Copyright ⓒ 2014 KSII

Group Key Management based on (2, 2) Secret Sharing

Lih-Chyau Wuu[1], Chi-Hsiang Hung[2] and Wen-Chung Kuo[3]
[1] Graduate School of Computer Science and Information Engineering,
National Yunlin Unviersity of Science and Technology, Yunlin, 640 - Taiwan
[e-mail: wuulc@yuntech.edu.tw]
[2] Graduate School of Engineering Science and Technology-Doctoral Program,
National Yunlin Unviersity of Science and Technology,Yunlin, 640 - Taiwan
[e-mail: g9510802@yuntech.edu.tw]
[3] Graduate School of Computer Science and Information Engineering,
National Yunlin Unviersity of Science and Technology, Yunlin, 640 - Taiwan
[e-mail: simonkuo@yuntech.edu.tw]

*Corresponding author: Lih-Chyau Wuu

Received October 28, 2013; revised January 4, 2014; accepted February 9, 2014; published February 9, 2014

Abstract

In Internet, IP multicast has been used successfully to provide an efficient, best-effort delivery service for group communication applications. However, applications such as multiparty private conference, distribution of stock market information, pay per view and other subscriber services may require secure multicast to protect integrity and confidentiality of the group traffic, and validate message authenticity. Providing secure multicast for group communication is problematic without a robust group key management. In this paper, we propose a group key management scheme based on the secret sharing technology to require each member by itself to generate the group key when receiving a rekeying message multicast by the group key distributor. The proposed scheme enforces mutual authentication between a member and the group key distributor while executing the rekeying process, and provides forward secrecy and backward secrecy properties, and resists replay attack, impersonating attack, group key disclosing attack and malicious insider attack.

Keywords: Group key management, dynamic group, mutual authentication, secret sharing

http://dx.doi.org/10.3837/tiis.2014.03.025

1. Introduction

The IP multicast is an efficient protocol to delivering group traffic in a group-based application by requiring a sender to transmit data only once to many receivers along a multicast distribution tree. IP multicast can use the network bandwidth efficiently that it is widely deployed in group-based applications such as televised company meetings, commercial stock exchanges, pay-per-view stream video [7], chat-room and so on. Some group-based applications require multicast to embed certain security mechanisms to protect the integrity of group traffic from modifications, guard for confidentiality of communication from electronic eavesdrop, and verify message originator. To provide the above security-enhanced services for IP multicast, it is necessary to have a robust group key generation and distribution scheme.

A simple method to generate a group key is to rely on a specific server called as group key distributor (GKD) [19]. Each group member shares a pre-shared key with the GKD. After the GKD generates a new group key, it encrypts the group key by each member's pre-shared key and sends out the encrypted group key to each member separately. In this way, the rekeying message complexity is O(t) for member joining/leaving a group, where t is the group size.

Many schemes [1, 3-9, 11-13, 16-18] address group key management to reduce the number of the rekeying messages. The schemes [1, 5-9, 11-12, 16, 18] establish a key tree for a group. The tree root stores the group key, the internal nodes of the tree store the auxiliary keys used to renew the group key, and each leaf of the tree stored the pre-shared key known by both of the GKD and a group member only. For t members in a group, the GKD must maintain a group key, O(t) auxiliary keys and O(t) pre-shared keys. The number of auxiliary keys maintained by a member is dependent on the level of this member (leaf node) on the tree. For a balanced tree, the GKD must send O(log t) rekeying messages during the rekeying process. If the key tree is a skewed one, the rekeying messages will be O(t). It is obvious that a balanced tree has better rekeying performance than a skewed one. However, in a highly dynamic group, the overhead to maintain a tree being balanced is heavy. Furthermore, such schemes require that each member must keep O(log t) auxiliary keys.

Harn and Lin [4] propose an authenticated group key transfer protocol (AGKTP) based on secret sharing scheme that the GKD broadcasts group key information to all group members and only the authorized group members can recover the group key. No auxiliary keys are needed and only two broadcast messages are sent out by the GKD, but the AGKTP protocol requires each member to send a random challenge to the GKD during the rekeying process. That makes the complexity of the rekeying messages to be O(t).

Naranjo et al. [20] extend the Extended Euclidean algorithm to develop a suite of algorithms for key distribution and authentication (SAKDA) in centralized secure multicast environments. The SAKDA allows the GKD to renew a group key by a single multicast message. However, the security is dependent on the practical difficulty of factoring a private secret value which is the product of t large and different primes. It is not affordable for large groups and it has heavy computation overhead.

In this paper, we propose a secure authenticated group key management based on (2, 2) secret sharing [10] to improve the drawbacks of the above schemes. By our protocol, no key tree is needed, and each member can generate a group key by itself after receiving the rekeying message multicast by the GKD. No message is required to be sent out by members in contrast with the work of Harn and Lin [4]. Only one multicast message is generated at our rekeying scheme. During the self-generation of group key process, each member and the GKD perform mutual authentication implicitly. Our GKD generates a unique polynomial of degree one for

Copyright ⓒ 2014 KSII

each member, and the group key is stored at the constant term of the polynomial. By the secret sharing scheme, the group key can be reconstructed when two shadows of the polynomial are combined together. That using Lagrange interpolation [10] to recover the group key makes our scheme have better computation performance in comparison with the works of AGKTP [4] and SAKDA [20].

The rest of the paper is organized as follows. Section 2 describes the concept of the proposed protocol. Section 3 illustrates our protocol in detail. Section 4 and Section 5 are security analysis and performance analysis respectively. Finally, the conclusion remark is given in Section 6.

2. Overview

Our system has a trusted GKD to setup all system parameters, generate a pre-shared key when a user registers at the system, and enforce mutual authentication between the GKD and each registered user while executing the rekeying process. Henceforth a registered user is called as a member whenever the user joins a group.

In our protocol, the GKD maintains a pre-shared key K_i with each registered user i and requires each member by itself to generate a group key after receiving a rekeying message. It is assumed that a group is consisted of t members whenever a group key GK is required. The first step for the GKD is to generate t polynomials of degree 1 as the form $f_i(x)=a_ix+GK$, where i=1, 2,…, t. Each $f_i(x)$ is designed as a line passing through the two points (0, GK) and (K_i, $H(K_i||T)$), where T is a timestamp. Note that the coefficient a_i must be computed as $a_i=K_i^{-1}\cdot(H(K_i||T)\text{-}GK)$. After that, the GKD selects a random number R and multicasts T, R and $f_i(R)$ (for i=1 to t) to the group members.

Fig. 1 illustrated that each member regards the group key GK as a secret of a polynomial of degree one. By (2, 2) secret sharing scheme, the group key GK can be reconstructed by each member only when two shadows of the polynomial are combined together. In our protocol, the GKD generates a unique polynomial f_i for each member i, and then reveal one shadow for each polynomial f_i, that is, (R, $f_i(R)$). Member i by itself can derive another shadow (K_i, $H(K_i||T)$) based on its pre-shared key K_i and the timestamp T from the multicast message sent by the GKD. After having the two shadows of its specific polynomial, each member derives the group key GK by applying Lagrange interpolating polynomial [10]. The detail of our protocol is described at the next section.

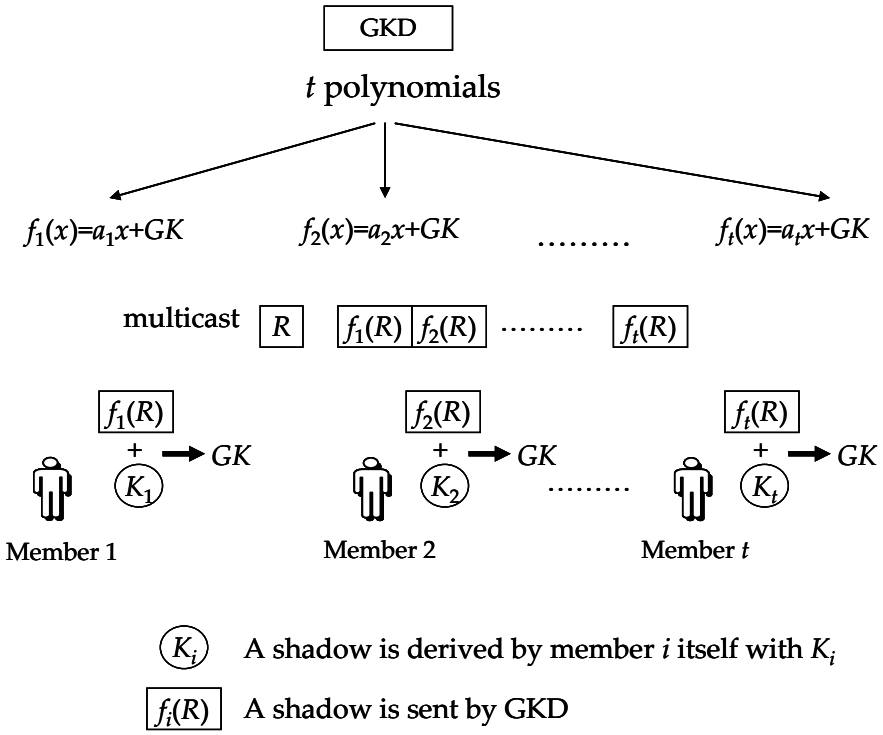


Fig. 1. Self-generation of group key *GK* by (2, 2) secret sharing scheme

3. Our Protocol

Our protocol consists of four processes: system initialization, group creation, member join and member leave. Table 1 illustrates the notations used in this paper.

Table 1. Notations

Notation	Description
ID_i	the real identity of user *i*.
K_i	a pre-shared key between user *i* and GKD.
T	a timestamp.
Auth	an authentication code.
E_K [*data*]	*data* being encrypted by a key *K*.
D_K [*data*]	*data* being decrypted by a key *K*.
GK	a group key.

3.1 System Initialization

The GKD announces a prime number p, a one-way hash function $H()$: $\{0, 1\}^* \rightarrow Z_p^*$, and a symmetric encryption algorithm E. Before joining any group, a user i, by presenting his/her real identity ID_i, must register to GKD to get his/her pre-shared key K_i through a secure

Copyright ⓒ 2014 KSII

channel. **Fig. 2** illustrates the message exchanged between the GKD and user i during the registration process. After that, GKD and user i must keep K_i secretly.

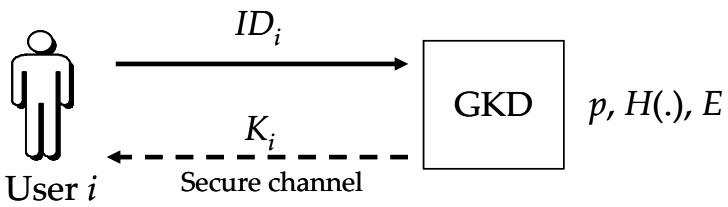
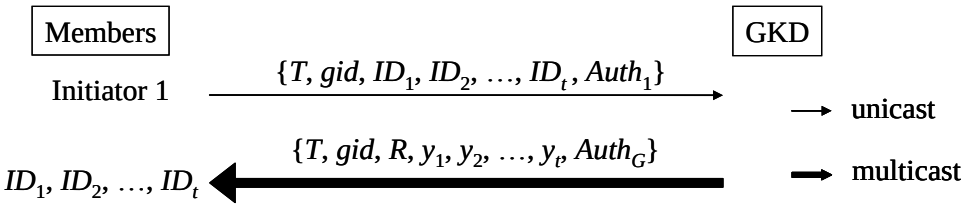


Fig. 2. User registration messages

3.2 Group Creation

A member who creates a new group is called as Group Initiator. For the sake of simplicity, we assume that Group Initiator is ID_1. An Initiator sends a group creation request message, including a timestamp T, group identity gid, a list of group members $\{ID_1, ID_2, \ldots, ID_t\}$ and an authentication code $Auth_1 = H(T||gid||ID_1||ID_2||\ldots||ID_t||K_1)$, to GKD. **Fig. 3** shows the messages exchanged in the group creation process.



Each member i applies (R, y_i) and $(K_i, H(K_i||T))$ to get new group key $GK=f_i(0)$ by using Lagrange Interpolation polynomial.

$Auth_1=H(T||gid||ID_1||ID_2||\ldots||ID_t||K_1)$

$Auth_G=H(GK||T||gid||R)$

For each member i, GKD generates $f_i(x)$ passing through $(0, GK)$ and $(K_i, H(K_i||T))$, and then computes $y_i=f_i(R)$.

Note that, there have t distinct polynomial function $f_i(x)$ and t distinct y_i values.

Fig. 3. Group creation messages

Upon receiving a group creation request message, GKD executes the following steps:

Step 1 : Check the validation of the timestamp T and $Auth_1$.

Step 2 : Generate a group key $GK \in Z_p^*$ and select a random number $R \in Z_p^*$, such that $GK \neq H(K_i||T)$ and $R \neq K_i$, for $i=1, 2, ..., t$.

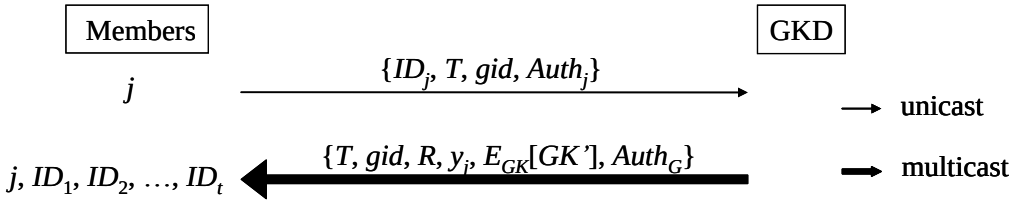
Step 3 : Generate a polynomial $f_i(x)$ of degree 1 for each member i ($i=1, 2, \ldots, t$). $f_i(x)$ can be regarded as a line passing through the two points $(0, GK)$ and $(K_i, H(K_i||T))$, i.e. $f_i(x)= K_i^{-1} \cdot (H(K_i||T)-GK) \cdot x+GK \bmod p$. Then compute $y_i=f_i(R)$, where $i=1, 2, ..., t$. Note that there are t distinct f_i polynomials and t corresponding y_i values.

Step 4 : Compute $Auth_G=H(GK||T||gid||R)$ and multicast $\{T, gid, R, y_1, y_2, \ldots, y_t, Auth_G\}$ to the group members $(ID_1, ID_2,\ldots ID_t)$.

Each member i (i=1, 2,…, t) derives the shadow $(K_i, H(K_i||T))$ by itself and gets another shadow (R, y_i) from the multicast message sent by the GKD. After that, each member i can reconstruct the group key GK by applying the equation: $GK=f_i(0)=((K_i \cdot y_i)-(R \cdot H(K_i||T))) \cdot (K_i-R)^{-1} \bmod p$. The last step for member i is to check whether the $Auth_G$ is equal to $H(GK||T||gid||R)$ to validate the integrity and authenticity of GK.

3.3 Member Join

Suppose that a member j wants to join a group gid which is composed of t group members (ID_1, ID_2, …, ID_t). The member j sends a join request message containing its identity ID_j, a timestamp T, the group identity gid, and an authentication code $Auth_j = H(T||gid||K_j)$ to the GKD. **Fig. 4** shows the messages exchanged in the member joining process.



Joining member j applies (R, y_j) and $(K_j, H(K_j||T))$ to get new group key $GK'=f_j(0)$ by using Lagrange Interpolation polynomial.

Group member i (i=1, 2,…, t) decrypts $E_{GK}[GK']$ to get new group key GK'.

$Auth_j=H(T||gid||K_j)$

$Auth_G=H(GK'||T||gid||R)$

For the joining member j, GKD generates $f_j(x)$ passing through $(0, GK')$ and $(K_j, H(K_j||T))$, and then computes $y_j=f_j(R)$.

GKD sends a multicast message containing y_j and encrypted GK' to all group members.

Fig. 4. Member joining messages

Upon receiving a join-requesting message from member j, GKD executes Step 1 and Step 2 of the group creation process at first. The new group key is denoted as GK'. After that, the GKD only needs to generate one polynomial $f_j(x)$ of degree 1 passing through $(0, GK')$ and $(K_j, H(K_j||T))$ and compute a corresponding shadows $y_j=f_j(R)=K_j^{-1} \cdot (H(K_j||T)-GK') \cdot R+GK' \pmod p$ for the joining member j. GKD sends a multicast message $\{T, gid, R, y_j, E_{GK}[GK'], Auth_G\}$ to the joining member j and member i (i=1, 2, …, t), where $Auth_G=H(GK'||T||gid||R)$. Note that the new group key GK' is encrypted by the old group key GK in the multicast message for existing group members. In this way, the multicast message size is smaller than that in the group creation process.

After receiving $\{T, gid, R, y_j, E_{GK}[GK'], Auth_G\}$ from GKD, the joining member j generates the new group key by $GK'=f_j(0)=((K_j \cdot y_j)-(R \cdot H(K_j||T))) \cdot (K_j-R)^{-1} \bmod p$. As for each existing member i (i=1, 2, …, t), it executes $D_{GK}[E_{GK}[GK']]$ to get the GK'. After that, each member

Copyright ⓒ 2014 KSII

checks whether the $Auth_G$ is equal to $H(GK'||T||gid||R)$ to validate the integrity and authenticity of GK'.

3.4 Member Leave

The member leave process can be treated as the group creation process. It is assumed that the group consists of t members (ID_1, ID_2,…, ID_t) after a member leaving the group. The first member ID_1 is required to send a group creation request message to the GKD to renew the group key.

4. Security Analysis

In this section, we analyze the security of the proposed protocol. We first show that the protocol enforces mutual authentication between the GKD and each group member while executing the rekeying process, and assures forward secrecy and backward secrecy. Then we describe that the protocol is able to resist from replay attack, impersonating attack, group key disclosing attack and malicious insider attack.

4.1 Mutual Authentication between GKD and Group member *i*:

The proposed protocol requires that each member i must register to GKD to get a pre-shared key K_i, which is shared by the member i and GKD only. During group creation phase, the GKD authenticates a group initiator by validating the authentication code $Auth_1$ containing the pre-shared key K_1. For authenticating the other members i in the group, the GKD divides the group key GK into two shadows (K_i, $H(K_i||T)$) and (R, y_i). Only the legal member i with K_i can recover the group key GK. It indicated that a member i is authenticated by the GKD implicitly, though members authenticate the GKD by $Auth_G$.

At the member join phase, the GKD authenticates the joining member j by validating the authentication code $Auth_j$. For authenticating the other existing members, the GKD uses the old group key GK to encrypt the new group key GK'. Only the legal members with the old GK can decrypt $E_{GK}[GK']$ to be authenticated by the GKD implicitly. After members get the group key GK, they authenticate the GKD by $Auth_G$.

4.2 Forward Secrecy

When a member leaves a group, the group creation process is executed to renew the old group key. The ID of the leaving member is excluded from the group creation request message. Since no shadow of the new group key for the leaving member is generated by the GKD, the leaving member can not recover the new group key even he/she eavesdrops the multicast message.

4.3 Backward Secrecy

When a member joins a group, the GKD randomly generates a new group key. It is impossible that the new group key can be inferred from the old group key since they are totally irrelevant.

4.4 Replay Attack

In our protocol, the messages are designed to contain a timestamp T and an authentication code $Auth$ to resist replay attacks.

4.5 Impersonating Attack

Neither member impersonating attack nor GKD impersonating attack could succeed in our protocol. An attacker can't impersonate a group member i unless the attacker got the pre-shared key K_i to generate a valid authentication code $Auth_i$. The same reason is given for an attacker to impersonate the GKD successfully only if the attacker got all the pre-shared keys of a group.

4.6 Group Key Disclosing Attack

In our protocol, the group key is recovered by each member individually. An attacker without knowing any pre-shared key can not recover the group key even eavesdroping the rekeying message $\{R, y_1, y_2, \ldots, y_t\}$. It is shown by the property of the linear underdetermined system. A system of linear equations is called underdetermined if there are fewer equations than unknowns, and an underdetermined system has either no solution or infinitely many solutions. By eavesdroping the rekeying message, the attacker has a system of t equations of the form $y_i=a_i \cdot R+GK \bmod p$. The attacker has t+1 unknows ($a_1,\ldots, a_t$ and GK). In our protocol, $a_i=K_i^{-1} \cdot (H(K_i||T)-GK)$, when p is large enough or p is kept secret, it is infeasible for an attacker to know the GK without any pre-shared key K_i.

4.7 Malicious Insider Attack

In this attack, we consider that a member i attempts to find out the pre-shared key K_j of a member j. Since the member i by itself can recover the group key GK, and also knows the y_j by the message multicast by the GKD. Recall that $f_j(x)$ is designed as a line pass through the two points (0, GK) and (K_j, $H(K_j||T)$). The member i can reconstruct the line $f_j(x)$ of member j by applying the two points (0, GK) and (R, y_j), and then try all possible points under Z_p^* to figure out the point (K_j, $H(K_j||T)$). To resist such an attack, the size of p must be large enough to make it infeasible for a member to find out the pre-shared key of other member.

5. Performance Analysis

In this section, we compare the performance of the proposed protocol with the GKMP [19], the AGKTP [4] and the SAKDA [20], in terms of the number and the size of the rekeying messages, the number of stored keys and the computation overhead during rekeying process.

5.1 Rekeying Messages

Table 2 gives the comparison results of the number of the messages and the size of a broadcast/multicast of the works of GKMP [19], AGKTP [4], SAKDA [20] and ours.

Table 2. The compare results of rekeying messages

Scheme	The number of rekeying messages	The size of messages sent by the GKD
GKMP [19]	t unicast	tl
AGKTP [4]	t unicast +2 broadcast	$(3t+1)l$
SAKDA [20]	1 multicast	$(2t+3)l$
Our Scheme	1 multicast	$(t+4)l$ **or** $6l$

t: the number of group members
l: bit length of each parameter in a multicast/broadcast message

Copyright ⓒ 2014 KSII

- **The number of rekeying messages**

  At the rekeying process, the GKD of the GKMP [19] sends out t unicast messages to members. The AGKTP [4] requires each member to send a challenge message to the GKD after receiving a broadcast message, and then the GKD of the AGKTP broadcasts a message containing t public points of their polynomial $f(x)$ of degree t. Thus the AGKTP needs 2 broadcasts and t unicasts while a group membership changes. As for the SAKDA [20] and our scheme, the distribution and renewal of a group key can be done in a single multicast message sent by the GKD without any challenge messages of group members.

- **The size of the rekeying messages sent by the GKD**

  The last column of Table 2 gives the size of messages sent by the GKD, and the bit length of each parameter in the rekeying message is assumed to be l. Each unicast message of GKMP [19] contains a group key encrypted by a member's preshared key. Thus, the size of messages sent by the GKMP is $t \cdot l$. Recall that the GKD in AGKTP needs 2 broadcasts while a group membership changes. The first broadcast is a list of all group member. The second broadcast contains an authentication code and t points (x-coordinate and y-coordinate). Thus, the size of the messages sent by the AGKTP is $tl + (2t+1)l = (3t+1)l$. In the SAKDA [20], there has a special L which is the product of member tickets of all group members, and then the message contains the modular multiplicative inverse of L. Thus, the bit length of the modular multiplicative inverse of L is $t \cdot l$. The size of the rekeying message including an authentication code in SAKDA is $(2t+3)l$. As for our scheme, the message size is $(t+4)l$ for the group creation process. The message size is reduced to $6l$ for our member leaving process. The message size of our scheme is half of the works of the AGKTP [4] and the SAKDA [20].

5.2 The Number of Stored Keys

In AGKTP [4], the GKD shares a pair of secrets (x_i, y_i) with each member i. It means that each group member needs to store 2 keys, and the GKD needs to store $2t$ keys for a group with t members. Each member in GKMP [19], SAKDA [20] and our scheme is required to store only one key and the GKD needs to store t keys for a group with t members. Thus, the GKMP, the SAKDA, and our scheme require less number of keys than the work of the AGKTP. The comparison result is shown in Table 3.

Table 3. The number of stored keys

Scheme	GKD	Each group member
GKMP [19]	t	1
AGKTP [4]	$2t$	2
SAKDA [20]	t	1
Our Scheme	t	1

t: the number of group members

5.3 Computation Overhead during Rekeying Process

Table 4 depicts the computation overhead of the GKD and each group member during rekeying process. For simplicity, we only consider the computation overhead for generating a polynomial, restoring the constant term of a polynomial by using Lagrange interpolation, and computing the modular multiplicative inverse of L.

- **Computation Overhead of GKD**

  In the GKMP, the GKD needs to use t pre-shared keys to encrypt the gorup key for members. The GKD of the AGKTP [4] needs to construct a polynomial $f_t(x)$ with degree t. In the SAKDA [20], the GKD executes two modular multiplicative inverse operations by using Extended Euclidean Algorithm. In our scheme, the GKD generates t polynomials with degree 1 for the group creation process. As for the member join process, the GKD generates one polynomial of degree for the joining member and encrypts the new group key by the old group key.

- **Computation Overhead of Each Member**

  In the GKMP, each member must decrypt the received unicast message to get the gorup key. Each member in AGKTP [4] needs to restore the constant term of the polynomial $f_t(x)$ with degree t by applying Lagrange interpolation [10] to get the group key. In the SAKDA [20], each member executes one modular multiplicative inverse operation and one modular exponentiation to get the group key. In the group creation process of the proposed scheme, since our polynomial is degree 1, each member has less computation overhead than the AGKTP to restore the constant term of $f_1(x)$ to get the group key. As for the member join process, each member executes the symmetric decryption to get the new group key, and the joining member executes the same operation in the group creation process.

Table 4. The computation overhead

Scheme	GKD	Each group member
GKMP [19]	t *enc*	1 *dec*
AGKTP [4]	1 *Gen_*$f_t(x)$+1 *hash*	1 *Gen_*$f_t(0)$+ 1 *hash*
SAKDA [20]	2 *inv*+1 *hash*	1 *inv* +1 *exp*+1 *hash*
Our Scheme	t *Gen_*$f_1(x)$+1 *hash* **or** 1 *Gen_*$f_1(x)$+1 *hash* +1 *enc*	1 *Gen_*$f_1(0)$+1 *hash* **or** 1 *dec*+1 *hash*

t: the number of group members
*Gen_*$f_d(x)$: generate a polynomial of degree d [4]
*Gen_*$f_d(0)$: restore the constant term of $f_d(x)$ [4]
inv: modular multiplicative inverse
exp: modular exponentiation
enc/dec: symmetric encryption/decryption

Copyright ⓒ 2014 KSII

6. Conclusion

In this paper, we design a secure authenticated group key management scheme based on (2, 2) secret sharing technology without maintaining a key tree as the works of Wuu and Chen [12] and Yu et al. [15] to reduce the number of rekeying messages to be one. Each member in our protocol only needs to maintain a pre-shared key safely, and then apply the pre-shared key to recover a group key by itself at the rekeying process. The proposed scheme enforces mutual authentication, forward secrecy and backward secrecy properties, and resists replay attack, impersonating attack, group key disclosing attack and malicious insider attack. In comparison with the works of Harn and Lin, [4] and Naranjo et al. [20], our protocol has smaller multicast message size, and less key storage requirement as well as the computation overhead.

The proposed member joining protocol uses the current group key to encrypt the new one. An attacker could have all future group keys for this session if the attacker had gotten the currect group key. To overcome the problem, a simple way is to apply the group creation process whenever a member joins. However, the enhanced security of the group key is at the expsense of the efficncy of the member joining process. It would be our future work to improve our protocol to have both efficiency and security.

Acknowledgment

The authors would like to thanks the anonymous reviewers for their valuable comments. The authors would also like to thanks the National Science Council of the Republic of China, Taiwan for financially supporting this research.

References

[1] Adusumilli P, Zou X, and Ramamurthy B., "DGKD: distributed group key distribution with authentication capability," in *Proc. of the IEEE Workshop on Information Assurance and Security*, pp. 286-293, June 15-17, 2005. Article (CrossRef Link).

[2] Ateniese G, Steiner M, Tsudik G. "New multiparty authentication services and key agreement protocols, " *IEEE Journal on Selected Areas in Communications* , Vol. 18, No.4, pp.628-639, April, 2000. Article (CrossRef Link).

[3] Daza V, Herranz J, Saez G. "On the computational security of a distributed key distribution scheme," *IEEE Transactions on Computers*, Vol.57, No.8, pp.1087-1097, August, 2008. Article (CrossRef Link).

[4] Harn L, Lin C. "Authenticated group key transfer protocol based on secret sharing," *IEEE Transactions on Computers*, Vol.59, No.6, pp.842-846, June, 2010. Article (CrossRef Link).

[5] Kwak DW, Kim J. "A decentralized group key management scheme for the decentralized P2P environment," *IEEE Communications Letters*, Vol.11, No.6, pp.555-557, June, 2007. Article (CrossRef Link).

[6] Ng WHD, Cruickshank H, Sun Z. "Scalable balanced batch rekeying for secure group communication," *Computers and Security*, Vol.25, No.4, pp.265-273, June, 2006. Article (CrossRef Link).

[7] Ng WHD, Howarth M, Sun Z, Cruickshank H. "Dynamic balanced key tree management for secure multicast communications," *IEEE Transactions on Computers*, Vol.56, No.5, pp.590-605, May, 2007. Article (CrossRef Link).

[8] Parvatha Varthini B, Valli S. "Generation of group key using enhanced one way function tree group rekey protocol," in *Proc. of Int. Conf. on Computing: Theory and Applications*, pp.176-181, March 5-7, 2007. Article (CrossRef Link).
[9] Pham T, Watters PA. "The efficiency of periodic rekeying in dynamic group key management," in *Proc. of 4th European Conf. on Universal Multiservice Networks*, pp.425-432, February, 2007. Article (CrossRef Link).
[10] Shamir A. "How to share a secret," *Communications of the ACM*, Vol.22, No.11, pp. 612-613, November, 1979. Article (CrossRef Link).
[11] Sun Y, Liu KJR. "Hierarchical group access control for secure multicast communications," *IEEE/ACM Transactions on Networking*, Vol.15, No.6, pp.1514-1526, December, 2007. Article (CrossRef Link).
[12] Wuu LC, Chen HC. "A scalable framework for secure group communication," in *Proc. of First International Conference on Networking-Part 2*, pp.225-238, 2001. Article (CrossRef Link).
[13] Xu L, Huang C. "Computation-efficient multicast key distribution," *IEEE Transactions on Parallel and Distributed Systems*, Vol.19, No.5, pp.577-587, May, 2008. Article (CrossRef Link).
[14] Yi X. "Authenticated key agreement in dynamic peer groups," *Journal of Theoretical Computer Science*, Vol.326, No.1-3, pp.363-382, October, 2004. Article (CrossRef Link).
[15] Yu W, Sun Y, Liu KJR. "Optimizing rekeying cost for contributory group key agreement schemes," *IEEE Transactions on Dependable and Secure Computing*, Vol.4, No.3, pp.228-242, 2007. Article (CrossRef Link).
[16] Je DH, Lee JS, Park Y, Seo SW. "Computation-and-storage-efficient key tree management protocol for secure multicast communications," *Computer Communications*, Vol.33, No.2, pp.136-148, 2010. Article (CrossRef Link).
[17] Kulkarni SS, Bruhadeshwar B. "Key-update distribution in secure group communication," *Computer Communications*, Vol.33, No.6, pp.689-705, April, 2010. Article (CrossRef Link).
[18] Wu W, Li M, Chen E. "Optimal tree structures for group key tree management considering insertion and deletion cost," *Theoretical Computer Science*, Vol.410, No.27-29, pp.2619-2631, June, 2009. Article (CrossRef Link).
[19] Harney H, Muckenhirn C, Rivers T. "Group key management protocol (GKMP) architecture," RFC 2094, IETF 1997. Article (CrossRef Link).
[20] Naranjo JAM, Antequera N, Casado LG, López-Ramos JA. "A suite of algorithms for key distribution and authentication in centralized secure multicast environments," *Journal of Computational and Applied Mathematics*, Vol.236, No.12, pp.3042-3051, June, 2012. Article (CrossRef Link).

Copyright ⓒ 2014 KSII

Lih-Chyau Wuu received her B.S. degree in the department of information engineering from National Taiwan University, Taipei, Taiwan, in 1982, and her Ph.D. degree in the department of computer science from National Tsing Hua University, Hsinchu, Taiwan in 1994. She is currently a Professor in the department of computer science and information engineering, National Yunlin University of Science & Technology, Touliu, Taiwan. Her research interests include IP switches /routing, multicast routing, network security and distributed self-stabilizating systems.

Chi-Hsiang Hung received his B.S. and M.S. degrees in the department of electronic engineering from National Yunlin University of Science & Technology (Touliu, Taiwan), in 2003 and 2006, respectively. Then He received the Ph. D. degree from graduate school of engineering science and technology-doctoral program, National Yunlin University of Science & Technology, in 2013. He is an Post-Doctoral Researcher in Department of Electrical and Computer Engineering, National Chiao Tung University. His research interests include network security, information security and software-definded network.

Wen-Chung Kuo He received the B.S. degree in Electrical Engineering from National Cheng Kung University and M.S. degree in Electrical Engineering from National Sun Yat-Sen University in 1990 and 1992, respectively. Then, He received the Ph.D. degree from National Cheng Kung University in 1996. Now, he is an associate professor in the Department of Computer Science and Information Engineering, National Yunlin University of Science & Technology, Touliu, Taiwan. His research interests include steganography, cryptography, network security and signal processing.