Computers and Mathematics with Applications 65 (2013) 1300–1309

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Polynomial-based key management for secure intra-group and inter-group communication

Yanji Piao [a,*], JongUk Kim [a], Usman Tariq [b], Manpyo Hong [a]

[a] Department of Computer Science, Ajou University, Suwon, Republic of Korea

[b] Department of Computer Science, Imam Muhammad bin Saud University, Saudi Arabia

ARTICLE INFO

Keywords:
Group communication
Polynomial-based key management
Overhead reduction

ABSTRACT

Secure group communication has become an important issue in many applications. Both intra-group and inter-group multicast traffic must be protected by shared secret keys. In order to communicate securely in the same group and among different groups, we employed a polynomial P to achieve efficient intra-group key refreshment and generated a polynomial $H(x)$ to create an inter-group key. Proposed polynomial-based key management schemes have the following advantages: (1) Group members and the group controller can share the intra-group key without any encryption/decryption. (2) When the members of the group get changed, the group controller needs to update and distribute the renewed group keys. The proposed mechanism can reduce the number of re-keying messages. (3) The proposed mechanism lessens the storage overhead of group members and the group controller by adopting a polynomial-based key management scheme. (4) As compared with previous approaches, the group controller does not need to broadcast heavy messages which are necessary for creating an inter-group key. Hence, it introduces only a small amount of broadcast traffic to the group members. The analysis of the proposed mechanism is conducted to demonstrate the improvements.

1. Introduction

Group key management is a fundamental building block for secure group communication systems. The challenges are how to efficiently generate the secure key and re-keying without increasing the storage and communication overhead.

In this paper, we focus on the problem of key generation and update of a secure dynamic group. There are various applications in which multicast traffic exists both within the same group and among different groups, such as conferences, hospitals, and battlefields. For instance, in a network jointly formed by a group of doctors and patients, the doctors' group can divided into department A, B, and C. When one of the patients (denoted by v) sends his/her personal information to doctors in department A. Doctors in department B and C cannot decrypt the information of patient v because of the privacy of v, other patients cannot decrypt the information of patient v as well, and doctors in department A can communicate with each other securely. Another application is related to a company that has several teams such as audit team, development team, research team, personnel team and so on. If a member of research team wants to send an inspection report to the audit team securely, it must be encrypted with secret key. Only the audit team members will be able to recover the message from the sender. Enforcing security in these applications involves both intra-group and inter-group key management.

* Corresponding author.
E-mail addresses: piaoyj@ajou.ac.kr, piaoyanji@gmail.com (Y. Piao), kju@ajou.ac.kr (J. Kim), usman@usmantariq.org (U. Tariq), mphong@ajou.ac.kr (M. Hong).

0898-1221/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2012.02.008

There are more examples that can adopt intra-group and inter-group key management scheme in previous researches. One example comes from [1], the nodes are divided into soldiers, officers, and generals. Between the nodes in different groups routing requests and multicast traffic are needed, for instance, a soldier may send a message that can be read only by all of the generals. Another example comes from [2], three groups of soldiers coming from countries A, B, and C, when an event is observed by a soldier of country A, a description with different contents will be provided to different soldier groups. To support such requirements, secret keys must be deployed.

The straightforward solution to this problem is to encrypt messages with a shared secret key, so that entities who do not have the shared secret key cannot decode them. For the above application, when a member A in the research team wants to send an inspection report that should be read only by audit team members, it would be like a sender join operation, the audit team leader drops the previous group key which ciphered the past communication and generates a new shared secret key, and broadcasts it to the sender and all of the audit team members. When another member B also wants to send a message to the audit team, the audit team leader should regenerate a secret key which is shared between the member B and audit team members. This solution is simple, yet with a disadvantage: the leader will be overwhelmed by the communication overhead for distributing the secret keys to each of senders. The more senders want to send messages, the more the communication overhead increases.

We developed a key generation and update method for secure information sharing in the same group and among different groups. We employ a polynomial P to achieve efficient intra-group key refreshment and generate polynomials $H(x)$ from the intra-group key in order to communicate among different groups (notations P and $H(x)$ are illustrated in Section 3.1). Our proposed approach drastically reduces the amount of broadcast traffic in the inter-group communication. The additional number of re-keying messages and communication overhead caused by the proposed scheme has been properly justified. Through using the proposed mechanism, we can improve system efficiency and increase the network lifetime under the same traffic scenarios. We confirm that we adopt the inter-group polynomial-based mechanism proposed by Wang et al. [2–4].

The contributions of the proposed schemes are: (1) Sharing the intra-group key between the group controller and group members do not need to adopt any encryption/decryption mechanisms. (2) When membership changes happen, the keys are renewed immediately. The designed mechanism reduces the number of re-keying messages during group membership changes. (3) The adoption of the polynomial which is used for deriving an intra-group key can reduce the key storage overhead at the group members and the group controller. (4) After the intra-group key is derived, the members self-generate the polynomial functions which are necessary for creating an inter-group key. It helps to reduce the communication overhead at the group controller.

The rest of this paper is organized as follows: Section 2 illustrates related works. In Section 3, we introduce the intra-group and inter-group key management scheme in detail, and describe the key update operations when a node joins or leaves the group. In Section 4, we describe the evaluation of the proposed approach. Section 5 draws conclusions and Section 6 discusses future works.

2. Related works

As a result of the increased popularity of group oriented applications, there is a growing demand for the security services to achieve secure group communication [5–7]. Both intra- and inter-region group communication are an increasingly popular research area. There are a couple of survey papers [8–10] to address the group communication security issues.

In Group Key Management Protocol (GKMP) [11], the key server shares a secret key *KEKs* with each group member, the centralized controller will distribute a Group Key Packet (GKP) that contains a group traffic encryption key (GTEK) and a group key encryption key (GKEK) when a re-key is needed. The GTEK is used to encrypt the traffic and the GKEK is used to secure the distribution of a new GKP. The GKMP requires $O(n)$ re-key messages for each member leaving (or joining) the group, n being the number of the remaining group members. This one-to-one distribution mechanism does not scale to large networks with highly dynamic members.

Wong et al. [12,13] and Wallner et al. [14] proposed Logical Key Hierarchy (LKH). In this approach the members of a group have been organized into a hierarchy. Each member holds a copy of the secret key which is a leaf in the tree and all the keys from the leaf to the root. LKH allows reducing the number of re-key messages when membership changes. In the LKH protocol, when a new *KEK* is generated, it is encrypted with its two child *KEKs*. However, in One-Way Function Trees (OTF) [15], when a blinded key is changed in a node, it has to be encrypted only with the key of its sibling node. OTF lessens the key distribution overhead when a node leaves the group.

The centralized key management has a single point of failure problem, in order to avoid the 1-*affects*-n phenomenon, several mechanisms have been developed to divide members into subgroups. Each subgroup uses its own independent key. Mittra [16] proposed Iolus, the overall subgroups form a virtual multicast group. Each subgroup is managed by a Group Security Agent (GSA) which is responsible for key management inside the subgroup. Even though Iolus alleviates the 1-*affect*-n phenomenon, a disadvantage also exists: inter-subgroup traffic must be translated by the agents.

In several mechanisms the group members cooperate to establish a group key. Diffie and Hellman [17] proposed a well-known mechanism called the Diffie–Hellman protocol, but it involves exponential computation. The approach such as GDH [18] is one of the extensions of the Diffie–Hellman protocol [17], the first member calculates the first value and sends it to the next member, the last member raises all intermediate values to its secret value and multicasts the whole

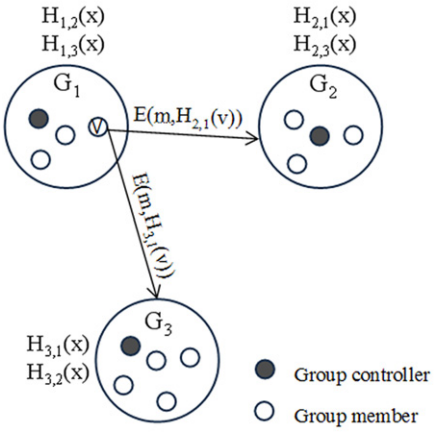

Fig. 1. A member v in G_1 sends message to all of the members in G_2 and G_3.

set to all of the group members. Thus, the setup time and the length of messages are linear. In the protocol proposed by Ingemarson et al. [19], each member is organized into a virtual ring and the members perform n exponentiations.

In order to make up for the weak points of intra-group key management, we adopt the polynomial-based mechanism to derive the intra-group key. The group controller generates a polynomial which contains intra-group key, and broadcasts it to the members; then the members are easily able to derive the intra-group key from the polynomial. It will be described in Section 3.2 in detail.

Inter-group key management is an important issue in the secure group communication as well. Both intra- and inter-group key management are considered to secure the group communication. Huang et al. [20] proposed an inter-group roaming protocol for inter-group key management to improve the survivability when group controllers fail or members from one group roam into another group. But it is not suitable for our applications.

To ensure secure inter-group communication Wang et al. [2–4] have proposed a polynomial-based scheme. A polynomial-based scheme was first used to implement threshold secret sharing [21]. Staddon et al. [22] and Liu et al. [23] proposed a self-healing group key distribution mechanism. In the paper [2–4], authors adopt polynomials to support the distribution of personal key shares. They use t-degree polynomial $H(x)$ to determine the personal key shares and protect inter-group multicast traffic. In Fig. 1, $H_{2,1}(v)$ means personal key share to encrypt multicast traffic from v in the group one (G_1) to the members of group two (G_2). $H_{2,1}(x)$ means a polynomial to determine the keys for decrypting the multicast traffic from a node in G_1 to the members of G_2. A node v in G_1 will get its personal key shares $H_{2,1}(v)$ from the group controller, the group controller of G1 will request $H_{2,1}(v)$ from the group controller of G2, and node v encrypts the message using $H_{2,1}(v)$ and sends it to the members in G_2, in this time, the nodes in G_2 already get $H_{2,1}(x)$ from the group controller and they know that the message comes from node v, so, the nodes in G_2 can calculate $H_{2,1}(v)$ and decrypt the message from node v. For example, $H_{2,1}(x) = 5x + 8$, all the members in G_2 have $H_{2,1}(x)$, if the value of node v is 5, the inter-group key will be calculated as $H_{2,1}(v) = 5 * 5 + 8 = 33$. So, in this case only the sender v and members in G_2 will be able to read the information which is encrypted with 33, because other nodes cannot calculate the polynomial. Here, the polynomials $H(x)$ is generated by the group controller, the group controllers consume energy not only when they generate the polynomials, but also when they send it to the group members. In Section 3.3, we develop an efficient generation method of $H(x)$ by making up for the weakness of previous works.

3. Polynomial-based key management scheme

We designed polynomial-based mechanisms to protect both intra-group and inter-group multicast traffic. A polynomial-based scheme was first adopted in the paper [21] proposed by Shamir. The possible advantages of adopting polynomials in our scheme will be present at the beginning of Sections 3.2 and 3.3. In the proposed group key management scheme, we applied two kinds of polynomials, the first polynomial (denoted by P) is used to derive the intra-group key, and the second polynomial (denoted by $H(x)$) is used to create the inter-group key. In this section, firstly, we will describe the method of intra-group key generation using polynomial P, secondly, we will present the generation method of polynomial $H(x)$ which is used for creating the inter-group key.

Before introducing the polynomial-based intra-group and inter-group key management scheme, we explain the process of the intra-group and inter-group key management scheme. The purpose of the process is that only the sender and the members in the target group can get access the encrypted messages. As seen in Fig. 2, the ultimate goal is that the member v in the group G_j sends a message to the members in G_t securely. To achieve the goal the following steps are presented. Firstly, all the members in the target group G_t have to share the intra-group key GK_t to maintain the secure intra-group communication. The methodology of intra-group key generation will be described in the following Section 3.2. After sharing

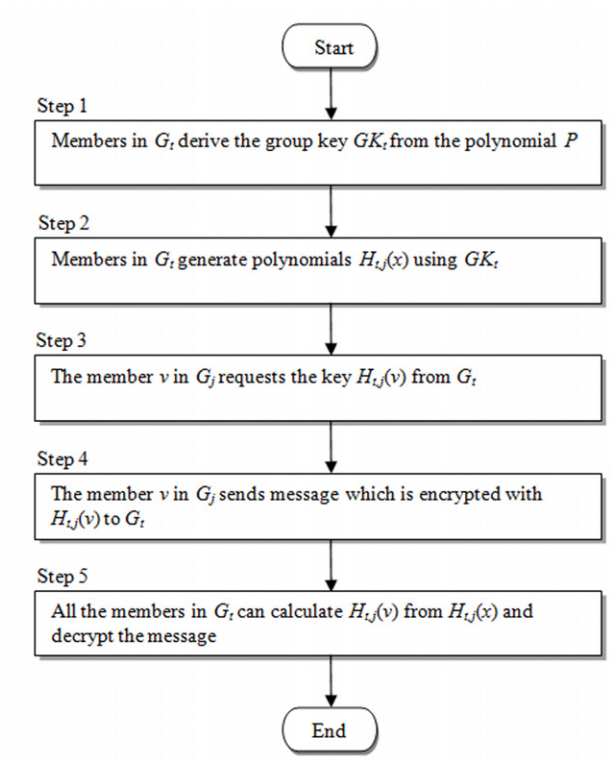

Fig. 2. The process of the intra-group and inter-group key management scheme.

Table 1
Notations and usage.

Notations	Usage
G_k	The k-th group
GK_k	Intra-group key for members of Gk
KEK_i	Secret key which is shared between group controller and a member i, it is used to create polynomial P
P	Polynomial function which is used for deriving intra-group key GK
$H(x)$	t-degree polynomial which is used for determining inter-group key
$H_{t,j}(x)$	Polynomial to determine the keys for decrypting the multicast traffic from a node in G_j to the members of G_t
$H_{t,j}(v)$	Personal key share to encrypt multicast traffic from v in G_j to the members in G_t

the GK_t securely, the members in G_t generate a polynomial $H_{t,j}(x)$ using GK_t, the method in step 2 is the main approach of inter-group key management, and more of this will be presented in Section 3.3. $H_{t,j}(x)$ is a polynomial to determine the keys for decrypting the multicast traffic from a node in G_j to G_t. When the node v in G_j wants to send a message to G_t, it will request the inter-group key $H_{t,j}(v)$ from G_t, then the node v sends encrypted messages to G_t, here, all the members in G_t can decrypt the messages using $H_{t,j}(x)$ since they know the message comes from the node v, because the members of G_t already generated polynomial $H_{t,j}(x)$ in the past phase and the sender v already sent his/her values to G_t. The details of step 3, step 4, and step 5 are presented in the paper [2–4].

3.1. Notation

We assume that i is the node ID, where $i \in \{1 \ldots n\}$ and n is the number of nodes in a group. All the node IDs in the networks are different and the nodes in the networks are divided into d different groups, where $k \in \{1 \ldots d\}$ and k is the group index. The intra-group key is represented by GK. Table 1 shows the notations and their usage.

3.2. Group key sharing for secure intra-group communication

In this section, we describe the intra-group key management scheme which adopts a polynomial P. This scheme uses the inherent feature of the broadcast mechanism, since the group controller generates the polynomial P and broadcasts it to the members in the group. The obvious broadcast encryption scheme is: the group controller binds every member's own key and broadcasts it to the members, and then the members in the group only select the key which they need from the bunch

of keys and use it. It seems such a waste to select the key from the broadcast messages. In our intra-group key management scheme, the group controller also broadcasts P, but the main advantage of using polynomial P is that the group members do not need to choose the necessary keys from the broadcast message.

Below we would like to discuss the details of the intra-group key management scheme which adopted the polynomial-based method. The goal is that every member in G_k shares the intra-group key GK_k securely and efficiently. First, Key Encryption Key (KEK_i) which is shared between the group controller and the member i is pre-distributed. Secondly, the group controller generates a polynomial P according to KEK_i, and then broadcasts expanded polynomial to the members. Finally, each group member receives the polynomial and derives an intra-group key GK_k.

1. There are n members in the group G_k, and i is the node ID. We assume that the group controller gives every member i the keys KEK_i using a secure channel corresponding to the group it belongs to.
2. The group controller generates a polynomial P which uses all secret keys. The polynomial is:
$$P = (x - KEK_1)(x - KEK_2) \ldots (x - KEK_n) + GK_k, \quad i = 1, \ldots, n \tag{1}$$
where GK_k is the group key of G_k generated by the group controller, the group controller broadcasts P to the members.
3. When group members receive P, the group key GK_k is computed as:
$$(x - KEK_1)(x - KEK_2) \ldots (x - KEK_n) + GK_k, \quad x = KEK_i. \tag{2}$$

Hence, group members can derive the group key GK_k efficiently and securely. The reason of 'efficiently' and 'securely' will be analyzed in Section 4.1.

3.3. Group key sharing for secure inter-group communication

Our purpose is that to generate an inter-group key which is shared between the sender and all of the target group members, in other words, only the sender and members in the target group can read the messages. The traditional method is that the sender shares the key with target group members through the group controller; put simply, the sender sends the key request message to the group controller of his/her group, then this group controller requests the target group controller for the key, and finally the target group controller sends back the shared key in a key reply message. The method requires every member in the target group to store a huge number of keys.

To solve this problem Wang et al. [2–4] proposed a mechanism which is suitable for this environment. In [2–4], the polynomial $H(x)$ is adopted to determine the keys to protect inter-group communication. Instead of sharing the secret key with each of the senders, target group members only need to store $H(x)$ which determines the keys for decrypting the message from the sender. Consequently, it can reduce the storage overhead of the group members. But, it involves a disadvantage: the polynomial $H(x)$ is generated by the group controller, and the group controller broadcasts it to the members in the group. Therefore, a majority of the broadcast traffic comes from the distribution of polynomials.

In order to fill the disadvantage, we proposed the self-generation method of the polynomials by using the intra-group key. When nodes are divided into some groups, the intra-group key must be shared first, therefore the shared intra-group key can make it convenient to generate a polynomial $H(x)$ by the members. As compared with previous research (illustrated in Section 2), the proposed scheme can reduce the number of distributions since the generation of the polynomials does not rely on a group controller. Several kinds of methods should be adopted to generate $H(x)$ using the intra-group key. In this paper we just introduce one of them, which uses a pseudo-random number generator (PRNG) to generate coefficient of a polynomial $H(x)$. In this section, we will discuss the polynomial $H(x)$ generation method which is used for creating inter-group keys shared between the sender and the members in the target group.

Fig. 3 shows the method of generating the polynomials using PRNG. We assume there are three groups G_1, G_2, and G_3 in the networks. As illustrated in the paper [2–4], members in G_1 need $H_{1,2}(x)$ and $H_{1,3}(x)$ in order to decrypt messages from G_2 and G_3, members in G_2 need $H_{2,1}(x)$ and $H_{2,3}(x)$ in order to decrypt messages from G_1 and G_3, members in G_3 need $H_{3,1}(x)$ and $H_{3,2}(x)$ in order to decrypt messages from G_1 and G_2. As seen in Fig. 3, each intra-group key is used to create a coefficient of polynomial $H(x)$, the value of the first byte of GK_1 98 is used to generate the coefficient of $H_{1,2}(x)$, and the value of the second byte 207 is used to generate the coefficient of $H_{1,3}(x)$. All the members in G_1 using GK_1 to generate $H_{1,2}(x)$ and $H_{1,3}(x)$ by themselves. The members in the same group have the same PRNG. The generation of the group key GK_i is already mentioned in Section 3.2. Let us explain Fig. 3 in more detail, the first byte of GK_1 98 is the first seed of PRNG. The first output of PRNG 23 is the first coefficient of the polynomial $H_{1,2}(x)$, it is also used as the second seed of PRNG, and the second output 118 is the second coefficient of the polynomial $H_{1,2}(x)$. One more explanation here, the first byte of GK_1 207 is the first seed of PRNG. The first output of PRNG 113 is the first coefficient of the polynomial $H_{1,3}(x)$, it is also used as the second seed of PRNG, and the second output 88 is the second coefficient of the polynomial $H_{1,3}(x)$. Each coefficient of polynomials $H_{2,1}(x)$, $H_{2,3}(x)$, $H_{3,1}(x)$, and $H_{3,2}(x)$ is determined in this way. When using one byte of GK to generate the coefficient of $H(x)$, the probability of the same $H(x)$ is $1/28$. We can use two bytes or three bytes as the seed of PRNG, in this case the probability of same $H(x)$ is $1/216$ or $1/224$. If and when the collision attack occurs, we assume that all the IDs of the members in the networks are different. Hence, the value of $H(x)$ will be different.

To make a long story short, all the members in the group derive GK_i from the polynomial P as described in Section 3.2, and the coefficient of the polynomial $H(x)$ is generated from GK_i by each member in the group; finally, they share the

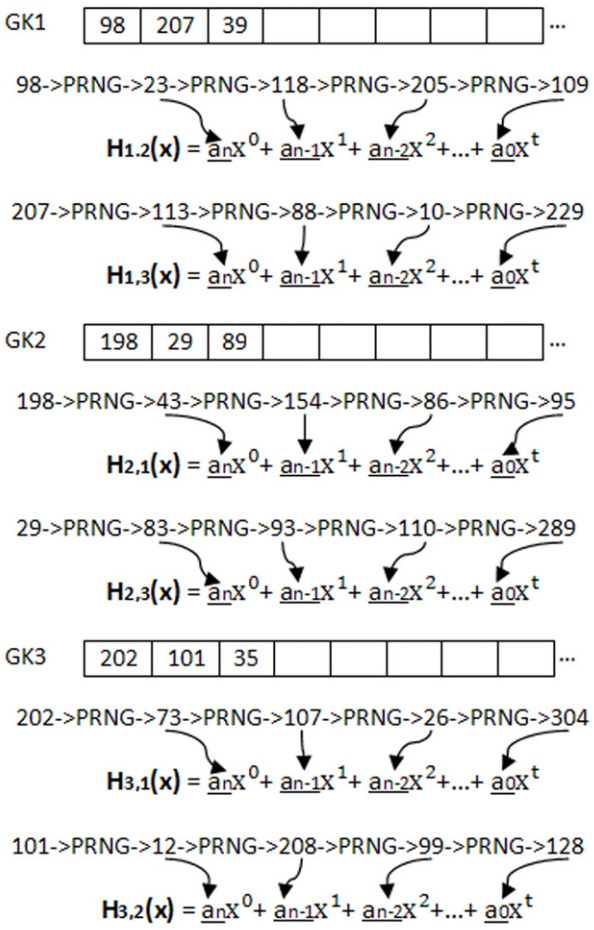

Fig. 3. An example of generating polynomials using an intra-group key.

inter-group key using $H(x)$ which presented in the paper [2–4]. The inter-group key management mechanism reduces the cost of group controllers, since the group controllers do not need to generate a polynomial $H(x)$ and broadcast it to the members.

3.4. Re-keying operations

Any secure group key management scheme should have four main security properties, that is, perfect forward secrecy, group forward secrecy, group backward secrecy, collusion resistance [24]. When group membership change happens, the corresponding intra-group and inter-group keys must be renewed to enforce forward and backward secrecy [25]. Forward secrecy assures that when a node leaves a group, it cannot access the traffic after leaving using the old keys. Backward secrecy guarantees that when a node joins a group, it cannot discover the old keys based on its current knowledge. In this section we present the key update operations. The methodology (discussed in Section 3.2) has been used for intra-group re-keying to generate polynomial P. The proposed scheme also facilitates members to generate $H(x)$ by using an intra-group key mechanism.

3.4.1. Re-keying for member join

We assume that a node w wants to join the group G_k, the current members of G_k have been using GK_k to encrypt the multicast traffic within the group. To prevent node w from getting access to the previous messages, the group key GK_k must be replaced by GK'_k. The re-keying messages are described as follows:

1. The node w shares secret key KEK_w with the group controller.
2. In order to maintain backward secrecy the group controller generates a new polynomial P:

$$P = (x - KEK_1)(x - KEK_2)\ldots(x - KEK_w)(x - KEK_n) + GK'_k, \quad i = 1, \ldots, n \tag{3}$$

where GK'_k is the new intra-group key. The group controller broadcasts the new polynomial to the members.
3. After obtaining the new polynomial P, all of the group members including w can derive the new group key GK'_k using their $KEK_i (x = KEK_i)$.

3.4.2. Re-keying for a member leaving

If a member leaves the group, the re-keying must be performed to ensure forward secrecy. We assume that when the node i is expelled from G_k, the group key GK_k must be replaced by the new secret GK'_k. The group controller regenerates a new polynomial P:

$$P = (x - KEK_1)(x - KEK_2) \ldots (x - KEK_{i-1})(x - KEK_{i+1}) \ldots (x - KEK_n) + GK'_k, \quad i = 1, \ldots, n \tag{4}$$

where GK'_k is the new group key generated by the group controller. Other nodes in G_k can derive GK'_k but the node i cannot derive GK'_k.

After the join/leave operation, in order to maintain backward and forward secrecy of inter-group communication the polynomial $H(x)$ must be replaced by new polynomials using GK'_k as described in Section 3.3.

4. Evaluating proposed approach

In this section, we investigate efficiency and safety, the number of re-keying messages, storage, and the communication overhead of the proposed intra-group and inter-group key management scheme. We also illustrate the overhead when every group uses GKMP [11], LKH [12–14], and OTF [15].

4.1. Efficiency and safety

We have investigated the following security and efficiency problems of the proposed mechanism.

4.1.1. Efficiency

In the previous work, the group controller should encrypt intra-group key with secret key and send it to the members; however, in our mechanism the group controller does not need to encrypt the polynomial P, only send the expanded polynomial P. Normally computing the polynomial is easier than performing encryption and decryption. Hosangadi et al. [26] proposed a technique to optimize the energy consumption for computing polynomial expressions. From the point of view of a member, it is easier to derive GK from the polynomial than decrypting GK, hence it is more efficient than previous work.

4.1.2. Attack models

In general, attacks affecting secure key management are usually overhearing and node capture [27], called *passive* adversaries and *active* adversaries. Passive adversaries obtain some information through overhearing without any physical access, and active adversaries secure information that is leaked through physical access to the node. The proposed mechanism is secure against overhearing and node capture, and we will explain the reason below.

In order to derive GK, the group controller sends a multiplied polynomial without any encryption; nevertheless, it is not easy to guess the intra-group key GK from the polynomial, since it is hard to do polynomial factorization because there is actually an $O(n \log n)$ solution to the polynomial expansion problem [28,29] (n being the degree of the polynomial) and the problem for polynomial factorization is NP-hard [30]. For instance, $P = x^3 - 15x^2 + 74x - 117$ is not easy to factor as $P = (x-5)(x-4)(x-6)+3$, for the reason, it is difficult to guess 3. Even if the multiplied polynomial P is sent without any encryption, GK will not be leaked out. Thus, it is difficult to generate the coefficient of the polynomial $H(x)$ by an attacker because of the safety of GK.

4.1.3. Creating group controllers

The group controllers perform a vital role in the discussed scheme: they are responsible for creating and spreading secret keys and polynomials P. Moreover, in addition to the ability of creating protected secrets, other features such as computational capability and reliability of the nodes should also be measured during the creation process. If a pre-circulated mechanism is available in the infrastructure, the controller creation process can use the benefit of those particular network nodes. In support of the argument consider the example, in the armed forces operational activity explained in the Introduction (Section 1) the representative with the top level in the soldier groups of each country can act as the controllers. In autonomic systems, such as mesh networks a complex controller creation and selection process should be implemented. As a starting point we can take on a variant of the protected controller selection scheme for mesh networks. The node that obtains the highest number of approvals will turn into the controller.

4.2. Re-keying overhead

We now describe the re-keying overhead of the proposed scheme. In Table 2 we compare the protocols against the following relevant criteria: (1) The re-keying overhead for members joining: the number of re-keying messages sent by

Table 2
Comparison of the number of re-keying messages after join/leave.

Protocol	The number of re-keying messages		
	Join		Leave
	Multicast	Unicast	
GKMP [11]	2	2	$2n$
LKH [12–14]	$\text{Log}_2(n) - 1$	$\text{Log}_2(n) + 1$	$2\text{Log}_2(n)$
OTF [15]	$\text{Log}_2(n) + 1$	$\text{Log}_2(n) + 1$	$\text{Log}_2(n) + 1$
Our idea	1	1	1

n: number of group members.

Table 3
Comparison of the storage overhead.

Protocol	Group controller	Member
GKMP [11]	$n + 2$	3
LKH [12–14]	$2n - 1$	$\text{Log}_2(n) + 1$
OTF [15]	$2n - 1$	$\text{Log}_2(n) + 1$
Our idea	$n + 1$	2

the group controller/key server to redistribute the intra-group key after a join. (2) The re-keying overhead for members leaving: the number of re-keying messages sent by the group controller/key server to redistribute the intra-group key after a leave.

Studying the results in Table 2, GKMP requires $O(n)$ re-keying messages after a member has left the group, with n being the number of the remaining group members. Therefore, this solution does not scale to large groups with highly dynamic members. LKH and OTF using a hierarchical tree of KEKs allow us to reduce the number of re-key messages when a member leaves the group [31].

In the proposed polynomial-based scheme for intra-groups, when a member joins the group, the group controller needs to unicast a secret key to the new member using secure channel and multicast a polynomial P to the members in the group including the newly joined member. When a member leaves the group, the group controller needs to change the polynomial P and multicast it to the members in the group except the leaving member. This approach achieves excellent results for re-keying messages. However, these results are achieved by increasing the computation overhead due to the polynomial P calculations. In [32], the authors investigate the computation cost to generate a 64-bit key on MICA2 sensors by evaluating a polynomial with an optimized method. The results show that when $t \leq 80$, t is the degree of the polynomial, the computation overhead is comparable to the calculation of a 64-bit MAC code on a 64-bit message using SkipJack. SkipJack is an algorithm for encryption [33].

4.3. Storage overhead

Suppose that we are interested in limiting the number of keys that a member and a group controller must store. Table 3 shows the comparison of the storage overhead. We compare the protocols against the following criteria: (1) Storage at a group controller: the number of keys that should be maintained by a group controller; (2) Storage at a member: the number of keys that should be maintained by a group member.

The values in Table 3 show the analysis results. In the proposed approach, group controller only needs to store one polynomial P and n KEK_i which shared between group controller and member i, with n being the number of group members and i being the group members' ID. The members in the group need to store one polynomial P and one KEK_i, as compare with previous schemes, it leads to a saving in the memory requirements described in the mechanism. So far, the good solutions for intra-group key management appear to be those using a polynomial P.

4.4. Communication overhead

Because a central mechanism is implicated in generating, dispensing, re-keying, and providing access control for every group, it is regularly accessed. The available communication resources to this method often turn out to be a bottleneck for groups. Consequently we need a larger pipe to handle such a burden. The protocol proposes to hand over a large number of the key formation, distribution, re-key and access control assignment to the hosts that require the protected communication. We no longer require an intermediation which demands consolation before any key management procedure. Consequently, the required communication to manage the key pool has changed to the cluster them.

In GKMP protocol [11], at the initial phase of the intra-group key generation, in order to share the group key, the group controller needs to unicast the group key which is encrypted with the Key Encryption Key (KEK). However, our scheme only needs to broadcast the polynomial P, and each member can derive the group key by themselves. Accordingly, in the initialization of intra-group key sharing, the number of unicast messages is reduced.

In the inter-group key management scheme, we assume that there are d groups in the network and n members in the group. In the previous approach [2–4], the group controller needs to send $n * (d - 1)$ t-degree polynomials $H(x)$ to the members. Compared to the overhead of generating the polynomials $H(x)$ by the group controller [2–4], all the members are contributors in the creation of the polynomials $H(x)$. Packet transmission is more power consuming than computation [34,35] and hence it will be much more efficient to create the polynomials by member themselves. The proposed inter-group key management scheme achieves an excellent result for distribution at the group controller.

5. Conclusion

Secure group communication has become an important component of many applications. Both intra-group and inter-group multicast traffic must be protected by shared secret keys. In order to communicate securely in the same group and among different groups, in this paper, we focus on the group key generation for intra-group communication and also focus on the polynomial generation which is used for creating secure inter-group key. In the proposed scheme, group members and group controllers can share the intra-group key without any encryption/decryption. Proposed mechanisms are also designed to reduce the number of re-keying messages during group changes using the polynomial P. Compared to previous approaches, the proposed scheme drastically reduces the amount of broadcast traffic in the inter-group communication since the polynomial is generated by members rather than attributed by the group controller. We also study the efficiency and safety of the proposed mechanism.

6. Future works

Normally, inter-group communication means that one group communicates with another group. However, what we call inter-group communication in this paper is that a sender communicates with all of the members in another group. We plan to integrate the polynomial-based mechanism into the group-to-group communication environment. We also plan to investigate the consideration of the group session. If a session runs for too long, the group key is renewed periodically. The proposed scheme refers to the session group key scheme such as [36]. We are going to study step 3 which is mentioned in Fig. 2 in depth; there is no authentication method in step 3 now. Because, inter-group impersonation is still a threat, for example, a malicious node x can impersonate node v by requesting the key $H_{t,j}(v)$ from G_t. A well-defined authentication method can prevent such attacks. The result will lead to a more secure key management protocol for intra-group and inter-group communication in multi-purpose environments.

Acknowledgment

This research is supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0011289).

References

[1] S. Yi, P. Naldurg, R. Kravets, Security-aware Ad Hoc routing for wireless networks, in: Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2001, pp. 299–302.
[2] W. Wang, T. Stransky, Stateless Key Distribution for Secure intra-group and inter-group multicast in mobile wireless network, Computer Networks 51 (15) (2007) 4303–4321.
[3] W. Wang, B. Bhargava, Key Distribution and Update for Secure inter-group multicast communication, in: Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks, in conjunction with ACM CCS, 2005, pp. 43–52.
[4] W. Wang, Y. Wang, Secure group—based information sharing in mobile Ad Hoc networks, in: IEEE International Conference on Communications, 2008.
[5] B. Xie, A. Kumar, D. Zhao, R. Reddy, B. He, On secure communication in integrated heterogeneous wireless networks, International Journal of Information Technology, Communications and Convergence (2010) 4–23.
[6] S. Wang, Y. Tsai, C. Shen, P. Chen, Hierarchical key derivation scheme for group-oriented communication systems, International Journal of Information Technology, Communications and Convergence (2010) 66–76.
[7] P. Sarkar, A. Saha, Security enhanced communication in wireless sensor networks using Reed–Muller codes and partially balanced incomplete block designs, Future Technology Research Association International, Journal of Convergence (2011) 23–30.
[8] B. Jiang, X. Hu, A survey of group key management, in: International Conference on Computer Science and Software Engineering, 2008, pp. 994–1002.
[9] S. Rafaeli, D. Hutchison, A survey of key management for secure group communication, ACM Computing Surveys 35 (3) (2003) 309–329.
[10] S. Devi, A.S. Danamani, A survey on multicast rekeying for secure group communication, International Journal of Computer Technology and Applications (2011) 385–391.
[11] H. Harney, C. Muckenhirn, Group Key Management Protocol (GKMP) Architecture, RFC 2094, 1999.
[12] C.K. Wong, M. Gouda, S.S. Lam, Secure group communications using key graphs, in: Proceedings of the ACM SIGCOMM'98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication 1998, pp. 68–79.
[13] C.K. Wong, M. Gouda, S.S. Lam, Secure group communications using key graphs, IEEE/ACM Transactions on Networking 8 (1) (2000) 16–30.
[14] D. Wallner, E. Harder, R. Agee, Key management for multicast: issues and architecture, RFC 2627, 1999.
[15] D. McGrew, A. Sherman, Key establishment in large dynamic groups using one-way function trees, Technical Report TR-0755, 1998.
[16] S. Mittra, Iolus: A Framework for Scalable Secure Multicasting, ACM SIGCOMM, 1997.
[17] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory (1976) 644–654.
[18] M. Steiner, G. Tsudik, M. Waidner, Diffie–Hellman key distribution extended to group communication, in: 3rd ACM Conference on Computer and Communications Security, March 1996, pp. 31–37.
[19] I. Ingemarson, D. Tang, C. Wong, A conference key distribution system, IEEE Transactions on Information Theory (1982) 714–720.
[20] D. Huang, D. Medhi, A secure group key management scheme for hierarchical mobile Ad Hoc networks, Ad Hoc Networks 6 (2008) 560–577.

[21] A. Shamir, How to share a secret, Communications of the ACM (1979) 612–613.
[22] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, D. Dean, Self-healing key distribution with revocation, in: Proceedings of IEEE Symposium on Security and Privacy, 2002.
[23] D. Liu, P. Ning, K. Sun, Efficient self-healing group key distribution with revocation capability, in: Proceedings of ACM conference on Computer and Communications Security, 2003, pp. 231–240.
[24] N.N. Karuturi, R. Gopalakrishnan, R. Sriniasan, P.R. Chandrasekaran, Foundations of group key management—framework, security model and a generic construction, IACR, 2008.
[25] A. Perrig, D. Song, J. Tygar, Elk, a new protocol for efficient large-group key distribution, in: Proceedings of IEEE Symposium on Security and Privacy, 2001.
[26] A. Hosangadi, F. Fallah, R. Kastner, Energy efficient hardware synthesis of polynomial expressions, in: 18th International Conference on VLSI Design, 2005, pp. 653–658.
[27] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, SPINS: security protocols for sensor networks, in: Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking, 2001, pp. 189–199.
[28] Michael Sipser, Introduction to the Theory of Computation, 1996.
[29] D.S. Roche, Space- and time-efficient polynomial multiplication, ACM ISSAC'09, July 2009, pp. 28–31.
[30] S. Gao, M.V. Hoeij, E. Kaltofen, V. Shoup, The computational complexity of polynomial factorization, American Institute of Mathematics, 2006.
[31] Y. Challal, A. Bouabdallah, H. Seba, A taxonomy of group key management protocols: issues and solutions, in: Proceedings of World Academy of Science, Engineering and Technology, 2005.
[32] D. Liu, P. Ning, R. Li, Establishing pairwise keys in distributed sensor networks, ACM Transactions on Information and System Security (2005) 41–47.
[33] Use of the KEA and SKIPJACK Algorithms in CMS, RFC 2876, 2000.
[34] L. Ferrigno, S. Marano, V. Paciello, A. Pietrosanto, Balancing computational and transmission power consumption in wireless image sensor networks, in: Proceedings of IEEE VECIMS, 2005.
[35] C. Huang, R. Cheng, S. Chen, C. Li, Enhancing network availability by tolerance control in multi-sink wireless sensor networks, Future Technology Research Association International, Journal of Convergence (2010) 15–22.
[36] E. Bresson, M. Manulis, Malicious participants in group key exchange: key control and contributiveness in the shadow of trust, in: The 4th International conference on Autonomic and Trusted Computing, 4610, 2007, pp. 395–409.