그룹키 관리를 위한 키트리 모델

한근희*, 정태의**, 윤여웅*
*공주대학교 응용수학과
**서경대학교 컴퓨터과학과
***충북대학교 컴퓨터학과
e-mail : tejeong@seokyeong.ac.kr

The Key Tree Model for Group Key Management

Keunhee Han*, Taeeui Jeong **, Yeowung Yun ***

*** Department of Applied Mathematics, Kongju National University**
**** Department of Computer Science, Seokyeong University**
***** Department of Computer Science, Chungbuk National University**

Abstract

For secure communications in using multicast applications such as Cable-TV, It is essential for us to manage shared keys to encrypt/decrypt data through crypto algorithm as DES, which is called Group Key Management. In GKM, It is a hot issue that reduces the number of join/leave operation and subgroup key in key tree model. In this paper, we propose optimized mechanism of group key management required for providing multicast security.

1. Introduction

Multicast circumstances include many issues like secure communications. To hide data from all but authorized receivers, you need to encrypt the data. But how do you efficiently distribute a key to thousands of receivers? In multicast environment, Key management is an essential part for secure communications. To securely communicate between any two communicating peers, both peers must share a secret key.

The technical problems involved in secure multicast arise from the fact that most multicast applications such as tele-conferencing application require real time processing of data. To encrypt/decrypt data quickly any multicast application can not afford to adapt asymmetric cryptography such as RSA. Hence, a single symmetric cryptographic key, called a group key, must be shared among all the users in multicast group for encrypting/decrypting multicast data. This group key need to be changed as the users of the group join and leave the group so that a new user can not decrypt the messages that transferred before that user joins the group and a leaving user can not decrypt the messages that will be transferred after the user leaves the group.

In multicast environment if the multicast data need to be secure, then the users must share a common group (secret symmetric) key, which is used to encrypting/decrypting all the group communication. The key management scheme used for unicast may also be used for multicast; however, it is not practical. For example, if there are n members in a multicast group, distributing the group key securely to all n members requires n messages encrypted by the n different individual keys. This means that the computational cost is proportional to group size n. Therefore, we certainly need special key management mechanisms for secure multicast.

In chapter 2 of this paper, we will introduce several existing group key management schemes and in chapter 3, especially we will survey centralized group key management in detail. Also, in chapter 4, we prove that constructing a group key tree model that contains optimum performances on both join and leave

operation is impossible if the group size is greater than two. Based on these observations, we propose a new tree structure that may be used to improve the performance of any tree-based centralized group key management mechanisms. In chapter 5, we will evaluate proposed group key management mechanism and draw a conclusion.

2. Introduction of Group Key Management

When we design a group key management scheme we must try to minimize the time required for initial setup, storage requirement for each user and the group key server(s), and the total number of transmissions required for initial setup, rekey and maintenance. These minimization efforts are very important for any group key management schemes to be successful because multicast applications, for example, a video conferencing, requires real-time processing.

We need several definitions for group key management[9][10]. Join is an operation that adds a new user to an existing multicast group. Leave is an operation that evicts a member from multicast group that the leaving user belongs to. To achieve a high level of security, the group key should be changed after every join and leave operations so that a leaving group member can not decrypt any further group messages and newly joining member can not decrypt any previous group messages. These security requirements of multicast are commonly referred as "Backward and Forward Secrecy" in literatures. Rekey is an operation that changes the current group key to a new group key. Note that not every join/leave operations followed by rekey operations. Security policy of a group key management system will decide if rekey is necessary for every join/leave operations.

There are two kinds of group key management schemes: Centralized Group Key Management and Distributed Group Key Management. This categorization is based on the number of key management servers involved in the group key management schemes. Group key servers are responsible for the creation, distribution and management of group keys. Centralized group key management involves only one group key server while distributed group key management involves more than one group key servers. Members in centralized group key management need to trust their unique group key server while members in Distributed group key management need to trust multiple group key servers. Generally, centralized group key management scheme provides more secure environment and efficient rekey operations than distributed group key management scheme. However, in terms of scalability, distributed group key management is far better than centralized group key management.

Examples of the distributed group key management are as follows:

- SMKD (Scalable Multicast Key Distribution) ([1]),
- MKMP (Multicast Key Management Protocol) ([2]),
- Iolus ([3]), and
- Intra-domain group key management ([4]),

Examples of the centralized group key management are as follows:

- Logical Key Hierarchy ([5]),
- Key Graph ([6]), and
- One-Way Function Tree ([7]).

3. Characteristics of Centralized Group Key Management

In section 3.1 we introduced three leading centralized group key management. All these three management schemes use tree hierarchy as their base architecture. Among these schemes, the two mechanisms, LKH ([5]) and Key Tree ([6]), actually suggest the same group key management technique. Hence, we will call these two schemes as "Key Tree".

key tree is a special graph of key graph such that it contains no cycles. Note that a connected graph G is a tree if and only if it contains no cycles. Figure 1 shows an example of a key tree that contains four user nodes u_1, u_2, u_3, and u_4. The key contains in the root of the tree (k_{1234}) is the group key. Keys k_1 through k_4 represents the individual keys for the users u_1 through u_4, respectively. Note that these individual keys are not shared among the users.

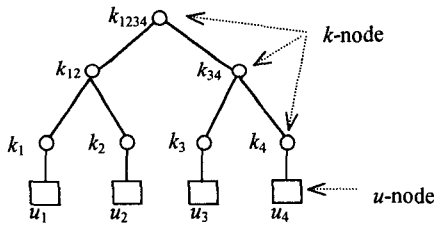

Figure 1 : A Key Tree

Note that group key server must store and manage all the keys used in the key tree. However, each user needs to store only the keys in the path between itself and the root of the tree. For example, in the figure 1, user u_2 needs to store keys $\{k_2, k_{12}, k_{1234}\}$.

The height h of a key tree is the length of the longest path from the root to any key node that contains individual key. The degree d of a key tree is the number of children nodes of a node that contains the greatest number of children among the internal nodes.

4. Optimization of Key Tree Model

In the previous chapter we introduced the key tree model and analyzed its performance on join and leave operations. The following table summarizes the performance of join and leave operations on key tree model.

Table 1 : Performance of Key Tree Model

Operations	Modes	Number of rekey messages	Enc. Costs
Join	user-oriented	h + 1	(h + 1)(h + 2)/2 – 1
	key-oriented	h + 1	2h
Leave	user-oriented	h(d – 1)	h(d – 1)(h + 1)/2
	key-oriented	h(d – 1)	hd – 1

From the above table, it is clear that the performances of key-oriented mode are much better than the performance of user-oriented mode. We will assume that the key server adapts key-oriented mode when sending rekey messages.

The key tree models we have analyzed up to now are perfectly balanced tree[8]. However, by varying the degree of internal nodes at different heights, we may be able to optimize the performance of key trees. Also, it is desirable if we can reduce the number of internal nodes since all the internal nodes must be stored in the key server and the key server must generate and manage the subgroup keys assigned to the internal nodes. The following table shows the performance and required resources for the various key tree models that use perfectly balanced trees to accommodate 100,000 users.

Table 2 : Performance of various key tree models with group size 100,000

Degree	Height	Rekey messages for a join operation	Rekey messages for a leave operation	Subgroup keys required	Max. group size
2	17	18	17	131,070	131,072
3	11	12	22	88,572	177,147
4	9	10	27	87,380	262,144
5	8	9	32	97,655	390,625
6	7	8	35	55,986	279,936

Note that the tree models used in the above tables are perfectly balanced tree. We will use the notation x-Tree as the tree with degree x. This example shows that the shorter tree generates less number of rekey messages required for a join operation while the longer tree generates less number of rekey messages required for a leave operation.

From the above analysis we can see that any tree model with fixed degree does not offer optimal performance in all aspects. However, the required number of rekey message for a leave operation for this model is more than double than that of the 2-Tree model. The following theorems generalize these observations.

Lemma 4.1 : For any key tree model that contains n users and generates L rekey messages for a leave operation, there exists a 2-Tree model (i.e., binary tree) that can contains at least (n + 1) users and generates less than or equal to L rekey messages for a leave operation.

From the above theorem the following theorem is immediate:

Theorem 4.1 : Binary tree model generates the least number of rekey messages for a leave operation among all group key management models based on tree.

Any group key server model based on tree structure must send at least one message if the group is empty and two messages if the group is not empty when the join operation occurs. Therefore, the following theorem is immediate:

Theorem 4.2 : n-Tree model generates the least number of rekey messages for a join operation among all group key management models based on tree and manages n (≥ 1) users.

Theorem 4.1 and 4.2 indicate that n-Trees contain the optimum performance for join operations and 2-Trees contain the optimum performance for leave operations. These observation leads to the fact that there exists no tree structure that contains optimum performance for both join and leave operations if the group size is larger than three.

The following is the algorithm that finds an optimal tree structure that can accommodate up to n users.

Algorithm 4.1 : OptimizedKeyTree

Input: The group size, n.

Output: Degree sequence for the optimized tree structure that can accommodate at least n users.

1 $h = \lceil \log_2 n \rceil$;

2

3 for (i = 1; i <= h; i ++)
4 {
5 maxDeg(i) = h – i + 2;
6 for(j = 1; j <= C(maxDeg + i – 3, i – 1); j++)
7 {
8 if (i == 1) set d_0 = n;
9 else
10 set d_k = 2, where $0 \le k \le i - 2$;
11 set $d_{i-1} = d_0 \times d_1 \times ... \times d_{i-2}$;
12 Let $T_i = T\{d_0, d_1, ..., d_{i-1}\}$ be the tree being considered
13 Compute leaveMsg(T_j), joinMsg(T_j), leaveEnc(T_j), joinEnc(T_j), subkey(T_j)
14 Compare these performances against previous one and select best one.
15
16 if (d_{i-2} != maxDeg)
17 $d_{i-2} = d_{i-2} + 1$;
18 else
19 find minimum k such that d_k is equal to maxDeg;
20 set d_x = 2, where $0 \le x \le k - 1$;
21 set d_y = maxDeg, where $k \le y \le i - 1$;
22 }
23 }

Let's take a running example of the algorithm with n = 20. With n = 20 the algorithm considers the following candidate trees that are represented by their corresponding degree sequences.

Table 3 : Example of the algorithm 4.1 with n = 20

Height	Deg. sequence	leaveMsg(T)	joinMsg(T)	subkey(T)	leaveEnc(T)	joinEnc(T)
4	2, 2, 2, 3	5	5	14	8	8
5	2, 2, 2, 2, 2	5	6	30	9	10

This is an example that there is a non-binary key tree structure that shows the same performance on leave operations but shows better performances for other aspects. Consider the key tree model T{2, 2, 2, 3}. T{2, 2, 2, 3} requires five rekey messages for a leave operation while 2-Tree also requires five rekey messages for a leave operation. However, T{2, 2, 2, 3} shows better performance on join operation than 2-Tree does. Also, T{2, 2, 2, 3} uses less than half subgroup keys than 2-Tree does. T{2, 2, 2, 3} also shows better performance on leaveEnc(T) and joinEnc(T) than 2-Tree does. Therefore, in this case, the algorithm selects T{2, 2, 2, 3} as the optimal key tree structure.

In the previous example, the algorithm selects T{2, 2, 2, 3} as the optimal key tree structure for the group size of 20. But, this optimal structure is based on our assumption that any optimal key tree structure must show the same performance on leave operation as the 2-Tree does. However, if we are able to relax this constraint we can select a key tree structure that shows better performances in overall.

5. Conclusion

This paper proposes optimized mechanism of group key management required for providing multicast security on internet environment. Proposed mechanism is a new tree structure that may be used to improve the performance of any tree-based centralized group key management mechanisms. Also, we confirmed that our optimized tree structure preserve the optimum performance on leave operation while contains reasonable performance on join operation and this optimization's efforts also reduce the number of subgroup keys that the group key server must store and manage.

Since our optimized tree structure preserve the optimum performance on leave operation of centralized group key management, it is necessary for us to research on the variety of group key management in consideration of scalability in the future.

References

[1] A. Ballardie, "Scalable Multicast Key Distribution", RFC1949, May 1996

[2] Harkins D., N. Doraswamy, "A Secure, Scalable Multicast Key Management Protocol(MKMP)"

[3] Mittra S., "Iolus : A Framework for Scalable Secure Multicast", In Proceeding of ACM SIGCOMM'97, Cannes, France, Sep. 1997.

[4] Thomas Hardjono, B. Cain, Indermohan Monga, "Intra-Domain Group Key Management Protocol", internet-draft, draft-ietf-ipsec-intragkm-00.txt.

[5] Debby M. Wallner, Eric J. Harder, Ryan C. Agee, "Key Management for Multicast : Issues and Architectures", internet-draft, draft-wallner-key-arch-01.txt, 1998

[6] Chung Kei Wong, Mohamed Gouda, Simon S. Lam, "Secure Group Communications Using Key Graphs"

[7] "Key Management for Large Dynamic Groups : One-Way Function Trees and Amortized Initialization", internet-draft, draft-balenson-groupkeymgmt-oft.txt

[8] M. J. Moyer, J.R. Rao, and P. Rohatgi, "Maintaining Balanced Key Trees for Secure Multicast", internet-draft draft-irtf-smug-key-tree-balance-00.txt, June 1999.

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

[10] T. Hardjono, B. Cain and N. Doraswamy, " A Framework for Group Key Management for Multicast Security", internet-draft, July 1998.