)

Figure 2.8: Flow chart of CSET management for leave operation.

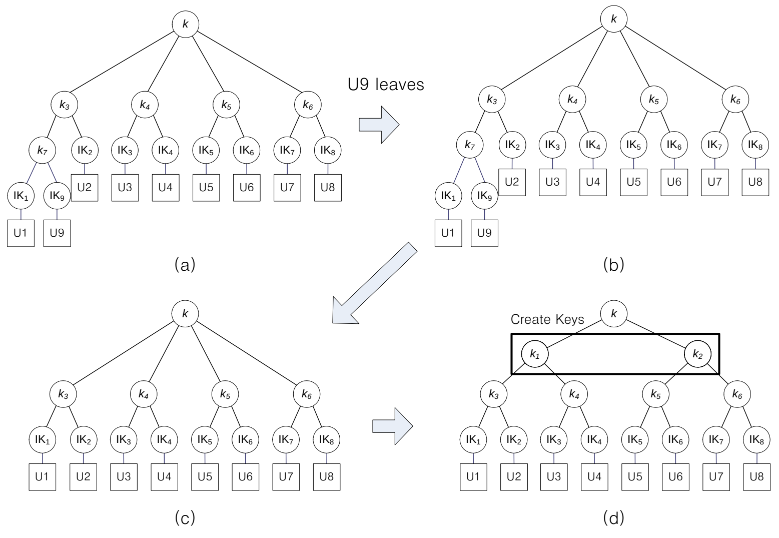

Figure 2.9: Illustration of leave operation procedures of CSET management when $K = 2$.

the levels of the key tree by creating KEKs from $K'' + 1$ to K' in the similar manner as for the OKD.

Fig. 2.9 shows an illustration of the leave operation procedures as a group member leaves. Assume a given requirement, $K = 2$, considering resource information including memory space and computation power. There are nine group members, as shown in Fig. 2.9(a) and U9 wants to leave the communication group. If U9 leaves, the height of the lower part of the changed key tree structure, K'', will be one, as depicted in Fig. 2.9(c). This value is smaller than K. So, the KDC should decide whether it heightens the height of the key tree structure. If the KDC has a policy to maximize the height of a key tree structure as much as possible, then it generates new KEKs on level two. Finally, the key tree structure will be one, as in Fig.

2.9(d).

2.4 Cost Analysis

In this section, we analyze the performance of the CSET from three different viewpoints of cost, and compare them with those of a binary tree, because a binary tree is proven to produce the smallest communication cost in OKD. Starting with a computation and storage cost analysis of a binary tree and the CSET structure. In the analysis of communication cost, we first analyze the communication cost of a level-homogeneous key tree by using the balls and baskets model in a leaving scenario since a binary tree and CSET structure are subsets of this type of level-homogeneous key trees, as shown in Fig. 2.1. The analysis of the communication cost of a binary tree and CSET set up are easily carried out by using the analysis results of a level-homogeneous key tree and, from the analysis results of a binary tree and CSET cases, the increment of the communication cost of CSET will be determined.

2.4.1 Analysis of the Computation Cost

Each group member should have one key on each level. Assume that a group member has a key set $\mathbf{K} = (k_1, k_2, \cdots, k_H)$, where k_H is the key on the H-th level. In the worst case, all keys that a group member has must be updated and the KDC needs to transmit the new keys ($\{k_1^{'}\}_{IK}$ and $\{k_{j+1}^{'}\}_{k_j^{'}}, j = 1, 2, \cdots, h-1$) to the group member. To get the new TEK($k_H^{'}$), the group member has to decrypt the encrypted H messages in a consecutive order $(\{k_1^{'}\}_{IK}, \{k_2^{'}\}_{k_1^{'}}, \cdots, \{k_H^{'}\}_{k_{H-1}^{'}})$. Therefore, the number of computations is same as the height of the key tree.

In case of a binary tree, $N = 2^H$, the height of a binary tree is H and hence the group member can finally obtain the new TEK after H number of decryptions. However, the CSET requires only $K + 1$ computation, because the height of the CSET is $K + 1$. Therefore, the reduction of computation cost is $((H - K - 1)/H)$. Even though the group members' devices have low computation power, the CSET can enable the group members to join the communication group.

2.4.2 Analysis of the Storage Cost

Each group member has to store one group key on each level. Thus, the number of group keys that a group member has to store is equal to the height of the key tree structure. In case of a binary tree, $N = 2^H$, the height of a binary tree is H. Thus, the group member has to store H number of group keys. However, the CSET structure requires only enough memory space to store $K+1$ number of group keys, because the height of the CSET is $K+1$. Therefore, the reduction of storage cost is $((H-K-1)/H)$. Even though a group member's device has small storage, the CSET enables the group member to join the communication group.

2.4.3 Analysis of the Communication Cost

To analyze the communication cost of a level-homogeneous tree, we derive an analytical model of the average number of update messages. In this analytical model, the focus is on the leaving scenario since when new group members join, the number of messages that should be unicast to each joining member is obviously reduced from H to $(K+1)$ in the CSET; the current members can generate new keys for themselves in the similar manner used in OKD.

In the leaving scenario, even though the number of leaving group members is the same, the number of keys to be updated varies according to the positions of the leaving group members in the tree hierarchy. For example, if U1, U2 leave, as shown in Fig. 2.2, k, k_1 and k_3 should be updated. But, if U1, and U8 leave, as shown in Fig. 2.2, k, k_1, k_2, k_3 and k_6 have to be updated. Therefore, the average number of rekeying messages needs to be analyzed using a probabilistic method. For the average case analysis, we model the average number of update messages as the average number of non-full baskets when balls are picked out randomly from the same-sized fully occupied baskets; this is called the balls and baskets model. In this model, the leaving members correspond to the picked balls, and sub-trees at each level correspond to the fully occupied baskets of size being the same as the size of the sub-trees at each level. Thus, the keys to be updated correspond to non-full baskets.

First of all, in order to compute the average number of non-full baskets, the probability that the number of non-full baskets is l is computed as follows:

$$\Pr[n(e,v,w)=l] = \frac{C_l^v \cdot F(e,l,w)}{C_e^{vw}}, \tag{2.1}$$

where $n(e, v, w)$ is the number of non-full baskets when e balls are picked out randomly from v identical w-sized full baskets. A w-sized full basket holds w balls and C_j^i is the binomial coefficient. $F(e, l, w)$ is the number of ways that there is no full basket among the l ones when e balls are picked out from l identical w-sized full baskets, and this can be computed using the inclusion-exclusion principle [16] as follows:

$$F(e, l, w) = \sum_{k=0}^{l-b} (-1)^k \cdot C_k^l \cdot C_e^{w(l-k)} \qquad (b \le l \le B), \tag{2.2}$$

where $b = \lceil e/w \rceil$ and $B = \min(e, l)$ are the minimum and maximum number of non-full baskets, respectively. $C_k^l \cdot C_e^{w(l-k)}$ is the number of ways that e balls are picked out from l baskets after k baskets among l ones are set aside as full. $C_0^l \cdot C_e^{wl}$ $(k = 0)$ includes the cases that the number of full baskets is $0, 1, \ldots, (l - b)$, where each case is counted once. Similarly, $C_1^l \cdot C_e^{w(l-1)}$ $(k = 1)$ includes the cases where the number of full baskets is $1, 2, \ldots, (l-b)$, each of which is counted once, twice, $\ldots$, and C_1^{l-b} times.

Using Eq (2.1) and (2.2), an analysis of the average number of update messages, when e members leave, can be obtained as follows:

$$m(\mathbf{T}, e) = \sum_{i=1}^{H} \sum_{k_i=b_i}^{B_i} \Pr\left[n\left(e, N/\prod_{j=1}^{i} a_j, \prod_{j=1}^{i} a_j \right) = k_i \right] \cdot k_i \cdot (a_i - 1) - e. \tag{2.3}$$

Here, $\mathbf{T}$ is a level-homogeneous key tree structure, i.e., $\mathbf{T} = T(a_1, a_2, ..., a_H)$.

Binary tree structure is denoted as $\mathbf{T}_{binary}$. Then, from the definition of a level-homogeneous key tree, $\mathbf{T}_{binary} = T(2, 2, \cdots, 2)$. From the analysis of the communication cost of a level-homogeneous key tree, the average communication cost of a binary tree can be calculated when e group members leave.

$$m(\mathbf{T}_{binary}, e) = \sum_{i=1}^{H} \sum_{k_i=b_i}^{B_i} \Pr\left[n\left(e, N/\prod_{j=1}^{i} 2, \prod_{j=1}^{i} 2\right) = k_i\right] \cdot k_i - e \tag{2.4}$$

Assume that the tree structure of the CSET is $\mathbf{T}_{CSET(K)}$. Then, $\mathbf{T}_{CSET(K)} = T(2, 2, \cdots, 2, 2^{H-K})$ and the average communication cost for the CSET when e group members leave, can be calculated as follows:

$$\begin{aligned} & m(\mathbf{T}_{CSET(K)}, e) = \\ & \sum_{i=1}^{K} \sum_{k_i=b_i}^{B_i} \Pr\left[n\left(e, N/\prod_{j=1}^{i} 2, \prod_{j=1}^{i} 2\right) = k_i\right] \cdot k_i \\ & + \sum_{k_{K+1}=b_{K+1}}^{B_{K+1}} \Pr\left[n\left(e, 1, N\right) = k_{K+1}\right] \cdot k_{K+1} \cdot (2^{H-K} - 1) - e. \end{aligned} \tag{2.5}$$

Here b_{K+1} is zero and B_{K+1} is one. From the fact that $C_k^0 = 0$ for positive integers k and as e is a positive integer, it is easy to compute $\Pr[n(e,1,N)=0] = C_0^1 \cdot C_0^0 \cdot C_e^{N(0-0)}/C_e^N = 0$, $\Pr[n(e,1,N)=1] = C_1^1 \cdot (C_0^1 \cdot C_e^{N\cdot(1-0)} - C_1^1 \cdot C_e^{N(1-1)})/C_e^N = 1$. The result means that the KDC always has to update TEK with a new one if group members leave a communication group. From the result, Eq.(2.5) can be simplified to

$$\begin{aligned} & m(\mathbf{T}_{CSET(K)}, e) = \\ & \sum_{i=1}^{K} \sum_{k_i=b_i}^{B_i} \Pr\left[n\left(e, N/\prod_{j=1}^{i} 2, \prod_{j=1}^{i} 2\right) = k_i\right] \cdot k_i + (2^{H-K} - 1) - e. \end{aligned} \tag{2.6}$$

2.4.4 Cost Tradeoff of CSET

Since a binary tree was proven to have the best performance in terms of the communication cost in OKD, the communication cost of the CSET must evidently be larger than that for a binary tree. The increment of the communication cost of the CSET is the tradeoff in the cost of the reduction of the computation and storage costs.

The increment of the communication cost can be calculated by subtracting the communication cost of a binary tree from that of the CSET as follows:

$$\begin{aligned} \triangle m(K,e) &= m(\mathbf{T}_{CSET(K)},e) - m(\mathbf{T}_{binary},e) \\ &= (2^{H-K}-1) - \sum_{i=K+1}^{H}\sum_{k_i=b_i}^{B_i} \Pr\left[n\left(e,2^{H-i},2^i\right)=k_i\right]\cdot k_i. \end{aligned} \tag{2.7}$$

Here, *CSET(K)* is the CSET, the height of whose lower part is K. The right hand side of Eq.(2.7) is expanded as follows:

$$\begin{aligned} &\sum_{i=K+1}^{H}\sum_{l=0}^{2^{H-i}} \Pr\left[n\left(e,2^{H-i},2^i\right)=l\right]\cdot l \\ &= \sum_{i=K+1}^{H} \frac{\sum_{l=0}^{2^{H-i}}\sum_{k=0}^{l}(-1)^k\cdot l\cdot C_e^{2^i\cdot(l-k)}\cdot C_l^{2^{H-i}}\cdot C_k^l}{C_e^{2^H}}. \end{aligned} \tag{2.8}$$

If $l = 2^{H-i} - n + k$, the numerator of Eq.(2.8) can be rewritten as

$$\begin{aligned} &\sum_{l=0}^{2^{H-i}}\sum_{k=0}^{l}(-1)^k\cdot l\cdot C_e^{2^i\cdot(l-k)}\cdot C_l^{2^{H-i}}\cdot C_k^l \\ &=\sum_{n=0}^{2^{H-i}} C_e^{2^i\cdot(2^{H-i}-n)}\sum_{k=0}^{n}(-1)^k\cdot C_{2^{H-i}-n+k}^{2^{H-i}}\cdot C_k^{2^{H-i}-n+k}\cdot(2^{H-i}-n+k). \end{aligned} \tag{2.9}$$

Furthermore, if n is not zero, $C_{2^{H-i}-n+k}^{2^{H-i}} \cdot C_k^{2^{H-i}-n+k}$ in Eq.(2.9) can be simplified to

$$\begin{aligned}
C_{2^{H-i}-n+k}^{2^{H-i}} \cdot C_k^{2^{H-i}-n+k} &= \frac{(2^{H-i})!}{(n-k)! \cdot (2^{H-i}-n+k)!} \cdot \frac{(2^{H-i}-n+k)!}{k! \cdot (2^{H-i}-n)!} \\
&= \frac{(2^{H-i})!}{(2^{H-i}-n)! \cdot n!} \cdot \frac{n!}{k! \cdot (n-k)!} \qquad (n \neq 0) \\
&= C_n^{2^{H-i}} \cdot C_k^n.
\end{aligned} \tag{2.10}$$

Thus, if the result of Eq.(2.10) is applied to Eq. (2.9), the result of Eq.(2.9) is given as

$$\begin{aligned}
&\sum_{n=0}^{2^{H-i}} C_e^{2^i \cdot (2^{H-i}-n)} \sum_{k=0}^{n} (-1)^k \cdot C_{2^{H-i}-n+k}^{2^{H-i}} \cdot C_k^{2^{H-i}-n+k} \cdot (2^{H-i}-n+k) \\
&= \sum_{n=0}^{2^{H-i}} C_e^{2^i \cdot (2^{H-i}-n)} \cdot C_n^{2^{H-i}} \sum_{k=0}^{n} (-1)^k \cdot C_k^n \cdot (2^{H-i}-n+k).
\end{aligned} \tag{2.11}$$

where for $n \geq 2$, $\sum_{k=0}^{n} (-1)^k \cdot C_k^n \cdot (2^{H-i}-n+k)$ is equal to zero from the results of the derivation of the Binomial Theorem presented in the Appendix (Eq.(6.3) and Eq.(6.6)). By using Eq.(6.7) and Eq.(6.8) in the Appendix, Eq.(2.11) can be simplified as

$$\begin{aligned}
&\sum_{n=0}^{2^{H-i}} C_e^{2^i\cdot(2^{H-i}-n)} \sum_{k=0}^{n}(-1)^k \cdot C_{2^{H-i}-n+k}^{2^{H-i}} \cdot C_k^{2^{H-i}-n+k} \cdot (2^{H-i}-n+k) \\
&=\sum_{n=0}^{1} C_e^{2^i\cdot(2^{H-i}-n)} \sum_{k=0}^{n}(-1)^k \cdot C_{2^{H-i}-n+k}^{2^{H-i}} \cdot C_k^{2^{H-i}-n+k} \cdot (2^{H-i}-n+k) \\
&=C_e^{2^i\cdot(2^{H-i}-0)} \cdot C_0^{2^{H-i}} \cdot 2^{H-i} + C_e^{2^i\cdot(2^{H-i}-1)} \cdot C_1^{2^{H-i}} \cdot ((2^{H-i}-1)-2^{H-i}) \\
&=2^{H-i} \cdot (C_e^{2^H} - C_e^{2^H-2^i}) \\
&=2^{H-i} \cdot C_e^{2^H} \cdot \left(1 - \frac{C_e^{2^H-2^i}}{C_e^{2^H}}\right).
\end{aligned} \tag{2.12}$$

By applying the result of the Eq.(2.12) to the Eq.(2.8), we can obtain the result as follows:

$$\sum_{i=K+1}^{H} \sum_{l=0}^{2^{H-i}} \Pr\left[n\left(e, 2^{H-i}, 2^i\right) = l\right] \cdot l = \sum_{i=K+1}^{H} \left\{2^{H-i} \cdot \left(1 - \frac{C_e^{2^H-2^i}}{C_e^{2^H}}\right)\right\}. \tag{2.13}$$

Noting that, by Stirling's approximation, $n!$ can be approximated to $\sqrt{2\pi n}(\frac{n}{e})^n$ for large n, if the number of group members is large enough, the equations can be simplified. Under this assumption, C_M^N can be calculated as follows:

$$C_M^N = \frac{N!}{M! \cdot (N-M)!} \approx \sqrt{\frac{1}{2\pi}} \left(\frac{N^{N+0.5}}{M^{M+0.5} \cdot (N-M)^{N-M+0.5}}\right). \tag{2.14}$$

From Eq.(2.14) and $2^H = N$, the part of Eq.(2.13) can be expanded as follows:

$$
\begin{aligned}
\frac{C_e^{2^H-2^i}}{C_e^{2^H}} &= \frac{C_e^{N-2^i}}{C_e^{N}} = \frac{\frac{N-2^i!}{e!\cdot(N-2^i-e)!}}{\frac{N!}{e!\cdot(N-e)!}} = \frac{\frac{N-2^i!}{(N-2^i-e)!}}{\frac{N!}{(N-e)!}} \\
&\approx \frac{\sqrt{\frac{1}{2\pi}}\left(\frac{(N-2^i)^{N-2^i+0.5}}{(N-2^i-e)^{N-2^i-e+0.5}}\right)}{\sqrt{\frac{1}{2\pi}}\left(\frac{N^{N+0.5}}{(N-e)^{N-e+0.5}}\right)}
\end{aligned}
\tag{2.15}
$$

Continuing the expansion in Eq.(2.15),

$$
\begin{aligned}
\frac{C_e^{2^H-2^i}}{C_e^{2^H}} \approx& \left(\frac{N-2^i}{N-2^i-e}\right)^{-2^i}\cdot\left(\frac{N-e}{N-2^i-e}\right)^{N-e+0.5}\cdot\left(\frac{N-2^i}{N}\right)^{N+0.5} \\
=& \left(1-\frac{e}{N-2^i}\right)^{2^i}\cdot\left(\frac{(1-\frac{e}{N})^{N-e+0.5}}{(1-\frac{2^i+e}{N})^{N-e+0.5}}\right)\cdot\left(1-\frac{2^i}{N}\right)^{N+0.5}.
\end{aligned}
\tag{2.16}
$$

From the formal definition of the exponential function, $\lim_{\chi\to\infty}(1+\frac{-a}{\chi})^{\chi} = e^{-a}$ for large N ($N \gg e, N \gg 2^i$), Eq.(2.16) is simplified as follows:

$$
\frac{C_e^{2^H-2^i}}{C_e^{2^H}} \approx \left(1-\frac{e}{N}\right)^{2^i}\cdot\frac{e^{-e}}{e^{-2^i-e}}\cdot e^{-2^i} = \left(1-\frac{e}{N}\right)^{2^i}. \tag{2.17}
$$

Finally, from the results of Eq.(2.7), (2.13), (2.17), the increase in the communication cost for CSET is simplified as follows:

$$
\begin{aligned}
\triangle m(K,e) &= m(\mathbf{T}_{CSET(K)}, e) - m(\mathbf{T}_{binary}, e) \\
&= (2^{H-K}-1) - \sum_{i=K+1}^{H}\left\{2^{H-i}\left(1-\left(1-\frac{e}{N}\right)^{2^i}\right)\right\} \\
&= \sum_{i=K+1}^{H}\left\{2^{H-i}\left(1-\frac{e}{N}\right)^{2^i}\right\}.
\end{aligned}
\tag{2.18}
$$

If Eq.(2.18) is rewritten as a function of the ratio of the leaving members, then the equation is given by

$$\triangle m(K,e) = \triangle m(K, <\alpha N>) \cong \sum_{i=K+1}^{H} \left\{ 2^{H-i}(1-\alpha)^{2^i} \right\}. \qquad (2.19)$$

Here, α is the ratio of the number of leaving group members to the number of total group members in a batch update interval, and $<X>$ is the nearest natural number to X. As either α or K increases, $2^{H-i}(1-\alpha)^{2^i}$ goes to zero. Therefore, if the number of group members and the ratio of members leaving in a batch update interval are moderately large, the protocol incurs a very small increase of communication cost.

2.5 Performance Evaluation

In this section, using three kinds of cost metrics, we evaluate the performance of the CSET protocol by varying the ratio of the leaving group members, the height of a binary tree and the height of the CSET.

2.5.1 Simulation Setup

Throughout the simulation, we consider the centralized group communication where the KDC takes charge of managing group keys. Let us assume that N $(= 2^H)$ group members are deployed. The KDC manages the group key by using the CSET $(T(2, 2, \cdots, 2, 2^{H-K}))$. To provide a general scenario, the group members are supposed to randomly leave the communication group regardless of their positions of a key tree. The counting unit of the communication cost is the number of messages. To compare the computation cost, we consider a sensor node with limited communication resource, the Telos B [12] with the processor MSP430 [13]. From the characteristics of the Telos B, the transmission data rate is set to 250 kbps and the energy consumption to receive the key update messages is set to 0.276 $\mu J/bit$. Assuming that the energy consumption per CPU cycle is fixed (which is justified in [28]), we can calculate the energy consumption to decrypt the data by measuring the number of CPU cycles. Since MSP430F149 operates at 1MHz clock frequency in an active mode and draws a nominal current of $420\mu A$ at 3V, the energy consumption is 1.26 nJ per CPU cycle. The energy consumption and decryption delay to decrypt encrypted messages containing new group keys depend on the computational complexity of a cipher algorithm. So, in the simulation, we consider 3 kinds of cipher algorithms, RC5-32, RC6-32 and Rijndael [25][26][27]. The key length is set to 128 bits equally.

2.5.2 Comparison of Communication Cost Between Analysis and Simulation

Fig. 2.10 compares the communication costs of the simulation with those of the analysis, when the ratio of leaving group members (α) ranges from 0 to 1, the height of a key tree structure(H) is 11 and K is 6, 7 and 8, respectively. Simulation was conducted 100 times, and we calculated the average communication cost from the simulation results. To compare accurately the results, Fig. 2.11 magnifies the part of Fig. 2.10 (0< $\alpha \leq 0.02$). As show in

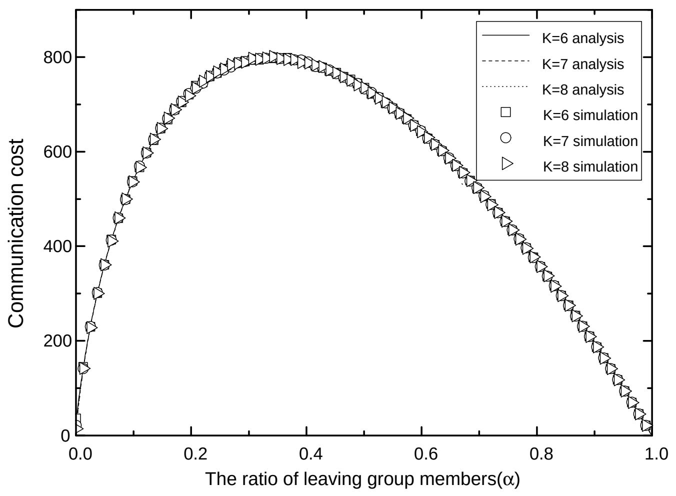

Figure 2.10: Comparison of communication cost between analysis and simulation (H=11, 0< $\alpha \leq 1$).

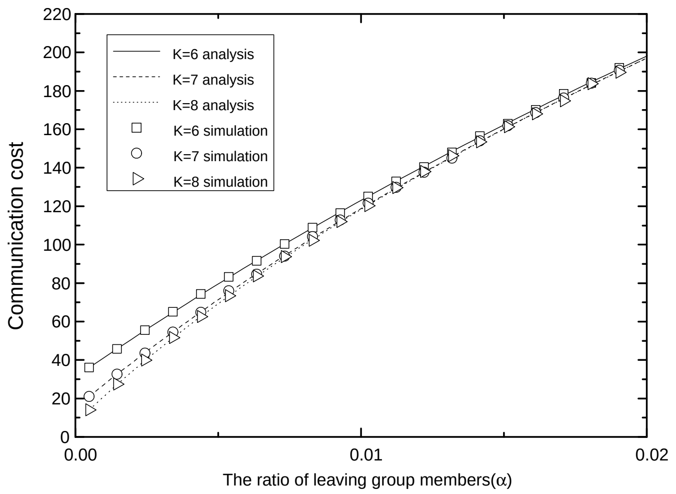

Figure 2.11: Comparison of communication cost between analysis and simulation (H=11, 0< $\alpha \leq 0.02$).

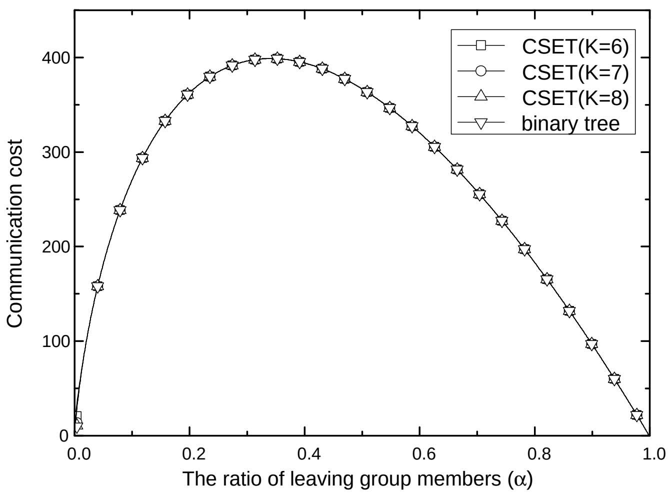

Figure 2.12: the communication cost of CSET and the binary tree (H=10, 0< $\alpha < 1$).

Fig. 2.10 and Fig. 2.11, the communication costs of analysis are very close to those of simulation over the entire range of α. Specifically, when K=7 and $\alpha = 0.01$, the communication cost of the analysis is 116.735, while the cost of the simulation is 117.06. The difference between the two results is less than 0.5%.

2.5.3 Performance Evaluation of CSET

In this subsection, we evaluate the performance of the CSET protocol using three kinds of cost metrics by varying the ratio of the leaving group members, the height of a binary tree and the height of the CSET.

Communication Overhead

Fig. 2.12 shows the communication cost of a binary tree, and that of the CSET, as the ratio of leaving group members increases, when H is ten.

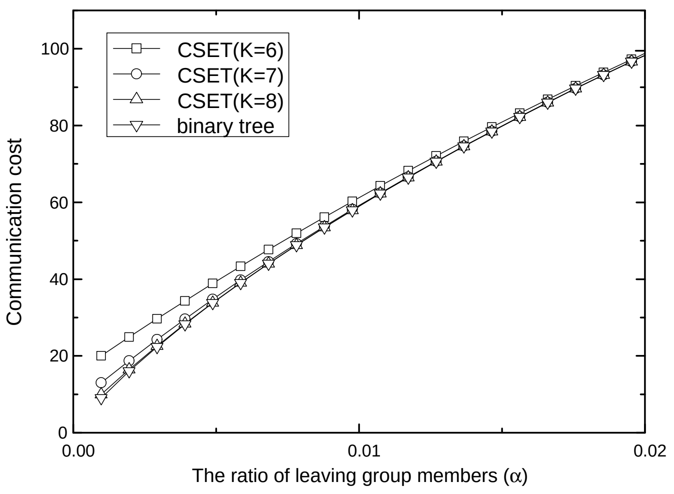

Figure 2.13: Comparison the communication cost of CSET with that of the binary tree (H=10, 0< $\alpha \leq 0.02$).

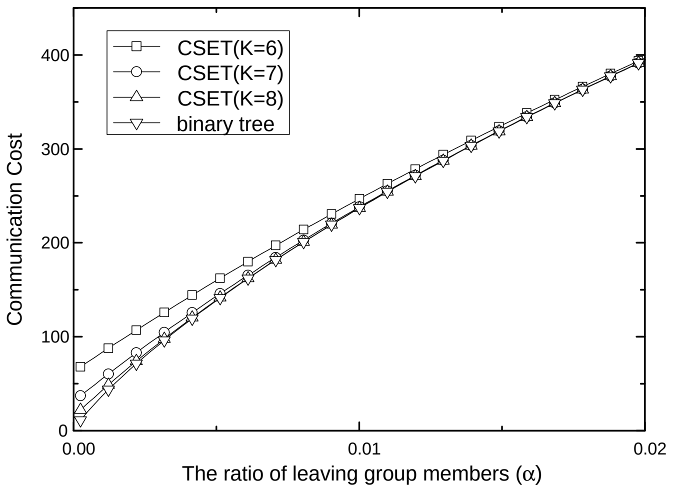

Figure 2.14: Comparison the communication cost of CSET with that of the binary tree (H=12, 0< $\alpha \leq 0.02$).

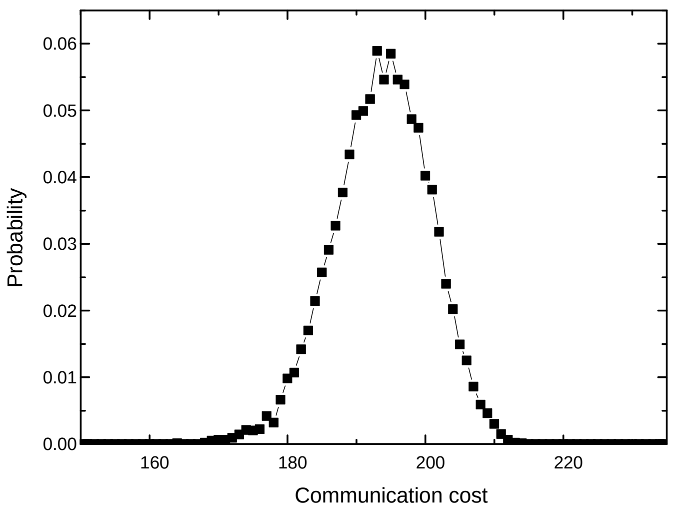

Figure 2.15: The communication cost distribution of CSET (H=11, K=7 and α=0.02).

The communication cost of a binary tree, and that of the CSET, is almost identical in the most part. By magnifying the part where α is small the increment of the communication cost can be observed as Eq.(2.19). Fig. 2.13 and Fig. 2.14 shows the communication cost of the CSET and binary trees, when α ranges from 0 to 0.02, the K ranges from 6 to 8 and the height of a binary tree H is ten and twelve, respectively. The results show that the communication cost of the CSET decreases as K increases and the communication cost of the CSET increases as α increases. The difference between the communication cost of CSET and that of a binary tree decreases as K and α increase. The difference between the communication cost of the CSET and that of a binary tree is considered as a tradeoff between the reduction of the computation and storage costs and the increment of the communication cost.

The shapes of the performance curves as shown in Fig. 2.16 demonstrate the normalized increment of the CSET, Eq.(2.19), which is calculated as $\triangle m(K, e)\ /m(\mathbf{T}_{binary}, e)$. Even if H grows, the shapes of the performance curves are almost the same on the condition of fixed K. Thus, if K is fixed, even though H increases, the reduction of the computation and storage costs is obtained for almost the same increment rate of the communication cost. In Fig. 2.16, the CSET seems to increase the communication cost seriously. However, in reality, the absolute increment of the communication cost is very small as shown in Fig. 2.12, Fig. 2.13 and Fig. 2.14. Therefore, we confirm that the proposed protocol results in a very small increase of the communication cost.

Fig. 2.15 shows the communication cost distribution when the number of group members is 2048 and the ratio of the number of leaving group members(α) is 0.02. The average communication cost is 193.78 and the standard deviation (σ) is 6.94. The communication cost in a single rekeying varies, because of the positions of randomly leaving group members. The standard deviation is relatively smaller than the average communication cost, and the costs are close to the average communication cost. So, we observe that the percentage requiring relatively larger communication cost is low.

Reduction of Storage and Computation Costs

The normalized increment of the communication cost necessary to reduce the computation and storage costs by 50% (i.e., $(H - K - 1)/H = 0.5$) is

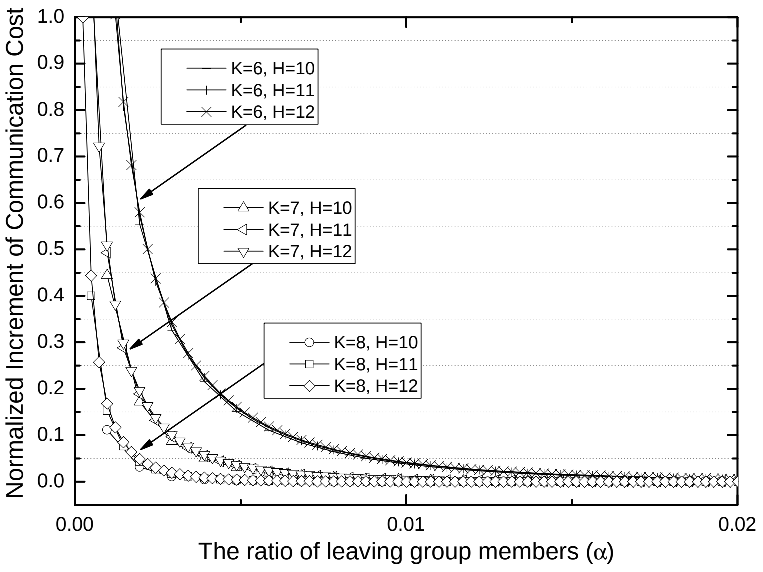

Figure 2.16: Normalized increment of communication cost of CSET.

Table 2.1: Normalized Increment of Communication Cost for Computation and Storage Reduction ($\alpha = 0.01$)

(H-K-1)/H (50%)	5/10	6/12	7/14	8/16	9/18
Increment	0.5196	0.1825	0.0422	0.0053	0.0002

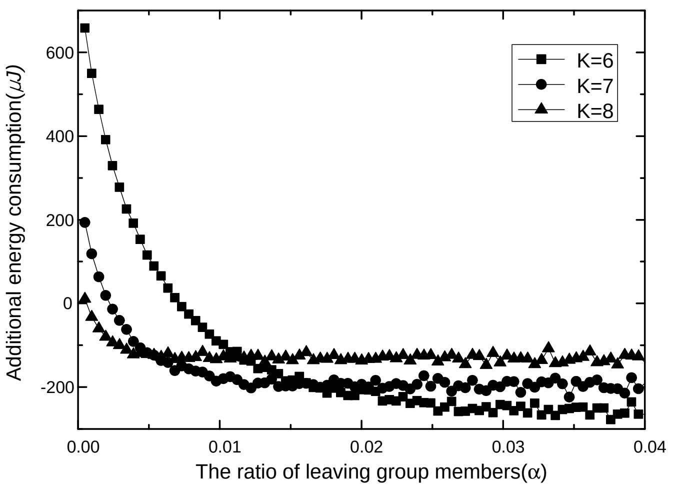

Figure 2.17: Additional energy consumption of CSET with RC5 (H=11).

shown in Table. 3.1, when the ratio of the leaving group members is 1% ($\alpha = 0.01$). The amount of the normalized increment is less than 1% when H is larger than 15. Even though H is small, the proposed protocol can make the normalized increment less than 1% by slightly increasing α by expanding the batch update interval. When $K = 6, H = 14$ ($(H - K - 1)/H = 0.5$) and $\alpha = 0.01$, the normalized increment of the communication cost is only 0.0422 (4.22%). These results confirm that this protocol saves storage cost at the expense of a small increase in communication cost as a tradeoff when the number of total members and the ratio of members leaving in a batch update interval are moderately large.

While the CSET can save the energy in decryption, the additional energy associated to the communication overhead as shown in Fig. 2.16 may be needed. The energy saved on decryption depends on the variables such as characteristics of a microprocessor, a key length and the complexity of a

Table 2.2: CPU cycles for key expansion(per key) and operation nodes(per byte), energy consumption and decryption delay

Cipher	Module	Decryption	Energy consumption (μJ, per 128 bits)	Decryption delay (ms, per 128 bits)
RC5-32	skey	40565	51.112	40.565
	CBC	699	14.091	11.184
	CFB	691	13.931	11.056
	OFB	668	13.467	10.688
	CTR	679	13.689	10.864
RC6-32	skey	93808	118.198	93.808
	CBC	1132	22.821	18.112
	CFB	1129	22.760	18.064
	OFB	1120	22.579	17.920
	CTR	1125	22.680	18.000
Rijndael	skey	5034	6.343	5.034
	CBC	223	4.496	3.568
	CFB	212	4.274	3.392
	OFB	204	4.113	3.264
	CTR	213	4.294	3.408

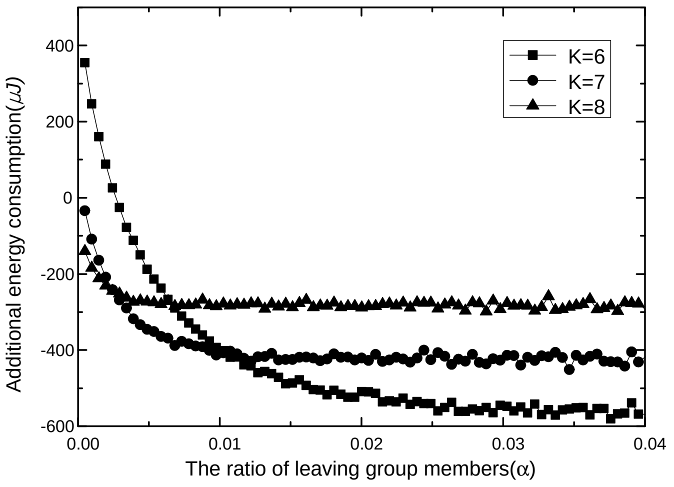

Figure 2.18: Additional energy consumption of CSET with RC6 (H=11).

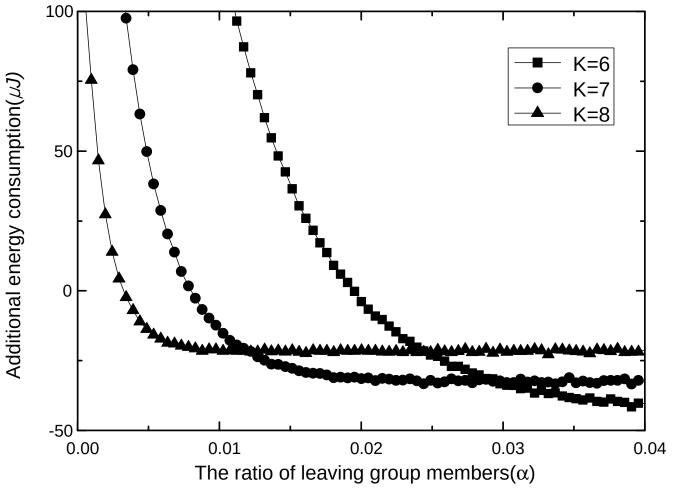

Figure 2.19: Additional energy consumption of CSET with Rijndael (H=11).

cipher algorithm. Also, the energy consumption associated to the additional communication overhead depends on the variables such as characteristics of a device and a key length. We consider the Telos B nodes as a device with limited computation power [12]. Table. 2.2 shows the energy consumption and decryption delay to decrypt the encrypted message according to the cipher algorithms and operation modes [28]. Fig. 2.17, 2.18 and 2.19 show the additional energy consumption per each sensor node to obtain a group key in the CBC operation mode, when the cipher algorithms are RC5-32, RC6-32 and Rijndael, respectively. In those figures, the fact that additional energy consumption is lower than 0 means that the CSET reduces the energy consumption. When the ratio of leaving group members (α) is larger than a specific threshold, the energy saved on decryption overcomes the energy consumption associated to the additional communication overhead. This is because the communication overhead rapidly decreases, as the ratio of leaving group members increases. From the figures, we can observe that the threshold depends on the complexity of algorithms.

In case of decryption delay, it takes only 0.512 ms (=128 bits / 250 kbps) to receive a 128 bits message. However, it takes 8.602 ms to decrypt a 128 bits encrypted message containing a group key when using the Rijndael algorithm, as shown in Table. 2.2. For example, when using H=11, K=7, α=0.01 and the Rijndeal algorithm with CBC, the reduced delay on decryptions is 25.806 ms, while the additional delay associated to the additional communication overhead is only 0.240 ms. Thus, the amount of delay reduction is 25.566 ms. In the most case, the delay saved on decryption overcomes the delay associated the additional communication overhead, because the additional communication overhead is relatively small as shown in Fig. 2.13 and Fig. 2.14.

As a result, the results confirm that the proposed protocol is useful in mobile communication environments where communication devices have very limited resources.

Chapter 3

Optimal Batch Rekeying Interval for Secure Group Communication

3.1 Introduction

As an effective method for simultaneously transmitting identical data to a number of users, the multicast is widely utilized in multimedia services such as video conference, IPTV and Video on Demand (VoD) applications [29], [30]. In multicast, data are generally protected by adopting cryptographic techniques to guarantee confidentiality. To adopt this, the legitimate group members have to share a group key called Traffic Encryption Key (TEK), in order that only the members are successfully able to decrypt the encrypted data. To manage the group key, a 'Key Distribution Center (KDC)' has to exist as a trusted third-party system. Whenever the group membership changes, the KDC takes charge of updating the group key. So, the KDC should encrypt the new group key with each member's Individual Key (IK)

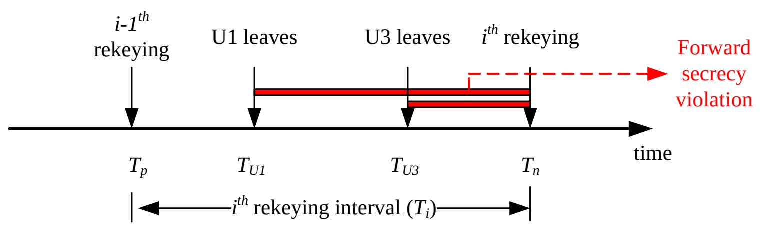

Figure 3.1: An illustration of the batch rekeying process and forward secrecy violation.

to provide data confidentiality during the key distribution process, and then distribute the messages containing the encrypted keys to the authorized group members. But, frequent key updating due to dynamic change of the group membership results in a scalability problem.

To alleviate the scalability problem, a batch rekeying has been proposed. Batch rekeying performs updating the group keys together simultaneously, after the KDC collects the join or leave requests during the time duration known as the batch rekeying interval [3]-[24]. Batch rekeying has advantages. The number of rekeying events can greatly be reduced. And, some Key Encryption Keys (KEK) and a group key are shared among the leaving group members. So, the shared keys are updated once in the batch rekeying, while the keys are updated at every joining and leaving event in an individual rekeying. For these reasons, the batch rekeying is known to be an efficient scheme for significantly reducing communication cost.

Although the batch rekeying alleviates the scalability problem, it sacrifices the data confidentiality, which is an essential requirement in secure group communications. In the case of an individual rekeying process, because the KDC immediately starts rekeying after receiving a *'Leave Request'*, the group communication is secure against the forward security. But, in the case of a batch rekeying, the KDC is supposed to delay updating the group key and KEKs during a batch rekeying interval. Because non-group members, having left, can still access the secure data, previously secure group communication may become vulnerable. Consider a pay-per-minute service in which a service provider, who manages the KDC, provides a service to subscribers for which each subscriber pays a fee. As shown in Fig. 3.1, assume that U1 and U3 leave a communication group at T_{U1} and T_{U3}, respectively. Since U1 and U3 can still access the service without charge for $T_n - T_{\mathrm{U1}}$ and $T_n - T_{\mathrm{U3}}$, respectively, the service provider suffers damage during those periods. Here, T_p denotes the time when the previous batch rekeying was executed, and T_n denotes the time when the next batch rekeying will be executed.

As the batch rekeying interval increases, the number of rekeying messages decreases because the KDC utilizes the advantages of the batch rekeying. However, as a batch rekeying interval decreases, the degree of data confidentiality increases because the KDC minimizes the period that a secure group communication is exposed to non-group members. It is noted that there is a tradeoff between the number of rekeying messages and the degree of data confidentiality, according to the batch rekeying interval. Thus, when designing a batch rekeying, the degree of data confidentiality of a group communication has to be considered [31], [32].

In this chapter, we propose a new batch rekeying scheme which determines the optimal batch rekeying interval to minimize the total cost per unit time, i.e., the sum of the communication cost and security damage cost per unit time. First, we quantify and analyze the communication cost and the security damage cost. To calculate the communication cost, the number of rekeying messages is analyzed by using the balls and baskets problem with a probabilistic approach. The security damage cost is derived from the assumption that the security damage cost is proportional to the duration, such as on a pay per minute basis. The required network and security characteristics are also considered for smooth operation of each service. Specifically, there are services requiring significantly strict security, while others do not. Also, there are services with plenty of communication resources which have significantly low costs at the time of the key update. On the contrary, there are services, limited in communication resources, which have significantly high costs at the time of the key update. By utilizing the proposed optimization function, the proposed batch rekeying scheme can achieve good cost efficiencies. The performance evaluations show that there is an optimal batch rekeying interval. Therefore, the KDC should update the group key to minimize the total cost according to the optimal batch rekeying interval.

The rest of this chapter is organized as follows: in Section II, the background of group key management is provided. We provide the proposed rekeying scheme in Section III. In Section IV, the costs for the optimal batch rekeying interval are analyzed and Section V presents the simulation results with respect to various variables and discusses the results. Finally, Section VI summarizes and concludes the chapter.

3.2 Background

Key Tree Structure

To reduce the number of rekeying messages, a key tree structure has been studied [4], [18], [10]. The KDC and members share the KEKs in addition to TEK and IK, and all of these keys comprise a key tree structure. So, the number of rekeying messages is reduced from $O(N)$ to $O(\log N)$. However, previous works have analyzed only the A-ary key tree structure in terms of the number of messages as this structure is one in which all the intermediate nodes have A number of child nodes. So, it is only possible to express the A-ary key tree in so far as, $N = A^H$. Here, H denotes the height of a key tree. In addition, these papers have considered an individual rekeying. Thus, to express and analyze a more general key tree structure in the batch rekeying, the *level-homogeneous* key tree structure has been adopted as, *a tree in which every node in the same level has the same degree* [19]. $\Upsilon{=}\Upsilon(a_1, a_2, ..., a_H)$ denotes a level-homogeneous tree of H-level such that all the parent nodes on the i-th level from the bottom have a_i number of children ($i = 1, 2, ..., H$).

Key Updating Algorithm

As well as a key tree structure, efficient key updating algorithms such as One-way Key Derivation (OKD) [10], One-way Function Trees (OFT) [7] and Logical Key Hierarchy (LKH) [6] have been proposed to alleviate the scalability problem. This chapter focuses on the OKD protocol for the following three reasons: (1) if the one-way key derivation function guarantees the confidentiality, the OKD is secure against collusion attacks, meaning that the OKD is more secure than the OFT. Here, a collusion attack means that any set of fraudulent leaving group members can deduce the current used keys; (2) the OKD is known as a key update algorithm which produces the smallest communication cost, i.e., the number of updating messages in a leaving scenario; and (3) the OKD requires no updating messages in joining scenarios. Therefore, only backward confidentiality is considered in the OKD. In the case of a leaving scenario, by using the same one-way key derivation function, some group members are able to generate a new group key or KEKs in the same manner as the KDC does. The KDC sends rekeying messages to only the members who cannot generate the new group key or KEKs so that the communication cost is much reduced. For these reasons, this chapter adopts the OKD as the key updating algorithm.

3.3 New Batch Rekeying Scheme

This section defines the two kinds of cost, the communication cost and the security damage cost. Then, we propose a new batch rekeying scheme, which optimizes the long term average cost in the batch rekeying process.

3.3.1 Cost Definition

Communication Cost

The communication cost is defined as a network burden caused from the rekeying messages when the KDC updates a new group key. So, the communication cost depends on the cost per key update message, the number of key update messages and the policy of the network provider. Among them, the number of key update messages can be varied according to the positions of the group members having left in a key tree structure. Therefore, an average number of the key updating messages are analyzed based on the expectation value of the number of sub-trees, including the nodes left from the level-homogeneous key tree structure and the cost per key update message depends on the amount of the communication resources.

Security Damage Cost

The security damage cost is a cost caused by breaking the forward confidentiality of the group communications. Based on the model of pay per minute, a service provider charges subscribers for the actual content delivered. This content is normally protected by using a group key shared by the subscribers (group members) who may be served and the service provider who may manage the KDC. Because the KDC does not update the group key immediately when group members leave in a batch rekeying application, the non-group members can be still served without paying for some time. The service provider can be damaged by this. Therefore, based on a pay

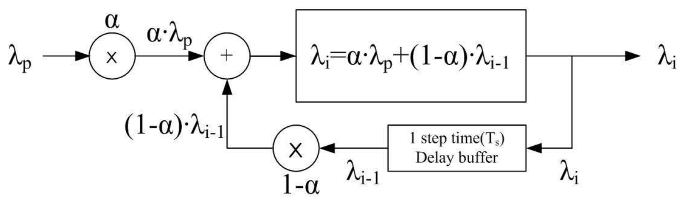

Figure 3.2: Illustration of a method for measuring the departure rate of the group members.

per minute model, the security damage cost is proportional to the duration over which the secure group communication is exposed to each non-group member who has already left the communication group. In addition, the service characteristics and the policy of the service provider also affects the security damage cost.

3.3.2 New Batch Rekeying Scheme

The proposed batch rekeying scheme consists of the following 3 steps:

Measuring Departure Rate

When a group member wants to leave the group, the group member has to send a *'Leave Request'* to the KDC. While collecting the *'Leave Request'*, the KDC measures the departure rate of group members shown in Fig. 3.2, where λ_p denotes the departure rate of group members having left the communication group, which is currently measured in real time, λ_{i-1} denotes the departure rate of the group members still left in the i-1 th rekeying. λ_i is the departure rate used for determining the following step time, α denotes an adjustable variable that is determined by the KDC, $(0 < \alpha \leq 1)$, and specifically, which allows the weights to be adjusted according to the departure rate of the group members currently leaving and another related to the departure rate of the group members having already left. Accordingly, α plays the role of coping with the departure rate that is rapidly changing, and is used for adjusting the costs created at the time of key updating, which may be rapidly distorted where the group members are rapidly leaving in a short time. A delay buffer is used for storing the departure rate of the group members that have previously left, over the following step time.

Optimization of the Batch Rekeying Cost

After summing up the communication cost and the security damage cost, the KDC calculates the total cost per single batch rekeying process. Even if the total cost per batch rekeying is small, the rekeying interval can be short and the number of the batch rekeying processes can be large. So, the total cost in a single batch rekeying cannot be optimal over a long term. Thus, the KDC has to determine an optimal batch rekeying interval (T_i) having a minimum value of the total cost per unit time over the long term average. Therefore, we can formulate the problem of determining an optimal batch rekeying interval as follows:

$$\text{minimize } \overline{C}_{tc}(\Upsilon, \lambda, T)$$
$$\text{subject to } T > 0, \lambda > 0$$

Algorithm 2 provides the optimal batch rekeying interval to minimize the total cost per unit time by using Newton's method, considering a departure rate which is calculated as shown in Fig. 3.2. Here, $\overline{C}_{tc}(\Upsilon, \lambda, T)$ is the total cost per unit time, and this will be fully analyzed in the next section. t and ε are a fixed step size and a tolerant small value properly selected by the KDC, respectively. The total cost per unit time is differentiated with respect to only the batch rekeying interval T. By using the above algorithm, the KDC can determine the optimal batch rekeying interval.

Lemma 1 *The batch rekeying interval obtained by Algorithm 2 is optimal enough to minimize the total cost per unit time.*

Proof 1 *Newton's method is a well-known optimization algorithm and has been proved in Convex Optimization [34]. Since algorithm 2 is based on Newton's method, the batch rekeying interval is optimal.*

Execution of the Batch Rekeying

Algorithm 3 provides the proposed batch rekeying algorithm. Since group members leave a communication group in real time, the optimal batch rekeying interval can be changed because the departure rate (λ_i) may be changed in real time. So, whenever a *'Leave Request'* is received from any leaving group members, the KDC has to recalculate the optimal batch rekeying interval by performing the optimization algorithm 2. Assuming that T_i is determined by the algorithm 2 as the i-th batch rekeying interval. When T_c denotes the current time, the time when the next batch rekeying will be executed is given as $T_n(= T_p + T_i)$. If $T_c < T_n$ is satisfied, the KDC minimizes the total cost per unit time by delaying the batch rekeying until T_n.

Otherwise, if $T_n < T_c$, the KDC has to immediately start the batch rekeying because the total cost per unit time increases over time. Thus, the KDC is supposed to update a new group key when the following requirement is satisfied. $T_n \leq T_c$

Algorithm 2 Algorithm for determining the optimal batch rekeying interval, Function OPTIMIZATION(Υ,λ, T)

```
 1: Given parameter $t$, tolerance $\varepsilon > 0$
 2: Repeat
 3:    $\Delta T_{nt} := -\nabla^2 \overline{C}_{tc}(\Upsilon, \lambda, T)^{-1} \nabla \overline{C}_{tc}(\Upsilon, \lambda, T)$
 4:    $\zeta := \nabla \overline{C}_{tc}(\Upsilon, \lambda, T)^{\mathrm{T}} \nabla^2 \overline{C}_{tc}(\Upsilon, \lambda, T)^{-1} \nabla \overline{C}_{tc}(\Upsilon, \lambda, T)$
 5: if $\zeta^2/2 \leq \varepsilon$ then
 6:    go to step 11
 7: else
 8:    $T := T + t\Delta T_{nt}$
 9:    go to step 3
10: end if
11: Return $T := T$
```

Algorithm 3 The proposed batch rekeying algorithm

```
 1: While(true){
 2: $T_i$:=$\infty$
 3: $T_p := T_c$
 4: While ($T_c < T_p + T_i$){
 5: if KDC receives 'Leave Request' then
 6:    recalculate $\lambda_i$
 7:    if $T_i$=$\infty$ then
 8:       $T_i$:=$T_{i-1}$
 9:    end if
10:    $T_i$:=OPTIMIZATION($\Upsilon$,$\lambda_i, T_i$)
11: end if}
12: Updating a group key and KEKs
13: i:=i+1}
```

3.4 Cost Analysis of the Batch Rekeying

This section firstly analyzes the communication cost and the security damage cost, respectively. Then, the total cost in a single batch rekeying interval and the total cost per unit time are both calculated.

3.4.1 Analysis of Communication Cost

To calculate the communication cost, we firstly derive an analytical model to calculate the average number of messages. In a join scenario, since the current members can generate new keys by themselves in the OKD, the number of messages is unicast to each joining member. Thus, we focus on the leaving scenario.

In the case of a leaving scenario, the number of rekeying messages depends on the positions of the leaving members in the tree structure, though the number of leaving members is the same. So, we analyze the number of rekeying messages by using a probabilistic method. The number of updating messages is modeled as the number of non-full baskets when balls are picked out randomly from the same-sized fully occupied baskets, called the balls and baskets model. In this model, the leaving members correspond to the picked balls, and sub-trees at each level correspond to the fully occupied baskets of size being the same as the size of the sub-trees at each level. Thus, the keys to be updated correspond to non-full baskets.

According to the analysis result of Eq. 2.3 in Chapter 2, the average number of updating messages, when e members leave, can be obtained as follows:

$$m(\Upsilon, e) = \sum_{i=1}^{H}\sum_{k_i=b_i}^{B_i} \Pr\left[n\left(e, N/\prod_{j=1}^{i} a_j, \prod_{j=1}^{i} a_j\right) = k_i\right] \cdot k_i \cdot (a_i - 1) - e. \tag{3.1}$$

Here, Υ is a level-homogeneous key tree structure, i.e., $\Upsilon = \Upsilon(a_1, a_2, ..., a_H)$.

In a batch rekeying interval, the KDC cannot predict the exact number of leaving group members. So, to calculate the average number of rekeying message in the batch rekeying, the Poisson distribution is used in calculating the number of leaving group members, $P_{\lambda,T}(k) = e^{-\lambda T} \cdot (\lambda T)^k / k!$. Here, k is the number of leaving group members. Therefore, the average number of messages in a batch rekeying interval is calculated as follows:

$$E_{\lambda,T}[m(\Upsilon, k)] = \sum_{k=1}^{N} m(\Upsilon, k) \cdot P_{\lambda,T}(k). \tag{3.2}$$

c_{cc} denotes the communication cost constant (the communication cost per unit rekeying message). Then, the communication cost in a single batch rekeying interval is as follows:

$$C_{cc}^{total}(\Upsilon, \lambda, T) = c_{cc} E_{\lambda,T}[m(\Upsilon, k)] = c_{cc} \sum_{k=1}^{N} m(\Upsilon, k) \cdot P_{\lambda,T}(k). \tag{3.3}$$

3.4.2 Analysis of Security Damage Cost

In a group key management protocol based on batch rekeying, the KDC postpones the rekeying process during a batch rekeying interval. Though some of the group members have left the communication group, the members are still able to access the group communication. So, a batch rekeying causes security damage and it is assumed that this security damage cost is proportional to the lengths of the durations in which non-group member can still access the communication group.

Firstly, the security damage cost, caused by a single leaving user, is calculated. Each group member leaves a communication group independently of other group members. So, the security damage cost in a single leaving group member is as follows:

$$\begin{aligned} C_{sd}^{user_i}(T) &= c_{sd}E[T] = c_{sd}\int\limits_{T_p}^{T_n}(T_n - t)P_{sd}(t)dt \\ &= c_{sd}\int\limits_{T_p}^{T_n}(T_n - t)\frac{1}{T_n - T_p}dt = \frac{1}{2}c_{sd}T. \end{aligned} \tag{3.4}$$

Here, c_{sd} denotes a security damage cost constant (the security damage cost per unit time) and $P_{sd}(t)$ is $1/(T_n - T_p)$. Assume that $S_{\Upsilon}(\lambda, T)$ is the set of leaving group members within a batch rekeying interval ($T = (T_n - T_p)$) with λ departure rate. Then, the average number of leaving group members is given as

$$\begin{aligned} E[n(S_{\Upsilon}(\lambda, T))] =& \sum_{k=1}^{N} k \cdot P_{\lambda,T}(k) \\ =& \sum_{k=1}^{N} k \cdot \left(\frac{e^{-\lambda T} \cdot (\lambda T)^k}{k!}\right) = \lambda T. \end{aligned} \tag{3.5}$$

With Eq. (3.4) and Eq. (3.5), the total security damage cost in a single batch rekeying interval is as follows:

$$\begin{aligned} C_{sd}^{total}(\Upsilon, \lambda, T) = & \sum_{user_i \in S_\Upsilon(\lambda, T)} C_{sd}^{user_i}(T) \\ & = n(S_\Upsilon(\lambda, T)) \cdot C_{sd}^{user_i}(T) = \frac{1}{2} c_{sd} \lambda T^2. \end{aligned} \quad (3.6)$$

3.4.3 Analysis of Total Cost

If the communication cost and the security damage cost are added, with weights (W_{cc} and W_{sd})), which denote the weighting values with respect to the communication cost and the security damage cost, respectively, the total cost per single batch rekeying interval is obtained as follows:

$$\begin{aligned} C_{tc}(\Upsilon, \lambda, T) = & W_{sd} \cdot C_{sd}^{total}(\Upsilon, \lambda, T) + W_{cc} \cdot C_{cc}^{total}(\Upsilon, \lambda, T) \\ = & \frac{1}{2} W_{sd} \cdot c_{sd} \lambda T^2 + W_{cc} \cdot c_{cc} \cdot E_{\lambda,T}[m(\Upsilon, k)]. \end{aligned} \quad (3.7)$$

In this case, the weight values (W_{cc} and W_{sd})) may be determined based on a policy provided by the contents provider. For example, if a content has to be securely protected, the content provider can provide more security by increasing the security damage weight. Therefore, the total cost per unit time created due to batch rekeying can be calculated as follows:

$$\begin{aligned} \overline{C}_{tc}(\Upsilon, \lambda, T) = & \frac{C_{tc}(\Upsilon, \lambda, T)}{T} \\ = & \frac{1}{2} W_{sd} \cdot c_{sd} \lambda T + \frac{W_{cc} \cdot c_{cc}}{T} \cdot E_{\lambda,T}[m(\Upsilon, k)]. \end{aligned} \quad (3.8)$$

3.5 Simulation Results and Discussion

This section provides the simulation results of the proposed batch rekeying scheme in terms of the cost per unit time and discusses the results.

3.5.1 Simulation Setup

In the simulation, we consider the centralized group key management protocol where the KDC takes charge of managing group keys with OKD. We deploy 1024 group members and the key structure is $\Upsilon(2,2,2,2,2,2,2,2,2,2)$. Group members are supposed to randomly leave a communication group and non-group members are also supposed to randomly join the communication group. The performance of the proposed scheme is evaluated by varying the communication cost weight (W_{cc}), the security damage cost weight (W_{sd}) and the departure rate (λ).

3.5.2 Performance Evaluation

Fig. 3.3 shows the total cost per unit time as the communication cost weight varies, while the security damage cost weight is the same. The results show that the communication cost per unit time increases as the communication cost weight increases. This is because the communication cost per unit time increases as the communication cost weight increases, while the security damage cost per unit time is the same. Therefore, the KDC has to increase a batch rekeying interval and delay the rekeying to reduce the increment of the communication cost per unit time from the communication cost weight.

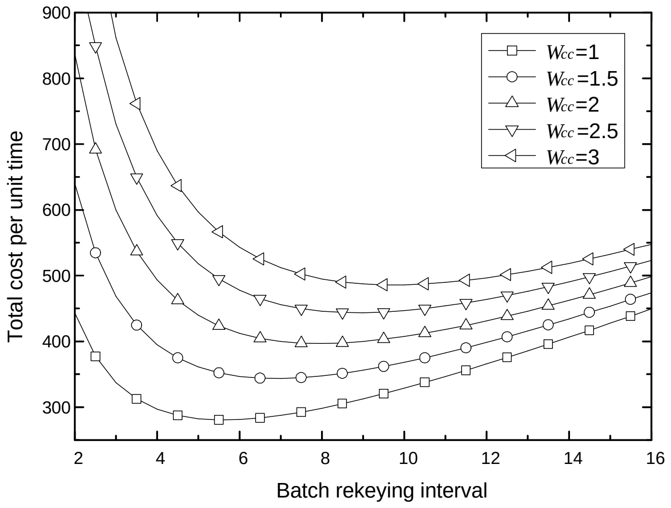

Figure 3.3: Total cost per unit time according to the communication cost weight.

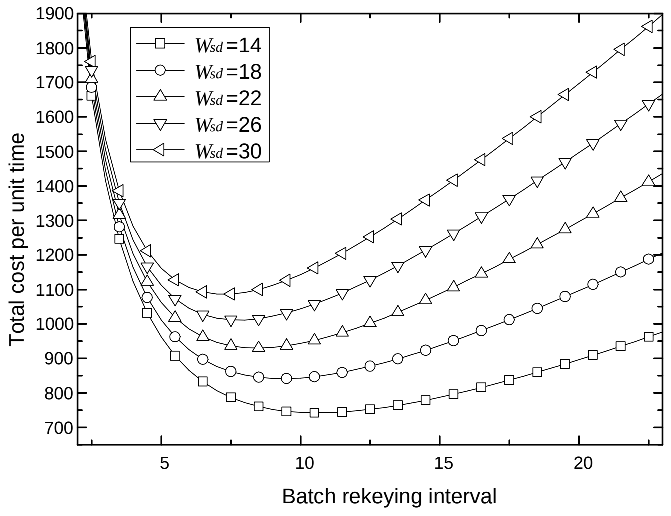

Figure 3.4: Total cost per unit time according to the security damage cost weight.

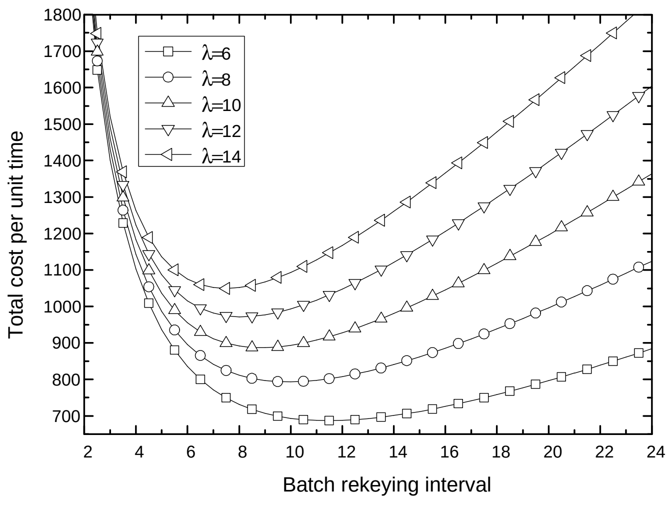

Figure 3.5: Total cost per unit time according to the departure rate.

Fig. 3.4 shows the total cost per unit time as the security damage cost weight varies. The results show that the batch rekeying interval has to decrease in order to reduce the security damage cost per unit time as the security damage cost weight increases. This is because the security damage cost per unit time increases as the security damage cost weight increases, while the communication cost per unit time is the same. Therefore, the KDC has to decrease a batch rekeying interval and advance the rekeying to reduce the increment of the security damage cost per unit time from the security damage cost weight.

Fig. 3.5 shows the total cost per unit time as a departure rate of group members increases. The figure shows that the KDC has to shorten the batch rekeying interval as the departure rate increases. Basically, both the communication cost and security damage cost per unit time increase as the

Table 3.1: Comparison between the optimal batch rekeying and periodic batch rekeying

	optimal batch rekeying		periodic batch rekeying		cost reduction	
λ	T	cost	T	cost	reduction	%
6	11.5	687.0	4.0	2031.4	1344.4	66.2%
8	10.0	793.3	4.0	2046.4	1253.1	61.2%
10	9.0	887.0	4.0	2066.4	1179.4	57.1%
12	8.0	971.6	4.0	2086.4	1114.8	53.4%
14	7.5	1030.6	4.0	2106.4	1075.8	51.1%

departure rate increases. Thus, the result means that the security damage cost per unit time is more affected than the communication cost per unit time by the departure rate.

In Fig. 3.3, as the batch rekeying interval increases, the curves converge regardless of the communication cost weights. This point means that the communication cost per unit time becomes zero because the number of shared group keys increases, as the batch rekeying interval increases. In contrast with the above result, Fig. 3.4 shows the curves converging regardless of the security damage cost weight as a batch rekeying interval decreases. The fact that the batch rekeying interval goes to zero means that the KDC updates a group key shortly after some group members leave. Therefore, the security damage cost goes to zero as the batch rekeying interval decreases and the curves in Fig. 3.4 converge.

The performance of the optimal batch rekeying interval is compared with that of the periodic batch rekeying in Table 3.1. The simulations are conducted with respect to a departure rate. A key tree structure is $\Upsilon(2, 2, 2, 2, 2, 2, 2, 2, 2, 2)$ and the other variables, except the batch rekeying interval, are fixed. The proposed scheme can select the optimal batch rekeying interval. The results are compared with those from a periodic batch rekeying ($T = 4$) situation. The results show that, while the periodic batch rekeying interval causes extra cost, the proposed scheme can adaptively adjust the optimal batch rekeying interval to minimize the total cost per unit time according to the variation of the departure rate.

Chapter 4

Membership Dynamics based Key Management (MDKM) for Secure Vehicular Multicast Communications

4.1 Introduction

Over a decade, the communication architecture for vehicle environments has actively been studied in both academia and industry. Much effort has been made in developing vehicular networks, consisting of *vehicle-to-vehicle* (V2V) and *vehicle-to-Infrastructure* (V2I) communications, for use in vehicular services such as road safety services, driver assistant services, and convenience services [35], [36], [37]. For providing road safety services (emergency warnings, and crash warnings), driver assistant services (hazard warning, and traffic information), and convenience services (Video-on-Demand(VoD), and navigation systems), multicast communication is known to be a promising solution as it allows them widely and quickly transmit the data of the services in a single delivery.

However, one of the major problems in deploying the vehicular multicast communication is the need to maintain service confidentialities which implies that only the authorized vehicles communicate with each other. To guarantee service confidentialities, a secure Group Key (GK) based on a cryptographic method is generally adopted in transmitting the service data, so that only the authorized vehicles having the secure GK can successfully encrypt the service data and decrypt the encrypted data. However, although the GK preserves the confidentiality in data delivery, it causes another problem in sharing the secure GK, i.e., the *scalability problem*. Specifically, whenever a new vehicle joins a service group or the vehicle in a service group leaves a service group, the GK must be updated, and the vehicles in the newly-changed service group have to share the updated GK. In this situation, the process of updating the GK called rekeying can create a serious bottleneck.

In this paper, we design the new GK management (GKM) scheme for secure vehicular multicast communication by considering the characteristics of vehicular communication environments.

4.1.1 Group Key Management

The GKM schemes have been intensively researched as a method for alleviating the communication overhead caused from key updating (or rekeying). *Logical Key Hierarchy* (LKH), one of the pioneering schemes, achieves the reduced overhead [4]. The communication cost is reduced from $O(N)$ to $O(\log N)$; while the storage cost is increased from $O(1)$ to $O(\log N)$ at a member, and from $O(1)$ to $O(N)$ at a *Key Distribution Center (KDC)*; the computation cost is increased from $O(1)$ to $O(\log N)$ at both a member and a KDC.

In addition, *Batch Rekeying* (BR) updating a GK at the end of the time duration has been proposed to further reduce the communication overhead [3], [4], [21], [22], [23], [24]. The BR can provide two advantages: (1) the number of rekeying events can be greatly reduced, and (2) the number of rekeying messages of BR is much smaller than that of *Individual Rekeying (IR)*. However, in exchange for the communication costs reduction, the BR sacrifices forward and backward confidentialities.

To efficiently manage a GK in the cellular network, Yan Sun *et al.* proposed the *topological-matching key management* (TMKM) scheme exploiting the network topology information [39]. TMKM constructs the logical key tree by using the topology information of a physical network; so TMKM can reduce the key updating overhead. However, since TMKM has to bind the logical key tree structure and location information, it causes additional overhead in managing the topological information.

4.1.2 Group Key Management in Vehicular Environments

In the predictable future, the vehicular multicast communication will be enabled by V2V and V2I communications. In this paper, we focus on the vehicular multicast communications between a central management server (KDC) and vehicles via a V2I network, through which a service provider provides vehicular services to drivers. These vehicular service scenarios seem to be similar to those of conventional network services. However, there are two distinguishable differences between them.

1. High mobility: In a vehicular network, the Road Side Unit (RSU), which takes charge of establishing the physical communication link with vehicles in a multicast manner, is supposed to manage the mobility of each vehicle in order to transmit data from the KDC to a specific vehicle and vice versa. Thus KDC and RSU have to continually keep track of the locations of each vehicle. However, some conventional GKM schemes exploiting the network topology information have not considered the effect of vehicle mobility [4], [23].

2. Vehicle safety: Unlike the conventional services, road safety services are significantly critical, since they are closely related to car accidents. Thus the service reliability is one of the most important factors in deploying vehicular multicast services. But urgent service scenarios are not considered in conventional GKM schemes [35].

Because of the above mentioned service characteristics, there are several critical problems in adopting the legacy key management schemes in vehicular networks. They are:

1. Scalability problem: The scalability problem is one of the longstanding issues in GKM. In vehicular networks, the large number of vehicles in wide service area and their dynamic mobility make the scalability problem more complex in designing GKM schemes [39], [38].

2. High delay of rekeying operation: The scalability of vehicular service group may require high communication and computation complexities in updating GK. However, high delay of rekeying operation from large communication and computation complexities may cause the instability and vulnerability of vehicular multicast services [33].

3. Service vulnerability: Though BR has been a promising approach to reduce the communication cost, BR basically sacrifices the data forward confidentiality and backward confidentiality, which are essential requirements in secure multicast communications. When considering the driver safety service, BR can lead to serious vulnerability in providing vehicular multicast services [35].

To solve the above design problems in vehicular multicast communication, we have to design a key management scheme that is totally different from conventional key management schemes.

4.1.3 Contribution

In this paper we design an efficient and secure GKM scheme that we have named the Membership Dynamics based Key Management(MDKM) for vehicular multicast services. The proposed scheme has four distinctive contributions as follows: (1) Elimination of the scalability problem caused from *1-affect-all problem*: The proposed MDKM scheme can individually derive a new GK from the pre-delivered keys by using time information sent from the

KDC, while conventional schemes adapt a key tree structure where the scalability problem is fundamentally inevitable. Thus the proposed scheme can greatly reduce the complexity from $O(\log N)$ to $O(1)$ in every key updating; (2) Low delay of rekeying operation: Because the proposed scheme adopts the key packing and unpacking algorithm which enables vehicles to derive a new GK from the pre-delivered keys by utilizing a derivation function requiring low computing overhead, the proposed scheme enables a vehicle to obtain a next GK within a low delay; (3) Guarantee of strict security: As security is the most important requirement of providing service confidentiality because of the need to support driving safety, the proposed scheme can provide strict security by utilizing IR; (4) Elimination of the topology information in GKM: Since the proposed MDKM scheme manages a GK not combined with network topology information, it does not incur the topology management overhead of the key tree caused from the vehicle location change.

4.1.4 Paper Organization

The rest of the paper is organized as follows. Section II gives the basic models. The MDKM scheme is described by using the models in section III. The various kinds of costs associated with the MDKM are fully analyzed in section IV. In section V, the cost is formulated into an optimization problem and optimized as the design problem. The simulation results are discussed in section VI. Finally, in section VII the conclusions of this paper are presented.

4.2 Model

Before explaining the proposed scheme, we first define the appropriate network, service, vehicle and key models, respectively. By using the models and definitions, we describe the proposed scheme in the following section.

4.2.1 Network Model

In this paper, we assume that V2I communication is supported by RSU or wireless Access Point (AP) of IEEE 802.11 WLAN, IEEE 802,16e WiMAX, or existing cellular facilities in the end link [40], [41]. In addition, the wired infrastructure network supports the communications among RSU, AP, and servers. But since deploying of these infrastructures requires a high cost, vehicular multicast services will be provided in urban area. Therefore, in this paper we work with the vehicular multicast service scenario incorporating heavy traffics.

The network model requires a *Time Synchronization* method, because the proposed key management scheme is operated with subscription time. Thus, we assume a time synchronization method such as *Network Time Protocol* (NTP), which is designed to synchronize the clocks of devices over packet-switched network and variable-latency data networks [42], [43], [44], [45]. The NTP is fault tolerant, reliable, scalable, and secure [44]. In addition, the NTP supports clock synchronization by using broadcast.

4.2.2 Service Model

In this paper, we assume that vehicular multicast services utilize *Pay-per-minute* (PPM) with which a subscriber can be provided with a service during the subscription period. In vehicular services with PPM, each vehicle delivers its subscription period information to the KDC for subscribing the services.

In this service model, there are three players: (1) KDC (who takes charge of managing GKs); (2) Vehicles (which subscribe to the services and receive GKs); (3) The service provider (who provides the vehicular multicast services). Vehicles are supposed to subscribe to the group service operated by a service provider with its subscription period. For a new subscription from a newly joining vehicle, the KDC has to update a GK and deliver new keys to the service provider and vehicles in order to maintain service confidentiality. In the case of leaving a group service, the KDC has to update a GK as well. In this process, since a service provider is not related to the key updating, we focus on the role of the KDC and vehicles in this paper.

4.2.3 Vehicle Model

Let N_u be the set of all possible vehicles that have been served by a service provider. Each vehicle $v_i \in N_u$ is supposed to join the service group at time t_j^i and leave the group at time t_l^i, without losing the generality of vehicle behavior, $t_j^i < t_l^i$.

Definition 4.2.1 Subscription Period Information *(SPI): Let* t_j^i *be a joining time and* t_l^i *be a leaving time. The Subscription Period Information (SPI) consists of the time information for i-th vehicle subscription* $T_S^i = \{t_j^i, t_l^i\}$.

For simplified dynamics, we assume that each vehicle is supposed to subscribe the service once and each vehicle has an unique *SPI* such that for any vehicle v_i and v_j, $t_j^i \neq t_j^j$, $t_l^i \neq t_l^j$ and $t_j^i \neq t_l^j$.

As a subset of the set N_u ($N_t \subset N_u$), we define the *valid vehicle set* at time t such as N_t=$\{v_i | t_j^i < t < t_l^i, v_i \in N_u\}$. Since vehicles in the set N_u dynamically join and leave the service group, the valid vehicle set N_t is also changed in real-time manner. We define the basic functions to accommodate the vehicles dynamics.

Definition 4.2.2 Membership *: Let* v *be an arbitrary vehicle of the universe set* N_u*. Membership at time* t *consists of vehicles in the valid vehicles set* N_t*, such as* $v \in N_t$.

By extension of the membership, we notate the membership dynamics.

Vehicle Join$(VJ\{v\})$: $N_t = N_{t-1} \cup \{v\} = \{x | x \in N_{t-1} \textit{ or } x \in \{v\}\}$.

Vehicle Leave$(VL\{v\})$: $N_t = N_{t-1} - \{v\} = \{x | x \in N_{t-1} \textit{ and } x \notin \{v\}\}$.

4.2.4 Key Model

We introduce the basic key model for explaining the proposed MDKM. For the cryptographic operations, all the *valid* vehicles at time t (N_t) have to share an identically same key(called GK) with the KDC, so that the *valid* vehicles communicate with the KDC via secure multicast channels. In this paper, we consider the centralized GKM with the KDC, which takes charge of updating a new GK and delivering the GK encrypted with a cipher method to vehicles in N_t through internet multicast protocols [52].

The secret key for multicast can be generated in two basic ways. One is through *key generation* $G_k(\cdot)$ which is a function to randomly generate a new *seed key* (SK). For the sake of confidentiality, only the KDC has the authority of *key generation*. The other is through *key derivation* using a *derivation function* $F_d(\cdot)$, such as Message-Digest algorithm (*MDx*), Secure Hash Algorithm (*SHA-x*) or other hash functions [53], [54]. For the sake of confidentiality as well, secure derivation function have to meet the following requirement: For $k_b = F_d(k_a)$, $F_d(\cdot)$ is computationally infeasible to derive k_a, given only k_b.

In addition, each vehicle has a unique *Individual Key (IK)*, IK_i defined as the IK of v_i, which guarantees the secure communication channel between v_i and the KDC.

The encryption and decryption functions are notated as follows: *ciphertext = Enc(key,plaintext)* and *plaintext = Dec(key,ciphertext)*, respectively. The encryption function can be further simplified to *ciphertext*=$\{plaintext\}_{key}$.

4.3 Membership Dynamics based Key Management (MDKM)

We begin this section by introducing the derivation function which is a basic component of the proposed scheme. Then, we propose the MDKM for vehicular multicast communication. We develop the optimization algorithms to minimize network resources of vehicular multicast communication and analyze the costs of the MDKM in the following sections.

4.3.1 Membership Dynamics based Key Derivation

In a group based vehicular communication, vehicle dynamics such as vehicle join (service subscription) or vehicle leave (service expiration) lead to the membership change of valid vehicle set (N_t). Here, we define *membership dynamics* as the membership change from the vehicle dynamics.

Definition 4.3.1 Membership Dynamics*: Let δ be a small positive number such that for any natural number n, $1/n > \delta$. Membership dynamics at time t exists if and only if for two valid vehicle sets N_t at time t and $N_{t+\delta}$ at time $t+\delta$, the following requirement is satisfied: $|N_t - N_{t+\delta}| + |N_{t+\delta} - N_t| > 0$.*

Here, $|N_t - N_{t+\delta}| > 0$ means that one or more vehicles leave the service group because of service expiration at time t, and $|N_{t+\delta} - N_t| > 0$ means that one or more vehicles newly subscribe to the service group at time t.

According to the membership dynamics, the KDC does not need to update keys as long as the membership is *static*.

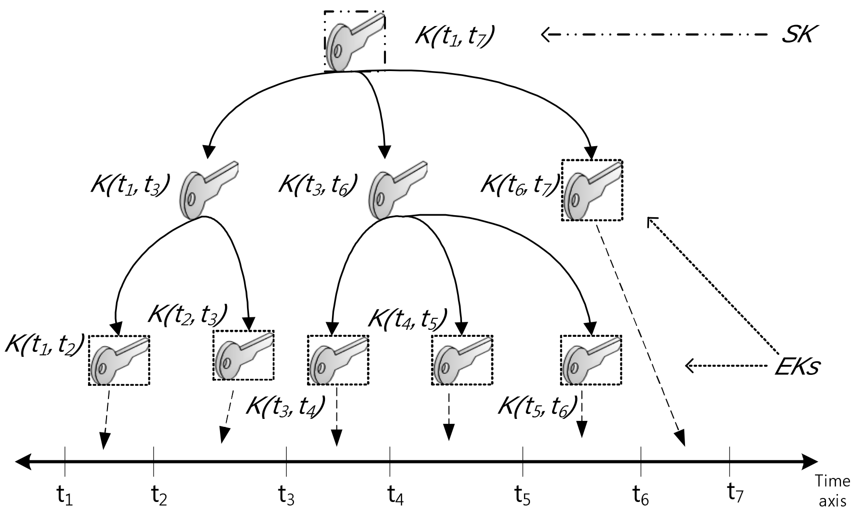

Figure 4.1: An illustration of an example for MDKD.

Definition 4.3.2 Static Period*: Let N_t be a valid vehicle set at time t. Time period, (a, b), is static if and only if the following requirements are satisfied for all possible t ($a < t < b$): $|N_t - N_a| + |N_a - N_t| = 0$, $|N_t - N_b| + |N_b - N_t| = 0$.*

Now, we define key validity as follows:

Definition 4.3.3 Key Validity *(K): Let K be a key. A key, $K(t_a, t_b)$, is valid in time period between t_a and t_b, ($t_a < t_b$) if and only if the KDC allocates the key, $K(t_a, t_b)$, in time period (t_a, t_b).*

For example, in Fig. 4.1, $K(t_1, t_2)$ and $K(t_1, t_3)$ are secret keys that are valid during the time period (t_1, t_2) and (t_1, t_3), respectively.

Among valid keys, not every valid key can be utilized as a GK, because of the confidentiality problem. Among them, only the *Elementary Key* (EK) can be used as a GK. The EK can be defined as the key located in a leaf of the key path. For example in Fig. 4.1, $K(t_1, t_2)$, $K(t_2, t_3)$, $K(t_3, t_4)$, $K(t_4, t_5)$, $K(t_5, t_6)$, and $K(t_6, t_7)$ are EKs.

Now, we introduce the Membership Dynamics based Key Derivation (MDKD). Based on the membership dynamics of the SPI, the KDC has to derive new valid keys.

Definition 4.3.4 Membership Dynamics based Key Derivation *(MDKD): Let t be a time information included in SPI, K be EK valid at time t, and $F_d(\cdot)$ be a secure derivation function. MDKD represents a new key derivation method by using vehicle dynamics information.*

Specifically, SPI includes membership dynamics such as joining time (t_j) and leaving time (t_l). If EK valid at t_j is same as EK valid at t_l, new EKs are derived as follows: $K(t_a, t_j) = F_d(K(t_a, t_b), 1)$, $K(t_j, t_l) = F_d(K(t_a, t_b), 3)$, and $K(t_l, t_b) = F_d(K(t_a, t_b), 2)$ ($t_a < t_j < t_l < t_b$, and $K(t_a, t_b)$ is an EK). Otherwise, new EKs are derived with t_j and EK valid at t_j, and with t_l and EK valid at t_l, respectively, as follows: $K(t_a, t_j) = F_d(K(t_a, t_b), 1)$ and $K(t_j, t_b) = F_d(K(t_a, t_b), 3)$ ($t_a < t_j < t_b$, and $K(t_a, t_b)$ is an EK); $K(t_c, t_l) = F_d(K(t_c, t_d), 1)$ and $K(t_l, t_d) = F_d(K(t_c, t_d), 3)$ ($t_c < t_l < t_d$, and $K(t_c, t_d)$ is an EK).

In MDKD, the derived keys stand for *children keys* (CK), and the original key that is used to derive the children keys stands for *parent key* (PK); so, we define the following sets: $prk(K) = \{K_p | K = F_d(K_p, i)\}$ ($i = 1, 2, 3$),

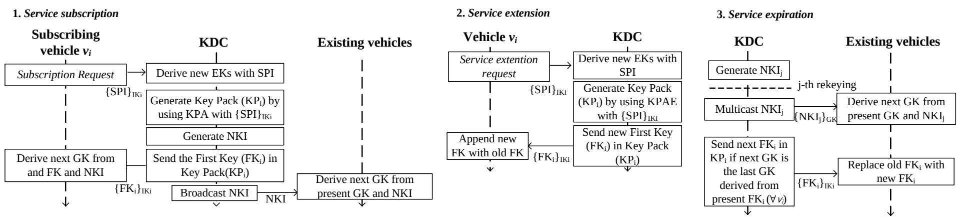
1. Service subscription
Subscribing vehicle v_i
KDC
Existing vehicles
Subscription Request
$\{SPI\}_{IKi}$
Derive new EKs with SPI
Generate Key Pack (KP_i) by using KPA with $\{SPI\}_{IKi}$
Generate NKI
Derive next GK from and FK and NKI
Send the First Key (FK_i) in Key Pack(KP_i)
$\{FK_i\}_{IKi}$
Broadcast NKI
NKI
Derive next GK from present GK and NKI
2. Service extension
Vehicle v_i
KDC
Service extention request
$\{SPI\}_{IKi}$
Derive new EKs with SPI
Generate Key Pack (KP_i) by using KPAE with $\{SPI\}_{IKi}$
Append new FK with old FK
$\{FK_i\}_{IKi}$
Send new First Key (FK_i) in Key Pack (KP_i)
3. Service expiration
KDC
Existing vehicles
Generate NKI_j
j-th rekeying
Multicast NKI_j
$\{NKI_j\}_{GK}$
Derive next GK from present GK and NKI_j
Send next FK_i in KP_i if next GK is the last GK derived from present FK_i ($\forall v_i$)
$\{FK_i\}_{IKi}$
Replace old FK_i with new FK_i

Figure 4.2: An illustration of MDKD scheme.

$crk(K) = \{K_c | K_c = F_d(K, i)\}$ $(i = 1, 2, 3)$, and $sbk(K) = \{K_i | prk(K) = prk(K_i)\}$ $(\textit{for all possible } K_i)$.

Here, $prk(K)$, $crk(K)$, and $sbk(K)$ stand for the parent key of a key K, the children key of a key K, and the sibling key of a key K, respectively. Based on the relation, it is noted that any key can not be an EK and a PK at the same time. Among PKs, there is a SK such that $|prk(SK)| = 0$. In other word, the SK is not the key derived from any key and can only be generated by the KDC. For example, in Fig. 4.1, $K(t_1, t_7)$ is the only SK.

To sum up, there are two ways to make keys. One is key derivation for deriving EKs by using the $F_d(\cdot)$. The other is key generation for deriving SKs by using the $G_k(\cdot)$.

4.3.2 Membership Dynamics based Key Management (MDKM)

In this subsection, we present Membership Dynamics based Key Management (MDKM), which has the following characteristics: (1) According to membership dynamics obtained from SPI, the KDC can be aware of the key updating schedule. Thus, over static period, the KDC does not need to update a GK. (2) Since unicast method utilizes little network resources

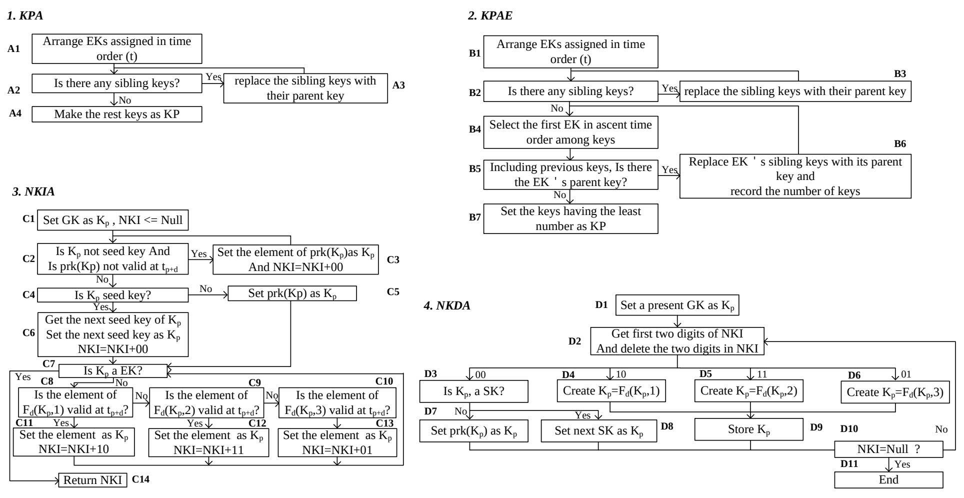
1. KPA
A1 Arrange EKs assigned in time order (t)
A2 Is there any sibling keys?
Yes
A3 replace the sibling keys with their parent key
No
A4 Make the rest keys as KP
2. KPAE
B1 Arrange EKs assigned in time order (t)
B2 Is there any sibling keys?
Yes
B3 replace the sibling keys with their parent key
No
B4 Select the first EK in ascent time order among keys
B5 Including previous keys, Is there the EK ' s parent key?
Yes
B6 Replace EK ' s sibling keys with its parent key and record the number of keys
No
B7 Set the keys having the least number as KP
3. NKIA
C1 Set GK as K_p , NKI <= Null
C2 Is K_p not seed key And Is prk(Kp) not valid at t_{p+d}
Yes
C3 Set the element of prk(K_p)as K_p And NKI=NKI+00
No
C4 Is K_p seed key?
No
C5 Set prk(Kp) as K_p
Yes
C6 Get the next seed key of K_p Set the next seed key as K_p NKI=NKI+00
C7 Is K_p a EK?
Yes
No
C8 Is the element of $F_d(K_p,1)$ valid at t_{p+d}?
No
C9 Is the element of $F_d(K_p,2)$ valid at t_{p+d}?
No
C10 Is the element of $F_d(K_p,3)$ valid at t_{p+d}?
C11 Yes
Set the element as K_p NKI=NKI+10
Yes
C12 Set the element as K_p NKI=NKI+11
C13 Set the element as K_p NKI=NKI+01
Return NKI C14
4. NKDA
D1 Set a present GK as K_p
D2 Get first two digits of NKI And delete the two digits in NKI
D3 00
Is K_p, a SK?
D4 10
Create $K_p=F_d(K_p,1)$
D5 11
Create $K_p=F_d(K_p,2)$
D6 01
Create $K_p=F_d(K_p,3)$
D7 No
Set prk(K_p) as K_p
Yes
Set next SK as K_p D8
Store K_p D9
D10 No
NKI=Null ?
D11 Yes
End

Figure 4.3: An illustration of algorithms for MDKD.

than multicast method, the proposed scheme utilizes unicast in delivering Key Pack (KP). The KP consists of some keys that are utilized in deriving all the GKs over specific SPI. Because a SPI of a vehicle is different from those of other vehicles, it is resource-efficient to transmit a KP by using unicast method. (3) The proposed scheme minimizes the amount of message to be multicast. Since each vehicle already has their own key set delivered by unicast, the KDC multicasts a simplified Next Key Information (NKI) which is utilized to derive a next GK. As a result, the proposed scheme can reduce the amount of network resource usage.

At first, the KDC has to generate SKs, which are periodically assigned to seed key period P_s. Then, the KDC efficiently manages keys by using the proposed methods according to the membership dynamics. These dynamics are caused from membership events called *Service subscription*, *Service* *extension*, and *Service expiration*.

Service Subscription

In case of a service subscription the KDC manages keys as shown in Fig. 4.2. When a vehicle (v_i) wants to subscribe to a service, the vehicle has to send a *subscription request* message including its SPI including joining time and leaving time. The KDC derives new EKs with the SPI. Then the KDC generates KP including the minimized number of keys that can derive all EKs in the time period between the joining time and leaving time by using the *Key Packing Algorithm* (KPA), which will be given in Fig. 4.3. In order of validity time, the keys in KP is supposed to be delivered to the subscribing vehicle. Among the keys in KP, the First Key (FK), which is denoted as the key having the earliest valid time, is delivered to the subscribing vehicle. By using the FK, the vehicle can derive a GK with one-way function. To provide the confidentiality of the FK message the KDC has to send the FK encrypted with the vehicle's IK. After sending the FK, the FK is deleted in KP.

- Key Packing Algorithm (KPA)

Fig. 4.3 shows the Key Packing Algorithm (KPA) used to generate Key Pack (KP). By using KPA as shown in Fig. 4.3, the KDC can generate a specific KP consisting of some keys that are utilized in deriving all the GKs in a specific subscription period. For example, when a new vehicle wants to subscribe over the period of Fig. 4.4, the KDC is supposed to generate KP by using KPA as follows:

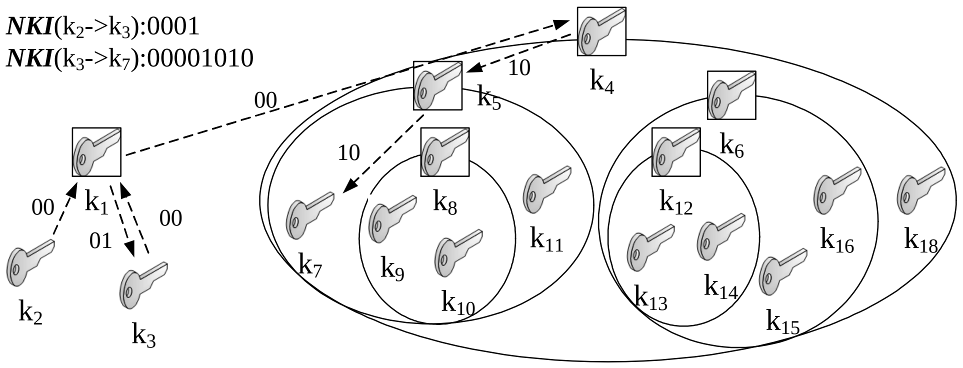

Figure 4.4: An illustration of an example for KPA, NKIA, and NKDA.

- A1: The KDC arranges EKs during SPI in time order. KP_x=\{k_2, k_3, k_7, k_9, k_{10}, k_{11}, k_{13}, k_{14}, k_{15}, k_{16}, k_{18}\}
- A3: Replace k_2 and k_3, k_9 and k_{10}, and k_{13} and k_{14} (the sibling keys) with k_1, k_8, and k_{12} (their PK), respectively. Then, KP_x=\{k_1, k_7, k_8, k_{11}, k_{12}, k_{15}, k_{16}, k_{18}\}
- A2, A3: Because there are still sibling keys in KP_x, replace k_7, k_8, and k_{11} and k_{12}, k_{15}, and k_{16} with k_5 and k_6, respectively. KP_x=\{k_1, k_5, k_6, k_{18}\}.
- A2, A3: There are still sibling keys (k_5, k_6, and k_{18}). So, They are replaced with k_4. KP_x=\{k_1, k_4\}
- A4: Finally, the KDC makes KP from the KP_x. (KP=\{k_1, k_4\}).

Service Extension

In case of a service extension, the KDC manages keys as shown in Fig. 4.2. When a vehicle ($v_i \in N_t$) wants to extend a service, the vehicle has to send a *service extension request* message including its SPI including new leaving time. The KDC derives new EKs with the SPI. Then the KDC generates KP covering the extended time period by using *Key Packing Algorithm for Extension* (KPAE), which will be given in Fig. 4.3. The KPAE generates KP similar to the way that KPA does. But, KPAE can further reduce the size of KP by utilizing the previous keys that the vehicle already has. The rest of the procedure is very similar to that of service subscription as shown in Fig. 4.2. Among the keys in KP, the First Key (FK) is delivered to the vehicle as well. By using the FK, the vehicle can derive a next GK with one-way function. To provide the confidentiality of the FK message the KDC has to send the FK encrypted with the vehicle's IK. After sending the FK, the FK is deleted in KP.

- Key Packing Algorithm for Extension (KPAE)

Fig. 4.3 shows the Key Packing Algorithm for Extension (KPAE) used to generate KP, when the subscribed vehicle wants to extend a service. KPAE includes the minimization algorithm utilizing the keys that the vehicle already has. Because the subscribed vehicle was aware of previous keys, KPAE does not violate backward confidentiality. For example, if a subscribed vehicle (from t_0 to t_6) wants extend a service from t_6 and t_{27} as illustrated in Fig. 4.5, the KDC has to generate KP by utilizing KPAE as follows:

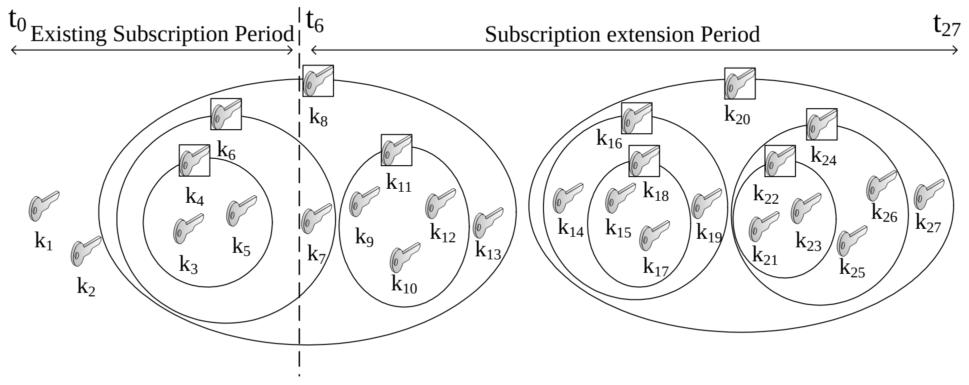

Figure 4.5: An illustration of an example for KPAE.

- B1-B3: The KDC generates KP_0={k_7, k_{11}, k_{13}, k_{20}}.
- B4: Among elements of KP_0, k_7 is selected.
- B5: Because the PK of k_7 is k_6,
- B6: replace k_7 with k_6. KP_1={k_6, k_{11}, k_{13}, k_{20}}. And, $|KP_1|$=4.
- B4: Among elements of KP, k_6 is selected.
- B5: Also k_8 is the PK of k_6.
- B6: Replace k_6, k_{11}, and k_{13}(Sibling keys of k_6) with k_8. KP_2={k_8, k_{20}}. And, $|KP_2|$=2.
- B5: Since k_8 is a SK,
- B7: Among the KP_x(x=0,1,2) obtained from each steps, the KDC selects a KP having the least number of elements. Thus, KP_2 is selected as a KP.

Service Expiration

In the case of service expiration of a certain vehicle, the KDC manages keys as shown in Fig. 4.2. Based on vehicles' SPI, the KDC is already aware of dynamics in a service group and the all relations of keys. Before present static period is over, the KDC generates NKI for existing subscribed vehicles to derive a next EK by using NKIA as shown in Fig. 4.2. Then, the KDC multicasts the NKI just before a present static period ends. After receiving the NKI, vehicles can successfully derive the next GK in their own FK with NKDA, shown in Fig. 4.3.

For some vehicles, their FK can be expired. Thus, to derive following GKs, the KDC has to deliver next FK in their KP to the specific vehicles. For the confidentiality of the FK message the KDC has to send the FK encrypted with the vehicle's IK. After sending the FK, the FK is deleted in KP.

• Next Key Information Algorithm (NKIA)

NKI is the path information used to derive a next EK from a present EK. NKI consists of four directions (*up*, *left*, *right*, and *middle*), and these four directions are supposed to deliver with the predesignated codes as 00, 10, 01, and 11, respectively. *up*, *left*, *right*, and *middle* of k indicates $prk(k)$, $F_d(k,1)$, $F_d(k,2)$, and $F_d(k,3)$, respectively. By way of exception, *up* of SK indicates next SK. The specific procedure of algorithm for generating NKI is as shown in Fig. 4.3.

For example, when a present GK is k_3 in Fig. 4.4, the KDC can generate NKI by using NKIA as follows:

- C1: Set a GK, k_3, as K_p. KP=().
- C2: k_3 is not SK, and prk(k_3), k_1, is not valid at $t_{p+\delta}$.
- C3: Set k_1 as K_p. And NKI=(00).
- C2, C4: Since k_1 is a SK,
- C6: get the next SK of k_1, k_4, and set k_4 as K_p. And NKI=(0000).
- C7: k_4 is not a EK.
- C8: Since k_5, $F_d(K_p, 1)$, is valid,
- C11: set k_5 as K_p, and NKI=(000010).
- C7: k_5 is also not a EK.
- C8: Since k_7, $F_d(K_p, 1)$, is valid as well,
- C11: set k_7 as K_p, and NKI=(00001010).
- C7: Because k_7 is a EK,
- C14: finally the KDC obtains NKI.

• Next Key Derivation Algorithm (NKDA)

When vehicles receive NKI from the KDC, they start to derive a next GK by using NKDA. Figure 4.3 shows the NKDA for deriving a new GK from a present GK with own FK. Let us assume that a present GK is k_3; a vehicle receives NKI of 00001010; and the vehicle is aware of $k_1,\ k_3,\ k_4$ as shown in Fig. 4.4. The vehicle can derive the next GK by using NKDA as follows:

- D1: A vehicle set k_3 as K_p
- D2: The first digits of NKI is 00 and delete the digits in NKI.
- D3, D7: Since K_p is not a SK, prk of K_p, k_1, is set as K_p.
- D10, D2: Because NKI(001010) is not an empty, get the first digit (00).
- D3, D8: Since K_p is a SK, next FK, k_4, is set as K_p.
- D10, D2: Because NKI(1010) is still not an empty, get the first digit (10).
- D4: Derive a child key of k_4, k_5=$F_d(k_4, 1)$.
- D10, D2: Since NKI(10) is not an empty, get the first digit (10) as well.
- D4: Derive a child key of k_5, k_7=$F_d(k_5, 1)$.
- D10, D11: Because NKI() is an empty, a vehicle can set k_7 as a next EK and utilize the EK as a GK.

Early Service Leaving

Sometimes, a vehicle (a driver) can request to leave a service before the its leaving time, i.e. early leaving. Considering the early leaving case, the proposed MDKM allows a vehicle to leave a service in the validity time unit of FK. In MDKM, the KDC is supposed to deliver an one FK at a time. Thus, by just deleting his KP and not delivering next FK to the vehicle further, the vehicle can leave service without breaking the service confidentiality.

4.4 Cost Analysis

This section gives the numerical performance analysis of the proposed MDKM with various aspects such as computation, storage and communication costs.

4.4.1 Computation Cost

Server: When a vehicle subscribes to a service, a vehicle sends a subscription request including their SPIs to the KDC. Each SPI consists of join time or leave time. To support service security against a new vehicle's dynamics, the KDC has to generate new keys corresponding to the new *static periods.* By using one time information included in SPIs, the KDC can derive two child keys ($K_i^1 = F_d(K_i, 1)$ and $K_i^3 = F_d(K_i, 2)$). Since time difference between joining time and leaving time included in a subscription request is normally greater than average static period, the KDC generally generates four keys per a single vehicle's subscription. Therefore the computation cost of a server is $U^s \simeq 4|N_t|/E[P_v]$. Here P_v is the subscription period of a vehicle.

Vehicle: Vehicles should be unaware of dynamics of other vehicles for guaranteeing privacy. Whenever a new key is adopted (or before a present static period end), vehicles must receive the new NKI for deriving a next EK. On average, the number of key derivations for vehicles is the same as that of the KDC. Since each vehicle causes two dynamics, the computation cost of a vehicle is U^v=U^s=$4|N_t|/E[P_v]$.

4.4.2 Storage Cost

Server: Whenever a vehicle subscribes to a service, the KDC has to generate four keys by using the joining time and leaving time included in a subscription request. Thus the number of keys to be stored in a sever(S^s) is $4|N_t|$.

Vehicle: Storage cost of vehicles (S^v) consists of three components: One comes from the depth (AD) of its first EK (S^{vb}); Another comes from the number of SKs within the joining time and leaving time (S^{vs}); The other comes from the depth of its last EK (S^{ve}); According to characteristics of Poisson distribution, it is likely that the vehicles' subscription time is well balanced. So, S^{vb}=S^{ve}. Thus S^v=S^{vs}+$2AD$. Let us firstly analyze the AD.

When the number of dynamics in a single SK is k, the number of EK is $k+1$. Note that the sum of the depth of EKs with k dynamics under a single SK is D_k and the average depth of GK is AD_k $(= D_k/k+1)$. For additional one dynamics, D_k increases as much as $AD + 2$ $(2(d + 1) - d = d + 2)$. According to the relation between additional dynamics and the sum of the depth of GKs in a single SK, we can derive the following recurrence formula.

$$D_k = D_{k-1} + AD_k + 2 = \frac{k+1}{k} D_{k-1} + 2 \tag{4.1}$$

If we further expand the Eq. 4.1, we can obtain the following equations.

$$\begin{aligned} D_k &= \frac{k+1}{k}\left(\frac{k}{k-1} D_{k-2} + 2\right) + 2 \\ &= \frac{k+1}{k-1} D_{k-2} + 2\frac{k+1}{k} + 2 \end{aligned} \tag{4.2}$$

If we expand the Eq. 4.2 until k becomes zero ($k = 0$ means that there is no dynamics), we can obtain the equations as follows:

$$\begin{aligned} D_k &= \frac{k+1}{1} D_0 + 2(k+1)\left(\frac{1}{k} + \frac{1}{k-1} + \cdots \frac{1}{2}\right) + 2 \\ &= 2(k+1)\sum_{i=1}^{k}\left(\frac{1}{i}\right) - k + 1 \ (D_0 = 1) \end{aligned} \tag{4.3}$$

Thus, the average depth of GKs can be obtained by dividing AD_k by $(k+1)$ as $AD_k = D_k/(k+1)$. Since $k{=}|N_t|P_s/E[P_v]$, the storage cost of vehicles are obtained as follows:

$$\begin{aligned} S^v &= S^{vs} + 2AD_k = \frac{1}{2}\left(\frac{E[P_v]}{P_s}\right) - \frac{1}{2} + 2AD_k \\ &= \frac{1}{2}\left(\frac{E[P_v]}{P_s}\right) + 4\sum_{i=1}^{\frac{|N_t|P_s}{E[P_v]}}\left(\frac{1}{i}\right) + \frac{4E[P_v]}{|N_t|P_s + E[P_v]} - \frac{5}{2} \end{aligned} \tag{4.4}$$

4.4.3 Communication Cost

MDKM includes two kinds of communication costs. First, when subscribing to a service, the vehicle is supposed to receive a unicast message containing the optimized key set (KP). Second, when there is a membership dynamics caused by other vehicles' service subscription or expiration, vehicles are supposed to receive multicast messages containing NKI, which is utilized to derive a next EK. Thus, this subsection investigates the two communication costs of unicast cost and multicast cost, respectively.

Unicast cost

When a vehicle subscribes to a service, the KDC sends the set of keys (KP) corresponding to its SPI to the vehicle. If the valid period of a SK is contained within that of a vehicle subscription period, the SK is included in the KP. So, unicast overhead (C^u) consists of three components: One comes from time period between the joining time and the expiration time of SK valid at the joining time (C^{ub}). Another comes from the number of SKs within the SPI (C^{us}). The other comes from the time period between the beginning time of SK valid at the expiration time and the vehicle's service expiration time (C^{ue}). According to characteristics of the Poisson distribution, it is likely that the vehicles' subscription time is well balanced. As a result, the overhead of the former time period is the same as that of the latter time period on average. Thus the unicast overhead is as follows:

$$C^u = C^{ub} + C^{us} + C^{ue} = C^{us} + 2C^{ue} \tag{4.5}$$

In addition, it is very likely that the service subscription period is much longer than the average key update period. Thus, keys under a single SK can be normally maintained with 2-ary relation. Given "k" level 2-ary relation, there are 2^k elementary keys under a single SK.

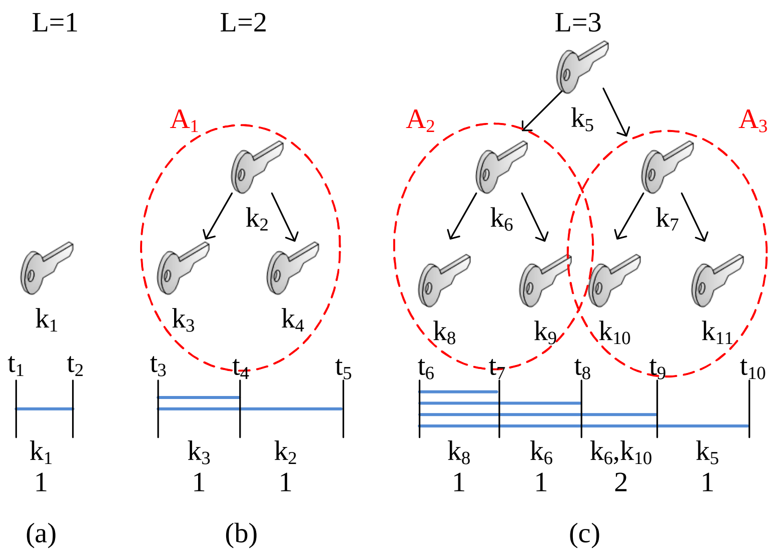

Figure 4.6: An illustration of an example for unicast communication cost of KPA.

Fig. 4.6 shows an example of key relations in KP. (a), (b), and (c) consist of structured keys, where, k_1, k_2, and k_5 are SK, which are valid for (t_1, t_2), (t_3, t_5), and (t_6, t_{10}), respectively. In the case that vehicles are supposed to subscribe to a service before t_1, t_3, and t_6, and leave before t_2, t_5, and t_{10}, respectively, the number of keys in the KP under a single SK is analyzed.

In Fig. 4.6 (a), there is no dynamics between t_1 and t_2. So, when a vehicle leaves at time t_2, the number of keys in the KP in the period will be one (k_1). In case of L=2 (layer = 2), there are two cases of vehicles' leaving. If a vehicle is supposed to leave at t_4, the number of keys in the KP is one (k_3). If a vehicle is supposed to leave at t_5, the number of keys in KP is one as well (k_2), because a vehicle can derive low layered keys from k_2

(k_3 and k_4). In the same way, the number of keys in the KP can be simply calculated according to vehicles' service expiration times as shown in Fig. 4.6 (c).

When comparing between A_1 and A_2 in Fig. 4.6, the structure of A_1 is same as that of A_2. In the case that a subscription time is the time period between t_8 and t_{10}, the KP includes k_6, which is on the top of A_2, and a key in A_3 which has the same structure of A_1. For example, when the KDC has to make the KP valid for (t_6, t_9), k_6 of A_2 and k_{10} of A_3 are included in the KP. By way of exception, when the time period covers the entire period of a SK (e.g., the time period between t_6 and t_{10}), the KP includes not lower layer keys but a SK, (e.g., not k_6 and k_7 but k_5). From the relations, we can derive the recurrence formula as follows:

$$C_L^u = 2C_{L-1}^u + 2^{L-2} - 1. \ (C_1^u = 1) \tag{4.6}$$

Here, C_L^u is the sum of the number of keys against all possible KP cases, and L is the number of layers of keys.

$$\begin{aligned} C_L^u &= 2C_{L-1}^u + 2^{L-2} - 1 \\ &= 2(2C_{L-2}^u + 2^{L-3} - 1) + 2^{L-2} - 1 \\ &= 2^2 \cdot C_{L-2}^u + 2 \cdot 2^{L-2} - 1 - 2 \end{aligned} \tag{4.7}$$

If we continuously expand the equation until L is one, the following equation is obtained as $C_L^u = 2^{L-1} \cdot C_1^u + (L-1) \cdot 2^{L-2} - 2^{L-1} + 1$. Since $C_1^u = 1$ in Eq. (4.6), Eq. (4.7) can be simplified as follows:

$$C_L^u = (L-1) \cdot 2^{L-2} + 1 \tag{4.8}$$

Thus $\frac{C^u_L}{2^L}$ is the average number of keys for generating the KP, which stands for C^{ue}. Note that $2^{L-1} - 1$=*the average number of dynamics in a single SK*=$2N_t/(\frac{E[P_v]}{P_s})$. P_s is the period of a SK. Therefore, from the Eq. 4.5 and Eq. 4.8, we can obtain the unicast communication overhead, the average number of keys in KP with heavy traffic scenario ($2^{L-1} >> 1$), as:

$$\begin{aligned} C^u = C^{us} + 2C^{ue} &\approx \left\lceil \frac{E[P_v]}{P_s} \right\rceil - \frac{1}{2} + 2\left(\frac{L-1}{2}\right) \\ &= \left\lceil \frac{E[P_v]}{P_s} \right\rceil + \frac{1}{2} + \left(log_2 \frac{|N_t|P_s}{E[P_v]}\right). \end{aligned} \tag{4.9}$$

Multicast cost

Since each vehicle has the keys used to derive new EKs over their subscription period, multicast overhead comes from the only NKI messages issued whenever vehicle dynamics (join and leave) exist.

When a vehicle subscribes to a service, the next GK is derived from a present EK based on the subscription time of a vehicle. Each subscription time information in the SPI makes vehicles perform one key derivation. Since each key derivation causes one key derivation equivalent to two directions, the overhead per vehicle subscription is 4 bits (2 bits per direction). On the other hand, in the case of service expiration of a vehicle, each expiration time information in the SPI makes vehicles perform two key derivations. Thus overhead per service expiration is 8 bits.

Since there are $|N_t|$ vehicles during $E[P_v]$ period, the multicast message overhead is as follows:

$$C^{mo} = 12(bits)/\frac{E[P_v]}{|N_t|} = \frac{12|N_t|}{E[P_v]}(bits). \tag{4.10}$$

Since the multicast message utilizes much more network resources, the multicast cost have to be considered the network overhead. Chuang *et al* have researched the network resource usage of multicast, which is $|groupsize|^{0.8}$ times greater than that of unicast [50]. Here, $|groupsize|^{0.8}$=Θ is the network overhead ratio between unicast message and multicast message. Thus, the multicast cost is as follows:

$$C^{m} = \Theta \cdot C^{mo} = |N_t|^{0.8} \cdot C^{mo}. \tag{4.11}$$

Communication Cost

The proposed scheme utilizes both unicast and multicast methods. Thus the communication cost of the proposed scheme can be the sum of the multicast cost and unicast cost [48], [49]. Therefore, we can obtain the communication cost as follows.

$$C^{mdkm} = C^{u} + C^{m}. \tag{4.12}$$

4.5 Cost Optimization

The previous section presents the computation, storage and communication costs, which depend on various parameters, including the number of vehicles $|N_t|$, the subscription period P_v, and the SK period P_s. By controlling the parameters, we can model vehicles' dynamics and set up various situations in vehicular environments.

In this section, to statistically analyze the performance of proposed scheme, we develop M/G/∞ system of vehicle membership dynamics in vehicular environments. Based on the statistical analysis, we solve a cost optimization problem as a design problem.

4.5.1 M/G/∞ system

At first we model vehicle dynamics as a statistical queening system. In this model we assume that the vehicle subscription rate is a *Poisson* distribution with mean rate λ_s, and the vehicle subscription period follows a *gaussian distribution* $G(\cdot)$ with the mean $E[P_v]$ and standard deviation D_v. Generally, it is well known that the assumptions are used in modeling group behavior [46], [47].

Since vehicles' dynamics are independent to each other's demands, vehicles' dynamics can be modeled as a M/G/∞ system, which is a starting point for statistical approach. Therefore, the probability that the number of vehicles in a group at time t can be calculated as follows:

$$\begin{aligned} P_k(t) &= Pr[N(t) = k] \\ &= \sum_{n=k}^{\infty} P_k(t|V(t) = n) \cdot \frac{e^{-\lambda_s t}(\lambda_s t)^n}{n!}. \end{aligned} \tag{4.13}$$

$N(t) = k$ represents that k vehicles who subscribed before time t are in service. Let us consider a single vehicle, v_i, who subscribed before time t.

Pr[v_i subscribed at time x, and v_i is in service at time x | v_i subscribed at time $(0, t]$] = Pr[v_i subscribed at time $(x, x + dx)$| v_i subscribed at time $(0, t]$]$\cdot$Pr[v_i is in service at time t | v_i arrived at time $(0, t]$]=$1/tdx$ $\cdot$ Pr[v_i's service is not over during the time period $(t - x)$]=$1/t \cdot [1 - G(t - x)]dx$.

If we substitute y for x, we can obtain an equation as follows:

$$\int_0^t \frac{1}{t} \cdot [1 - G(t-x)]dx = \frac{1}{t} \int_0^t [1 - G(y)]dy. \tag{4.14}$$

Let us expand the single vehicle case to n numbers of vehicles case. $V(t) = n$ represents that n vehicles subscribed during the time period $(0, t]$. The number of cases that k vehicles are still in service among the n vehicles is the same as those of picking out k samples among identically same n ones. Thus, the probability in Eq. 4.13 can be calculated as follows:

$$\begin{aligned} &P_k(t|V(t) = n) \\ &= C_k^n \cdot \left[\frac{1}{t}\int_0^t [1 - G(y)]dy\right]^k \left[1 - \frac{1}{t}\int_0^t [1 - G(y)]dy\right]^{n-k} \end{aligned} \tag{4.15}$$

By adapting the result of Eg. 4.15 to Eq. 4.13, we can obtain the following results.

$$\begin{aligned} P_k(t) = &\sum_{n=k}^{\infty} C_k^n \cdot \left[\frac{1}{t}\int_0^t [1 - G(y)]dy\right]^k \\ &\cdot \left[1 - \frac{1}{t}\int_0^t [1 - G(y)]dy\right]^{n-k} \cdot \frac{e^{-\lambda_s t}(\lambda_s t)^n}{n!} \end{aligned} \tag{4.16}$$

If we substitute $x + k$ for n in Eq. 4.16, Eq. 4.16 can be given as follows:

$$\begin{aligned} &\sum_{x=0}^{\infty} \frac{(x+k)!}{k!(x)!} \cdot \left[\frac{1}{t}\int_0^t [1 - G(y)]dy\right]^k \\ &\cdot \left[1 - \frac{1}{t}\int_0^t [1 - G(y)]dy\right]^x \cdot \frac{e^{-\lambda_s t}(\lambda_s t)^{x+k}}{(x+k)!} \end{aligned} \tag{4.17}$$

Eq. 4.17 can be further simplified as follows:

$$\begin{aligned} &\frac{[\lambda_s \int_0^t [1 - G(y)]dy]^k}{k!} \\ &\cdot \sum_{x=0}^{\infty} \frac{[\lambda_s t - \lambda_s \int_0^t [1 - G(y)]dy]^x \cdot e^{-\lambda_s t}}{x!}. \end{aligned} \tag{4.18}$$

According to the taylor series of exponential function ($e^x = \sum\limits_{n=0}^{\infty} \frac{x^n}{n!}$), Eq. 4.18 is following to

$$\frac{[\lambda_s \int_0^t [1 - G(y)]dy]^k}{k!} \cdot e^{-\lambda_s \int_0^t [1-G(y)]dy} \tag{4.19}$$

As a result, we can obtain the following equation.

$$P_k(t) = \frac{exp[-\lambda_s \int_0^t [1 - G(y)]dy] \cdot [\lambda_s \int_0^t [1 - G(y)]dy]^k}{k!} \tag{4.20}$$

Eq. 4.20 is a Poisson distribution with the mean of $\lambda_s \int_0^t [1 - G(y)]dy$. Thus, we can calculate the probability of the number of vehicles in a steady state as follows:

$$P_k = \lim_{t->\infty} P_k(t) = \frac{e^{-\rho_s}(\rho_s)^k}{k!}, \quad (\rho_s = \lambda_s E[P_v]). \tag{4.21}$$

Since the subscription and extension are independent, just by replacing λ_s with λ, we can obtain the results. Thus, the number is M/G/∞ system obeys a Poisson distribution with mean of $\rho = \lambda E[P_v]$, λ=λ_s+λ_e, where λ_e is the vehicle extension rate. From the characteristics of Poisson distribution, $E_{P_k}[k]$=ρ=$\lambda E[P_v]$=$|N_t|$.

If we apply the results of M/G/∞ to the three kinds of costs in the previous section, we can transform the costs into statistical M/G/∞ cost.

As a result, the computation cost of a server and vehicle per unit time is $\overline{U^s}$=$\overline{U^v}$=4λ.

The storage cost of a server per unit time is $\overline{S^s}$=$\lambda E[P_v]$.

The storage cost of a vehicle per unit time is

$$\overline{S^v} = \frac{1}{2}\left(\frac{E[P_v]}{P_s}\right) + 4\sum_{i=1}^{\lambda P_s}\left(\frac{1}{i}\right) + \frac{4}{\lambda P_s + 1} - \frac{5}{2}.$$

The unicast communication cost per unit time is

$$\overline{C^u} = \left[\frac{E[P_v]}{P_s}\right] + \frac{1}{2} + \left(log_2 \lambda P_s\right).$$

The multicast communication cost per unit time is $\overline{C^m} = 12\lambda(\lambda \cdot E[P_v])^{0.8}(bits)$.

Therefore the communication cost can be obtained as follows:

$$\begin{aligned}\overline{C^{mdkm}} = \overline{C^u} + \overline{C^m} =& [E[P_v]/P_s] + \frac{1}{2} + (log_2 \lambda P_s) \\ &+ 12(E[P_v])^{0.8} \cdot (\lambda)^{1.8}.\end{aligned} \tag{4.22}$$

4.5.2 Key Management Cost

In this subsection, we analyze the key management cost of MDKM (KMC^{mdkm}) as the weighted sum of each cost. To sum up the different costs as analyzed in the previous subsections, we adopt the coefficient to apply appropriate weight according to the proposed model: γ_S^s and γ_S^v denote the cost required to store a unit key in a server and vehicles, respectively. γ_U^s and γ_U^v denote the cost required to compute a unit computation by a server and vehicle, respectively. γ_C^u and γ_C^m denote the cost required to transmit a unit data by unicast and multicast, respectively. Then we can derive the key management cost as follows.

$$\begin{aligned}\overline{KMC^{mdkm}}(P_s) =& \gamma_S^s\overline{S^s} + \gamma_S^v\overline{S^v} + \gamma_U^s\overline{U^s} + \gamma_U^v\overline{U^v} \\ &+ \gamma_C^u\overline{C^u} + \gamma_C^m\overline{C^m}\end{aligned} \tag{4.23}$$

4.5.3 Cost Optimization

Now, we are going to formulate the design problem to optimize the weighted sum of the costs of MDKM which are analyzed in the previous section. According to the results of the previous subsection, $\overline{KMC^{mdkm}}(P_s)$ can be written as a simplified function with the design parameter P_s as follows:

$$\overline{KMC^{mdkm}}(P_s) \simeq \alpha\frac{1}{P_s} + \beta\log P_s + \delta\ (\alpha, \beta, \delta \in \mathbf{R}+). \tag{4.24}$$

Finally we define the cost optimization as below:

$$\overline{KMC^{mdkm}}(P_s^*) = \underset{\alpha,\beta,\delta\in R+}{\arg\min}\ \overline{KMC^{mdkm}}\,(P_s) \quad (P_s \in \mathbf{Z}+). \tag{4.25}$$

Since $\nabla\overline{KMC^{mdkm}}(P_s) = 0$ when P_s=α/β, $\overline{KMC^{mdkm}}(\cdot)$ is optimized when P_s^*=α/β.

4.6 Simulation

This section evaluates the performance of the proposed MDKM with various aspects in vehicular group service scenario. As the metric for evaluating the performance, communication cost, computation cost, and storage cost are used as measures, while a server operates a service and manages a GK by using the proposed method. Among the costs, computation and storage costs will be evaluated with a server (KDC) and vehicles aspects, respectively. Since each cost can be varied with various simulation parameters, we evaluate the three kinds of costs caused by changes in parameter values such as the mean of service subscription period ($E[P_v]$), the standard deviation of service subscription period (D_v), and the subscription rate (λ). Finally, in terms of key management cost, the MDKM scheme is compared with other schemes.

4.6.1 Simulation Setup

In order to evaluate the performance of the MDKM, we develop the simulation by using Microsoft Visual Studio C++. In the simulation, we assume vehicular multicast services. While various kinds of cars such as commuting car, bus and truck are moving around within vehicular service area, some of them are supposed subscribe to the vehicular service group. In the vehicular service group, vehicle subscription rate is modeled with *Poisson distribution*, and the service subscription period of each vehicle are modeled as *Gaussian distribution*. Since drivers' moving behavior is not affected by those of other drivers, it is reasonable to assume that each vehicle's dynamics are also independent from those of each other in the vehicular service group. As ciphering and derivation algorithm, we adopted *Advanced Encryption Standard* (AES) and SHA-1, and the size of keys is 128 bits [**?**] [54].

4.6.2 Computation Cost

In this subsection, we evaluate the computation costs of a server and vehicles with the variable λ. Table 4.1 shows that the average number of computations per minute is below 50 times in the all cases, when both a server and vehicle derive a new GK respectively. Since each AES operation and

Table 4.1: Computation cost of MDKM scheme ($\overline{U^s}$, $\overline{U^v}$)

λ (/min)	2.4	4.8	7.2	9.6	12.0
$\overline{U^s}$, $\overline{U^v}$ (The number of computations per min)	9.54	19.21	28.9	38.43	48.01

Table 4.2: Storage cost of a server of MDKM scheme ($\overline{S^s}$)

$\overline{S^s}$ (bits)		λ	(/min)			
		2.4	4.8	7.2	9.6	12.0
$E[P_v]$	333	410k	819k	1.23m	1.64m	2.05m
(min)	667	819k	1.64m	2.46m	3.28m	4.10m
	1000	1.23m	2.48m	3.69m	4.92m	6.14m
	1333	1.64m	3.28m	4.92m	6.55m	9.22m
	1667	2.05m	4.10m	6.14m	8.19m	10.2m

SHA-1 operation is done under a few micro seconds and a few nano second in the vehicular environment, the computation delay in deriving new keys is approximately negligible [51].

Table 4.3: Storage cost of a vehicle of MDKM scheme ($\overline{S^v}$)

$\overline{S^v}$ (bits)		λ	(/min)			
		2.4	4.8	7.2	9.6	12.0
$E[P_v]$	333	3.16k	3.51k	3.71k	3.86k	3.97k
(min)	667	4.44k	4.79k	4.99k	5.14k	8.25k
	1000	5.72k	6.07k	6.27k	6.42k	6.53k
	1333	7.00k	7.35k	7.55k	7.70k	7.81k
	1667	8.28k	8.63k	8.83k	8.98k	9.09k

4.6.3 Storage Cost

In this subsection, we evaluate the storage costs of a server ($\overline{S^s}$) and vehicles ($\overline{S^v}$) with two parameters, such as the variables of the vehicle subscription rate (λ) and the mean of vehicle subscription period ($E[P_v]$).

We first examine the effect of λ. Table 4.2 shows $\overline{S^s}$, where the x-axis is $E[P_v]$. Since the number of computations comes from the vehicle dynamics, we can observe that $\overline{S^s}$ increases proportional to the λ. Considering that the key size is 128 bits as explained, the amount of keys to be stored is not heavy to a key server. Table 4.3 shows $\overline{S^v}$. It is obvious that the λ does not affect $\overline{S^v}$ different from $\overline{S^s}$, because the new keys from λ can be derived from the minimized keys in KP. Furthermore, the amount of keys to be stored in each vehicle is not heavy either.

Next, we investigate the effect of $E[P_v]$. In Table 4.2, $\overline{S^s}$ is proportional to the $E[P_v]$, because the number of vehicles in a group is proportional to the $E[P_v]$ in given the fixed λ. This trend is also observed in $\overline{S^v}$ as shown in Table 4.3, because of the number of SKs in KP that is proportional to $E[P_v]$.

4.6.4 Communication Cost

The communication cost is decomposed into multicast cost from delivering the KP and unicast cost from delivering the NKI. In this subsection, we investigate each cost respectively.

Table 4.4: Unicast cost of MDKM scheme ($\overline{C^u}$)

$\overline{C^u}$ (bits per min)					
$E[P_v]$ (min)	167	333	500	667	833
$\overline{C^u}$	21.6k	32.2k	42.8k	51.1k	63.1k
D_v (min)	16.7	33.3	50.0	66.7	83.3
$\overline{C^u}$	42.8k	42.5k	42.5k	42.4k	41.9k
λ (/min)	2.4	4.8	7.2	9.6	12.0
$\overline{C^u}$	6.4k	13.5k	21.3k	29.2k	37.4k

Table 4.5: Multicast cost of MDKM scheme ($\overline{C^m}$)

$\overline{C^m}$ (bits per min)					
$E[P_v]$ (min)	167	333	500	667	833
$\overline{C^m}$	43.2k	73.7k	100.8k	124.9k	146.80k
D_v (min)	16.7	33.3	50.0	66.7	83.3
$\overline{C^m}$	100.42k	100.81k	100.35k	100.14k	99.00k
λ (/min)	2.4	4.8	7.2	9.6	12.0
$\overline{C^m}$	5.8k	20.3k	42.8k	72.3k	108.3k

Unicast Cost

Table 4.4 shows the unicast communication cost ($\overline{C^u}$) according to the mean of vehicle subscription period ($E[P_v]$). This result shows that the number of keys in the KP to be delivered is proportional to $E[P_v]$. In addition, the table shows $\overline{C^u}$ as the deviation of vehicle subscription period (D_v) varies, which shows that the performance is not much affected by the variable. According to the results, we can observe that the proposed scheme requires little unicast communication cost, only less 0.6 kbps when $E[P_v]$ is about 500 minutes.

In Table 4.4, $\overline{C^u}$ is $O(\lambda)$, while those of conventional schemes are $O(\lambda \log \lambda)$. $O(\lambda)$ is caused from the number of updating GK according to the vehicle subscription, which means that $\overline{C^u}$ is scalable ($O(1)$) at a single GK update. Therefore, the proposed scheme has no *1-affect-all problem* in terms of unicast cost, which is a critical obstacle in designing GKM schemes, while the conventional schemes requires $O(\log \lambda) = O(\log |N_t|)$ ($|N_t| = \lambda \cdot E[P_v]$).

Multicast cost

Table 4.5 shows the multicast communication cost ($\overline{C^m}$) according to the mean of vehicle subscription period ($E[P_v]$), the deviation of vehicle subscription period (D_v), and vehicle subscription rate (λ) as well. These results showed that the NKI to be delivered is not related with all parameters but only with λ, because of the two reason. First, the frequency of multicasting the NKI is related to only the number of updating GK ($O(\lambda)$).

Second, the MDKM reduces the increment of messages as the number of vehicles increases by using the proposed algorithms. The increments of $\overline{C^m}$ in Table 4.5 are not from *1-affect-all problem* but from the network overhead of multicast (Θ) and the frequency of updating GK. The result proves that the proposed scheme has no *1-affect-all problem* at a single key updating in terms of multicast cost as well, while conventional ones have. According to the results in the above table, we can observe that the proposed scheme requires little multicast communication cost. Even though λ increases, we can easily expect that the proposed scheme can successfully handle the cost without network burden in vehicular multicast service.

4.6.5 The Comparison of Key Management Cost

In this subsection, we compare the Key Management Cost of LKH with IR, Protocol A and B with BR, and the MDKM. Since the Protocol A and B proposed in Ji *et al* are designed by using the subscription information as well as MDKM [55], we compare the performance of the protocol A and B with that of MDKM. We set the coefficients based on the market prices of various devices such as flash memory and the price of data rate in cellular network; $\gamma_S^s = 0.0001$, $\gamma_S^v = 0.0005$, $\gamma_U^s = 0.0002$ and $\gamma_U^v = 0.001$, $\gamma_C^u = \gamma_C^m = 3$, (Assume that the unicast and multicast is delivered through same cellular network).

Fig. 4.7 shows the key management cost of four different key management schemes as the vehicle subscription rate (λ) varies. In the figure, LKH with IR shows the worst performance since LKH consists of a key tree not considering the vehicles dynamics. It notes that we designed efficient key management scheme by using subscription information.

MDKM greatly reduces the key management cost than Protocol A and B. This is because MDKM can minimize the usage of multicast delivery by decomposing the communication data into unicast data and multicast data, and by minimizing the multicast data, while Protocol A and B deliver most of key update messages via multicast. LKH with IR showed the worst performance. In addition, Protocol A and B, which utilize BR, do not guarantee the strict confidentiality in vehicular multicast services. As a result, we can deduce that the MDKM scheme is the most scalable and suitable method among the schemes for secure vehicular multicast services.

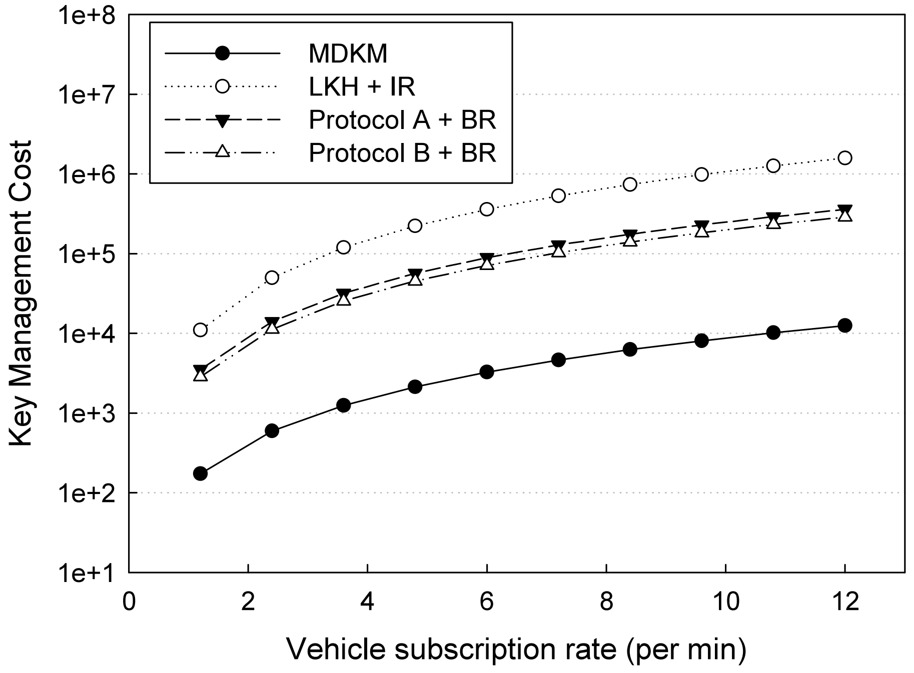

Figure 4.7: The Comparison of Key Management Cost for the LKH, Protocol A, B, and MDKM.

Chapter 5

Conclusion

In this dissertation, I proposed the efficient key management schemes in multicast communication.

First, I have presented a new key tree structure called the CSET, which is a subset of a level-homogeneous tree and a key tree management protocol which dynamically controls the key tree structure, considering the computation and storage efficiencies as well as the communication cost. By managing a key tree structure, the CSET greatly reduced the computation and storage costs. These are critical when designing a new group key management protocol in mobile communication environments. Though the CSET does not present the best communication cost, the CSET management protocol provided the method which minimizes the increment of the communication cost by controlling the key tree structure. To verify the efficiency of the protocol, I also analyzed and simulated the communication cost of a level-homogeneous tree structure in terms of the average number of update messages, as well as the storage and computation costs such as the energy consumption and the decryption delay. Based on these analysis and simulation, I have shown that the proposed protocol can greatly reduce computation and storage costs complexity at the expense of a small increase of the communication cost as a tradeoff. Even though the application scenarios in mobile communications are assumed, this approach is also applicable to wireless sensor networks in which sensor nodes have very limited resources.

Second, since previous researches have mainly focused on the scalability problem, a batch rekeying approach has been proposed and adopted in a group key management situation. However, a batch rekeying causes security damage by breaking the forward confidentiality. Therefore, I proposed a new batch rekeying scheme which optimizes the total cost per unit time, including the analytic model for the optimal batch rekeying interval. I modeled and analyzed the total cost into the communication cost and the security damage cost from the characteristics of the two costs. I conducted the simulations with respect to the various variables. The simulation results showed that the proposed scheme can provide the optimal batch rekeying interval to minimize the total cost per unit time.

Third, I have designed an efficient GKM scheme called MDKM which utilizes both unicast and multicast with KPA, KPAE, NKIA and NKDA to eliminate *1-affects-all problem* in vehicular multicast communication. In addition, I analyzed the performance of the proposed scheme in terms of communication, storage and computation costs in detail. Furthermore, as a design problem, I optimized the GKM cost which is the weighted sum of the three kinds of the costs. The simulation results showed that the proposed scheme has better efficiency than those of the conventional schemes.

Chapter 6

Appendix

6.1 Useful equations derived from the Binomial Theorem

The Binomial theorem is a formula giving the expansion of powers of sums. The equations that have been derived consist of combinations, most of which are quite complex to analyze. In order to simplify and analyze the Eq.(2.11), useful equations have been derived from the Binomial theorem and use the following results.

The simplest version of the Binomial theorem is as follows:

$$(a+b)^n = \sum_{k=0}^{n} C_k^n \cdot a^{n-k} \cdot b^k. \tag{6.1}$$

If $a = 1$ and $b = \chi$, Eq.(6.1) is given as

$$(1+\chi)^n = \sum_{k=0}^{n} C_k^n \cdot \chi^k. \tag{6.2}$$

If $\chi = -1$, Eq.(6.2) is following to

$$\sum_{k=0}^{n} C_k^n \cdot (-1)^k = (1+(-1))^n = 0. \tag{6.3}$$

Multiply χ to both sides of Eq.(6.2) then, the following equation is obtained:

$$\chi \cdot (1+\chi)^n = \sum_{k=0}^{n} C_k^n \cdot \chi^{k+1}. \tag{6.4}$$

Differentiating Eq.(6.4) with respect to χ, we can obtain

$$(1+\chi)^n + \chi \cdot n \cdot (1+\chi)^{n-1} = \sum_{k=0}^{n} C_k^n \cdot (k+1) \cdot \chi^k. \tag{6.5}$$

If $n \geq 2$ and $\chi = -1$ in Eq.(6.5), Eq.(6.5) is given as

$$\sum_{k=0}^{n} C_k^n \cdot (k+1) \cdot (-1)^k = (1+(-1))^n + (-1) \cdot n \cdot (1+(-1))^{n-1} = 0 \tag{6.6}$$

If $n = 1$ in Eq.(6.5), then Eq.(6.5) is

$$\sum_{k=0}^{1} C_k^1 \cdot (k+1) \cdot \chi^k = 1 + \chi + \chi = 1 + 2\chi. \tag{6.7}$$

If $\chi = -1$ in Eq.(6.5), then Eq.(6.7) is

$$\sum_{k=0}^{1} C_k^1 \cdot (k+1) \cdot (-1)^k = 1 + 2 \cdot (-1) = -1 \tag{6.8}$$

With Eq.(6.5), if $n = 0$, Eq. (6.5) is simplified to one.

Bibliography

[1] H.Harney, C.Muckenhirn and E. Harder, "Group Key Management Protocol(GKMP) Specification", RFC2093, 1997.

[2] H.Harney, C.Muckenhirn and E. Harder, "Group Key Management Protocol(GKMP) Architecture", RFC2094, 1997.

[3] D. M. Wallner, E. J. Harder and R. C. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, 1999

[4] X. S. Li, Y. R. Yang, M. G. Gouda and S. S. Lam, "Batch Rekeying for Secure Group Communications", in Proc. of 10th International World Wide Web Conference (WWW10), 2001.

[5] S. M. Ghanem and H. Abdel-Wahab, "A Secure Group Key Management Framework: Design and Rekey Issues", in Proc. of 8th IEEE International Symposium on Computers and Communication (ISCC), 2003.

[6] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication", in ACM Comput. Surv. 35, 3 (Sep. 2003), 309-329.

[7] D. A. McGrew and A. T. Sherman, "Key establishment in large dynamic groups using one-way function trees", IEEE Transaction on Software Engineering, Volume 29, Issue 5, May 2003 Page(s):444-458

[8] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas "Multicast security: a taxonomy and some efficient constructions", in Proc. IEEE International Conference on Computer Communication (INFOCOM), 1999.

[9] Y. Wang, J. Li, L. Tie and Q. Li, "An Efficient Key Management for Large Dynamic Groups", in Proc. IEEE 2th Annual Conference on Communication Networks and Services Research (CNSR), 2004.

[10] J. Lin, F. Lai and H. C. Lee, "Efficient Group Key Management Protocol with One-Way Key Deriviation," in Proc. IEEE Conference on Local Computer Networks (LCN), 2005.

[11] A. Hodjat and I. Verbauwhede, "The Energy Cost of Secrets in Ad Hoc Networks (Short Paper)," in Proc. IEEE Circuits and Systems Workshop (CAS), 2002.

[12] CrossBow Technology Inc. TelosB Mote Platform Datasheet. Available at: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.

[13] Texas Instruments, Inc.: MSP430x13x, MSP430x14x Mixed Signal Microcontroller. Datasheet, 2001.

[14] TinyOS, http://www.tinyos.net/

[15] W. H. Desmond Ng, M. Howarth, Z. Sun and H. Cruickshank, "Dynamic Balanced Key Tree Management for Secure Multicast Communications," in IEEE Trans. Comput., vol. 56, no. 5, pp. 590-605, May, 2007.

[16] A. Tucker, "Applied Combinatorics 3rd Edition", John Wiley & Sons, 1995, Chapter 8.

[17] J. Snoeyink and S. Suri and G. Varghese, "A Lower Bound for Multicast Key Distribution", in Proc. IEEE INFOCOM, 2001.

[18] W. T. Zhu, "Optimizing the Tree Structure in Secure Multicast Key Management", Communications Letters, IEEE, Volume 9, Issue 5, May 2005 Page(s):477 - 479.

[19] J. S. Lee, J. H. Son, Y. H. Park and S. W. Seo, "Optimal Level-homogeneous Tree Structure for Logical Key Hierarchy," In Proc. IEEE Conference on Communication System Software and Middleware workshop (COMSWARE), 2008.

[20] C. K. Wong, M. G. Gouda and S. S. Lam, "Secure group communications using key graphs", in Proc. ACM Special Interest Group on Data Communications (SIGCOMM), 1998.

[21] J. Pegueroles and F. Rico-Novella, "Balanced Batch LKH: New Proposal, Implementation and Performance Evaluation", in Proc. IEEE International Symposium on Computers and Communication (ISCC), 2003.

[22] J. Pegueroles, F. Rico-Novella, J. Hernandez-Serrano and M. Soriano, "Improved LKH for Batch Rekeying in Multicast Groups", in Proc. IEEE Information Technology: Research and Education (ITRE), 2003.

[23] S. Xu, Z. Yang, Y. Tan, W. Liu and S. Sesay, "An Efficient Batch Rekeying Scheme Based on One-Way Function Tree", in Proc. IEEE International Symposium on Communications and Information Technology (ISCIT), 2005.

[24] X. B. Zhang, S. S. Lam, Dong-Young Lee and Y. R. Yang, "Protocol Design for Scalable and Reliable Group Rekeying", in Proc. IEEE/ACM Transactions on Networking, 2003.

[25] R. Rivest, "The RC5 Encryption Algorithm", in Proc Leuven Workshop on Fast Software Encryption, 2005.

[26] R. Rivest, M. Robshaw, R. Sidney and Y. Yin, "The RC6^{TM} Block Cipher", 1998.

[27] J. Daemen and V. Rjimen, "AES Proposal: Rijndael", 1999.

[28] Y. Law, J. Doumen and P. hartel, "Survey and Benchmark of Block Cipers for Wireless Sensor Networks", ACM Transaction on Sensor Networks, Vol. 2(1), 2006.

[29] K. Almeroth and B. Quinn, "IP multicast applications: Challenges and solutions", IETF Draft, Nobember 1998, Filename: draft-quinn-multicast-apps-00.txt.

[30] R.Canetti, Juan Garay, Gene Itkis, Daniele Miccianancio, Moni Naor and Benny Pinkas, "Multicast security: a taxonomy and some efficient constructions", in IEEE INFOCOM, 1999, Page(s):708-716.

[31] J. H. Cho, I. R. Chen and M. Eltoweissy, "On optimal batch rekeying for secure group communications in wireless networks", Wireless Networks, Vol. 14, No. 6, pp. 915-927, 2008.

[32] Y. Ji and S. W. Seo, "Optimizing the Batch Mode of Group Rekeying: Lower Bound and New Protocols", in Proc. IEEE INFOCOM 10, 2010.

[33] D. H. Je, J. S. Lee, Y. S. Park and S. W. Seo, "Computation-and-storage-efficient key tree management protocol for secure multicast communications,", Computer Communications, ELSEVIER, Volume 32, Issue 2, pp.136-148, Feb. 2010.

[34] S. Boyd and L. Vandenberghe, "Convex Optimization", Cambridge University Press, 2006, Chapter 9.

[35] National Highway Traffic Safety Administration, CAMP Vehicle Safety Communications, Vehicle Safety Communications Project, Task 3 Final Report, Identify intelligent vehicle safety application enabled by DSRC," DOT HS 809 859, National Highway Traffic Administration, Washington, DC, March 2005.

[36] M. Nekovee, Sensor networks on the road: the promises and challenges of vehicular ad hoc networks and vehicular grids," in Proc. of the Workshop on Ubiquitous Computing and e-Research, Edinburgh, UK, May 2005.

[37] J. Blum, A. Eskandarian, and L. Hoffmman, Challenges of intervehicle ad hoc networks," *IEEE Trans. Intelligent Transportation Systems*, vol. 5, no. 4, pp. 347-351, Dec. 2004.

[38] M. Park, G. Gwon, S. Seo, and H. Jeong, RSU-Based Distributed Key Management (RDKM) for Secure Vehicular Multicast Communication". *IEEE Journal on Selected Areas in Communications* (JSAC), Vol 29, No. 3, March 2001.

[39] Y. Sun, W. Trappe, and K. J. R. Liu, A scalable multicast key management scheme for heterogeneous wireless networks", *IEEE/ACM transactions on Networking* vol. 12, no. 4, pp. 653-666, Aug. 2004

[40] IEEE Standard for Information technology. Telecommunications and information exchange between systems. Local and metropolitan area networks. Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2007.

[41] IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems, 2009.

[42] D. L. Mills, Internet time synchronization: The network time protocol", *IEEE Transaction on Communication*, vol. 3, no. 3, pp. 1482-1493, Oct. 1991.

[43] D. L. Mills, Improbed algorithm for synchromizing computer network clocks", *IEEE/ACM Transaction on Networking*, vol. 3, no. 3, pp. 245-254, Jun. 1995.

[44] D. L. Mills, J. Martin, Ed., J. Burbank, and W. Kasch, Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC5905, 2010.

[45] D. L. Mills, Precision synchronization of computer network clocks", *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 2, Apr. 1994.

[46] K. Almeroth and M. Ammar, Collecting and modeling the join/leave behavior of multicast group members in the mbone, in Proceeding 5th IEEE Int. Symp. High Performance Distributed Comput., pp. 209-216, 1996.

[47] K. Almeroth and M. Ammar, multicast group behavior in the Internets multicast backbone (MBone), IEEE Commun., vol. 35, pp. 224-229, Jun. 1999.

[48] T. Billhartz, J. Cain, E. Farrey-Goudreau, D. Fieg, and S. Batsell, Performace and Resource Cost Comparisons for the CBT and PIM Multicast Routing Protocols", *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, Apr. 1997

[49] H. Salama, D. Reeves, and Y. Viniotis, Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks", *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, Apr. 1997

[50] J. Chuang, and M. Sirbu. Pricing Multicast Communication: A Cost-Based Approach", *Telecommunication Systems*, pp. 281-297, 2001

[51] Crypto++ 5.6 Benchmarks. [Online]. Available: http://www.cryptopp.com/benchmarks.html

[52] S. Paul, Multicasting on the Internet and Its Applications, Boston, MA: Kluwer, 1998.

[53] R. Rivest, The MD5 Message-Digest Algorithm", RFC 1321, 1992.

[54] National Security Agency, US Secure Hash Algorithm (SHA)", RFC 3174, 2001.

[55] Y. Ji and S. Seo, Optimizing the batch mode of group rekeying: lower bound and new protocols", *in Proc. IEEE/ACM INFOCOM*, 2010.

국문 초록

엔터테인먼트, 통신, 기기 제어와 같은 다양한 어플리케이션에 대한 요구가 증가하고 네트워크 기술이 발달되면서, 멀티캐스트 통신은 한 번의 전송으로 그룹 사용자들에게 동시에 메시지를 전송할 수 있기 때문에 통신 부하를 줄이기 위한 유망한 솔루션들 중 하나가 되고 있다. 그러나 정보 노출이 주요한 관심사중 하나가 됨에 따라 키 관리 방법은 멀티캐스트 통신에서 상업용 어플리케이션을 성공적으로 보급하는데 필수적인 요소로 고려되고 있다. 유니캐스트와 다르게 멀티캐스트의 확장성은 데이터 기밀성을 제공하기 위한 비밀 키를 관리하는데 주요한 장애물이다. 따라서 본 논문에서는 멀티캐스트 사용자들이 비밀 키를 관리하는데 보안과 자원 요구사이의 관계에 대해서 초점을 맞추고, 멀티캐스트 통신에서 다양한 효율적인 키 관리 방법들에 대하여 제안한다: 연산 저장에 효율적인 키 트리 관리 프로토콜, 안전한 그룹 통신을 위한 최적 키 갱신 주기, 그리고 안전한 자동차용 멀티캐스트 통신을 위한 구성원의 역동성에 기반을 둔 키 관리 방법.

주요어: 안전한 멀티캐스트, 그룹 키 관리, 키 트리 관리, 일괄 키 갱신, 데이터 기밀성, 자동차용 멀티캐스트 서비스, 접근 제어

학 번: 2006-21289