On Key Distribution in Secure Multicasting

Kuen-Pin Wu
kpw@orchid.ee.ntu.edu.tw

Shanq-Jang Ruan
stj@orchid.ee.ntu.edu.tw

Feipei Lai
flai@cc.ee.ntu.edu.tw

Chih-Kuang Tseng
tck@orchid.ee.ntu.edu.tw

Department of Electrical Engineering
and
Department of Computer Science and Information Engineering
National Taiwan University, Taipei 106, Taiwan

Abstract

Multicasting has been widely utilized for delivering messages from one sender to multiple recipients. Nowadays in some applications such as pay per view or video-conferencing systems, the messages delivered via multicasting should be available to authorized recipients only. Therefore, secure multicasting becomes an important design issue in distributed environment.

To achieve secure multicasting, all authorized recipients form a group and share a group key; messages should be encrypted by the key before they are multicasted. For the sake of security, we need a new group key if someone joins/leaves the group. Consequently, our goal is aimed at finding a way to distribute the new key securely and efficiently.

In this paper, we propose a new solution to the problem; we distribute new keys through a special function called secure filter. Compared with related work, our solution requires less system resources to change group members.

1. Introduction

Multicasting has been widely utilized for delivering messages from one sender to multiple recipients. For instance, in a pay-per-view system, the video server delivers a movie to all subscribers through multicasting. Another example is video-conferencing systems. In such systems, messages should be shown on the electronic white boards of all remote members, and the messages are sent via multicasting. Note that in the two examples, the multicasted material should be only available to authorized users. In other words, they should be protected from external intruders. Therefore, secure multicasting becomes an important design issue in distributed environment.

To achieve secure multicasting, all members in a multicast group share a group key; messages should be encrypted by the key before they are multicasted. On receiving the encrypted messages, group members can extract the plain messages using the group key, while others cannot. The security of such systems is therefore depends on the group keys.

Under this multicasting model, however, the system becomes insecure if someone joins/leaves the multicast group. When someone joins the group, he may access past multicasted messages using his newly obtained group key, but he is not an authorized recipient while these messages were sent. Similarly, after someone leaves the group, he may access succeeding encrypted messages using the group key he holds. He is not an authorized recipient of these messages, either. Consequently, the group key should be changed if someone joins/leaves the group. So, distributing new group keys will take place often in dynamic groups whose membership changes frequently. Our goal is hence aimed at finding a way to distribute new group keys securely and efficiently.

Many schemes have been proposed for secure multicasting. In some static key management protocols [1, 7, 6], all members obtain an unchanged group key as they join the group. These schemes do not apply to dynamic groups. Chiou *et al.* [4] proposed a scheme called secure lock to broadcast securely, which is implemented using Chinese Remainder Theorem (CRT); only authorized members can extract the group session key from the secure lock. Performance issue becomes an important concern in this scheme, since CRT requires complex computation. Iolus [11] divides a large dynamic group into subgroups. Each subgroup requires an extra agent to deal with "key translation." Moreover, Iolus needs to encrypt the new group key for each member separately if a member exits, which takes a lot of computation time. The key management schemes for secure

0-7695-0912-6/00 $10.00 © 2000 IEEE

multicasting in [3, 5, 13] require each of the N members to store $\log(N) + 1$ keys. As the group growing large, the system requires a great deal of keys; it is more difficult to manage so many keys and keep system secure. Cliques [9] provides a way to distribute group session keys in dynamic groups. However, it does not scale well to large groups. Molva *et al.* [12] proposed a scalable solution for dynamic multicast groups. Nevertheless, the scheme has to modify the structure of intermediate components of the multicast communication such as routers or proxies. Waldvogel *et al.* [10] proposed a scalable scheme for dynamic group such that each member holds fixed number of keys even if the group membership changes. In this scheme, if the system uses k bits to represent a member ID, then each member holds $k + 1$ keys. It is obviously that the system has up to 2^k members. The system has to be reorganized if the number of members exceeds 2^k; on the other hand, if the number of members is far less than 2^k, each member will hold relatively too many keys. However, a very low-overhead operation exists to shrink the key space as the system grows.

In this paper, we propose a new key distribution scheme for dynamic groups in secure multicasting; the scheme distributes the new group key through a special function called secure filter.[1] In the proposed scheme, when a user joins/leaves the group, the central authority (CA) of the system first selects a new group key and encapsulates it into a secure filter sf, and then the CA multicasts sf. On receiving sf, the receiver m_j first uses its own secret key k_j to generate the input of sf, and then computes the value of sf. If m_j is an authorized recipient, sf returns the new group key to m_j, otherwise sf returns an invalid value. The secure filter is the building block of our scheme, it greatly reduces the number of required encryption/decryption operations in the key distribution process.

The rest of the paper is organized as follows: Section 2 gives a formal definition of the key distribution problem in secure multicasting. Our scheme is then presented in Section 3. In the section, we first define the secure filter formally and describe its properties, then we use the secure filter to construct our solution. Security concern and performance evaluation of the solution are also given in this section. Section 4 concludes this paper.

2. Key distribution in secure multicasting

A secure multicasting environment consists of users and secure multicast groups. In our model, we assume that each user m_i holds a private secret key k_i to perform secure point-to-point communication. A secure multicast group g_j is composed of more than one user. Moreover, g_j is associated with a group key gk_j. Members of g_j use gk_j to perform secure multicasting. Note that a user can be a member of more than one secure multicast group.

[1] Although not devoted to secure multicasting, the notion that users share secrets using a function can be found in [2].

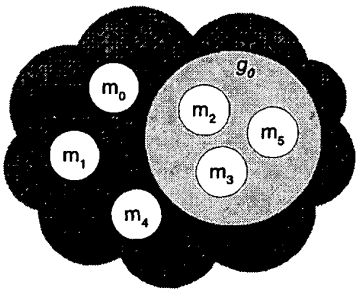

Figure 1. A secure multicasting environment.

There is an example illustrated in Figure 1. In the figure, the secure multicasting environment has six users m_0, m_1,..., m_5. A secure multicast group g_0 with three members m_2, m_3, and m_5 is also defined in the figure. Although not depicted in Figure 1, each user m_i has a private secret key k_i and the secure multicast group g_0 has a group key gk_0. Suppose that the member m_2 wishes to multicast a message, he first encrypts the message using gk_0, and then sends it to m_3 and m_5. On receiving the encrypted message from m_2, m_3 and m_5 use gk_0 to decrypt it.

Problems arise if the membership of g_0 changes. When a user joins g_0, he has to acquire gk_0 to encrypt/decrypt succeeding messages. However, it is possible that the user uses gk_0 to access past messages that are multicasted before he joins the group. This is not allowed because he is not an authorized recipient when the messages were sent.

Similarly, if a member leaves g_0, he is no longer authorized to access the succeeding multicasted messages. However, it is still possible that the member uses gk_0 to access them.

Therefore, group keys should be changed if some members join/leave the secure multicast groups. How to distribute new group keys securely is consequently an important design issue in dynamic groups whose membership changes frequently. Our goal is then primarily aimed at finding a way to distribute the new group key securely and efficiently. Moreover, when we distribute new group keys, we wish the required extra system resources as few as possible.

Our solution to the key distribution problem is presented in the next section. We define a special function as the building block of the solution. Discussions about the security and performance of the solution are also given in the section.

3. Our work

In this section, we first define a special function called *secure filter* as the main building block of our solution to the key distribution problem. Then we describe our solution formally. Security concern and performance evaluation of our solution are also discussed. Finally we give a refinement of our solution.

3.1. Secure filter

A secure filter is a polynomial in the indeterminate x over Galois field $\mathbf{GF}(p)$, p is a public large prime. For the set $C = \{k_0, k_1, \ldots, k_{t-1}\}$ and the number gk, where gk and k_i, for $0 \leq i \leq t-1$, all belong to $\mathbf{Z}_p$, we can construct a polynomial sf such that $sf(x) = (x - h(k_0))(x - h(k_1)) \ldots (x - h(k_{t-1})) + gk$. It is obviously that if k_j belongs to C, $sf(h(k_j)) \equiv gk \pmod{p}$. sf is then regarded as *the secure filter on gk with respect to C*; where h is a randomly chosen one-way hash function. For the sake of security, sf is represented as $a_t x^t + a_{t-1} x^{t-1} + \ldots + a_1 x + a_0$ in the system. Therefore, gk cannot be extracted directly from sf.

In the following section, we use the secure filter to design our solution to the key distribution problem.

3.2. The proposed solution

Our solution to the problem is called KDSM (KDSM stands for Key Distribution in Secure Multicasting). In KDSM, a central authority (CA) is to distribute/manage keys held by users. And further, all users share their private secret keys with the CA. Moreover, the CA holds all group keys. We assume that secret keys and group keys are chosen from $\mathbf{Z}_p$. Furthermore, mathematical operations are to be interpreted modulo p.

In the following two subsections, we describe how the CA works in KDSM when a user joins/leaves a secure multicast group.

Joining operation. There are two ways that the CA can distribute the new group key when a user joins a secure multicast group. The first way is the same as some proposed schemes in related work. When a user m_i wishes to join the secure multicast group g_j, the CA first selects a new group key gk'_j, and multicasts $E_{gk_j}(gk'_j)$[2] to old members of g_j; finally, the CA sends $E_{k_i}(gk'_j)$ to m_i directly. The original members and m_i can obtain the new group key gk'_j by decrypting the received messages using gk_j and k_i respectively. For an intruder, he cannot obtain gk'_j from $E_{gk_j}(gkj')$ or $E_{k_i}(gkj')$ without gk_j and k_i; for m_i, he cannot access past encrypted messages without gk_j. Security requirement is therefore guaranteed.

[2]The notation $E_a(b)$ means that the message b is encrypted by key a.

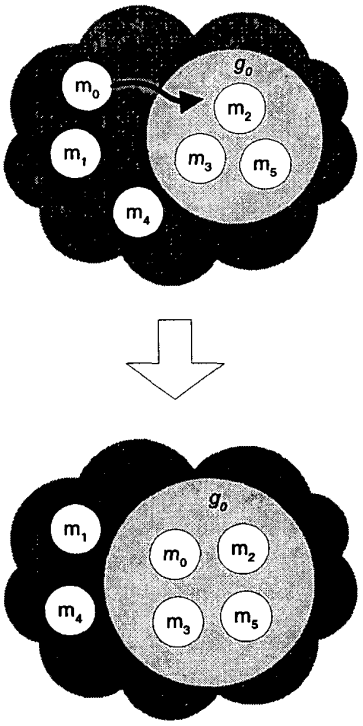

Figure 2. User m_0 joins the secure multicast group g_0.

The second way uses the secure filter. For a secure multicast group g_j, assume that C_j is the set of private secret keys held by all members in g_j. If user m_i wishes to join g_j, the CA selects a new group key gk'_j and a one-way hash function h_j, then the CA constructs a secure filter sf_j on gk'_j with respect to $C_j \cup \{k_i\}$. Finally, the CA multicasts the message (sf_j, h_j) to members of g_j and m_i. On receiving the message, a user m_l first figures out $h_j(k_l)$, then computes $sf_j(h_j(k_l))$. If m_l is a group member of g_j or m_i, k_l must be in $C_j \cup \{k_i\}$ and sf_j will return the new group key gk'_j to m_l; otherwise sf_j returns an invalid value. For an intruder, since his private secret key is not in $C_j \cup \{k_i\}$, he cannot acquire gk'_j form sf_j. Moreover, m_i has no way to acquire gk_j to access past encrypted messages in the scheme.

Example 1: Consider the scenario depicted in Figure 2. A new user m_0 joins the secure multicast group g_0. The CA first selects a new group key gk'_0 and a one-way hash function h_0, then the CA constructs the secure filter $sf_0(x) = (x - h_0(k_0))(x - h_0(k_2))(x - h_0(k_3))(x - h_0(k_5)) + gk'_0$. Finally, the CA sends (sf_0, h_0) to m_0, m_2, m_3, and m_5. It is obviously that for $i \in \{0, 2, 3, 5\}$, m_i can acquire gk'_0 by

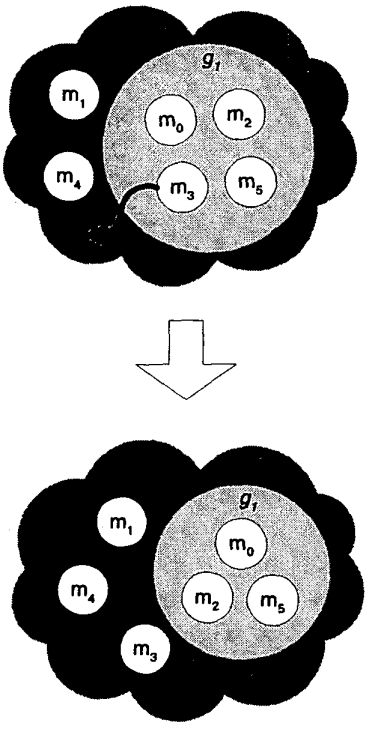

Figure 3. Member m_3 leaves the secure multicast group g_1.

computing $sf_0(h_0(k_i))$.

Leaving operation. For the secure multicast group g_j, we assume that C_j denotes the set of private secret keys held by all members in g_j. If some member m_i of g_j wishes to leave the group, the CA first selects a new group key gk'_j and a one-way hash function h_j, then the CA constructs a secure filter sf_j on gk'_j with respect to $C_j - \{k_i\}$. Finally, the CA multicasts the message (sf_j, h_j) to members of g_j except m_i. On receiving the message, a user m_l first figures out $h_j(k_l)$, then computes $sf_j(h_j(k_l))$. If m_l is a group member other than m_i, k_l must be in $C_j - \{k_i\}$ and sf_j will return the new group key gk'_j to m_l; otherwise sf_j returns an invalid value. For an intruder, since his private secret key is not in $C_j - \{k_i\}$, he cannot acquire gk'_j form sf_j. Moreover, we guarantee that m_i is no longer authorized to access further encrypted messages, because k_i is not in $C_j - \{k_i\}$ and m_i has no way to acquire gk'_j.

Example 2: Consider the scenario depicted in Figure 3. Member m_3 leaves the secure multicast group g_1. The CA first selects a new group key gk'_1 and a one-way hash function h_1, then CA constructs the secure filter $sf_1(x) = (x - h_1(k_0))(x - h_1(k_2))(x - h_1(k_5)) + gk'_1$. Finally, the CA sends (sf_1, h_1) to m_0, m_2, and m_5. It is obviously that for $i \in \{0, 2, 5\}$, m_i can acquire gk'_1 by computing $sf_1(h_1(k_i))$.

3.3. Security concern and Performance Evaluation of KDSM

For security, it requires that unauthorized users cannot acquire the correct group keys to access data. Assume that an unauthorized intruder u wishes to access the multicasted data in the secure multicast group g, he has to obtain the group key gk.

Since u is not a member of g, he has only two possible ways to acquire gk. The first way is that u extracts gk directly from the intercepted secure filter $sf = (x - h(k_{i1}))(x - h(k_{i2}))\ldots(x - h(k_{ij})) + gk$. u can only obtain gk by extracting it from the constant term of sf: $(C + gk) \bmod p$, where $C = h(k_{i1})h(k_{i2})\ldots h(k_{ij})$. Without knowing the numbers $h(k_{i1})$, $h(k_{i2})$, ..., and $h(k_{ij})$, it is infeasible for u to obtain gk, because p is a large prime.

The second way is that u first acquires some authorized member m_{in}'s private secret key k_{in}, and then computers $sf(h(k_{in}))$ to obtain gk. Because m_{in} keeps k_{in} private, the only way that u may obtain k_{in} is to extract it from the intercepted sf. To extract k_{in} from sf, u has to extract $h(k_{in})$ first. Whether u can get $h(k_{in})$ or not, it is infeasible that u can extract k_{in} from $h(k_{in})$, because h is a one-way hash function. Therefore, the security of KDSM is guaranteed.

Now we analyze the performance of KDSM from three aspects. We first consider the *storage* issue. For the secure multicast group g_j, unlike most related work, a member u_i holds only a private secret key k_i and a group key gk_j; other work usually requires more keys. Only the CA has to keep more objects in KDSM; the CA has to keep the private secret keys of all users and the group keys of all secure multicast groups.

For the *computation load*, when the CA distributes a new group key, only a few, even none, encryption/decryption operations are required in KDSM. Two primary operations may be performed by the CA or members: constructing secure filters and extracting new group keys from secure filters. To construct a secure filter, recall that the definition of secure filter sf on gk with respect to $S = \{k_0, k_1, \ldots, k_{t-1}\}$ is $sf = (x - h(k_0))(x - h(k_1))\ldots(x - h(k_{t-1})) + gk$. Transferring it into the form $a_t x^t + a_{t-1}x^{t-1} + \ldots + a_1 x + a_0$ requires $O(t^2)$ multiplications and $O(t^2)$ additions. To extract gk from sf, suppose sf is the polynomial over $\mathbf{GF}(p)$ of degree t, then it requires t multiplications, t additions, and a modular operation for an authorized member to extract the group key from sf by applying Horner's rule [8]. Note that we have to compute the

value of one-way hash functions in KDSM often, which can be done efficiently if the hash functions are well-selected.

Finally, for the *message* consideration, to distribute a new group key in KDSM, the CA only needs to multicast one message (sf, h). This feature is better than most researches.

3.4. Refinement

Note that the degree of a secure filter is proportional to the size of the associated group. If the group grows large, constructing a secure filter requires a great deal of computation. To overcome this drawback, we can partition the group into subgroups, and the size of subgroups is bounded into a small range. Moreover, each subgroup is associated a subgroup key. When the membership of the group is altered, for the subgroups that do not involve the joining/leaving member, the CA first encrypts the new group key using their subgroup keys, and then multicasts the encrypted group key to them. The members of these subgroup can extract the new group key by their subgroup keys.

For the subgroup involving the joining/leaving member, the CA first constructs a secure filter for the subgroup, and then multicasts the secure filter to the members of the subgroup. Though extra keys and encryption/decryption operations are required in the modified scheme, we get a better performance to distribute a new group key if the system is large enough.

4. Conclusion

We proposed a solution KDSM to the key distribution problem in secure multicasting environment. To develop the solution, we suggest a special function called secure filter as the building block of KDSM. The function prevents unauthorized users from getting group keys, and then makes the key distribution process secure and efficient.

The KDSM ensures the system security; and further, it has low storage requirement, efficient objects generating/extracting operations and few message requirement. Note that the secure filters can be dropped directly after the new keys have been distributed.

References

[1] A. Ballardie. Scalable multicast key distribution. RFC 1949, May 1996.

[2] D. Bonch and M. Franklin. An efficient public key traitor tracing scheme. In *Proc. Crypto /99, Lecture Notes in Computer Science*, volume 1666, pages 338–353. Springer-Verlag, 1999.

[3] M. G. C. K. Wong and S. S. Lam. Secure group communication using key graphs. In *Proc. ACM SIGCOMM '98*, pages 68–79.

[4] G. H. Chiou and W. T. Chen. Secure broadcasting using the secure lock. *IEEE Trans. Software. Eng.*, 15:929–934, 1989.

[5] E. J. H. D. M. Wallner and R. C. Agee. Key management for multicast: Issues and architectures. RFC 2627, June 1999.

[6] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) architecture. RFC 2094, July 1997.

[7] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) specification. RFC 2093, July 1997.

[8] D. E. Knuth. *The Art of Computer Programming, Seminumerical Algorithms*. Addison-Wesley, Reading, Mass., 1998.

[9] G. T. M. Steiner and M. Waidner. Cliques: A protocol suit for key agreement in dynamic groups. Technical report, IBM Zurich Res. Lab, Switzerland, Res. Rep. RZ 2984 No. 93030, Dec. 1997.

[10] D. S. N. W. M. Waldvogel, G. Caronni and B. Plattner. The versakey framework: Versatile group key management. *IEEE J. Selected area in Commun.*, 17:1614–1631, Sept. 1999.

[11] S. Mittra. Iolus: A framework for scalable secure multicasting. In *Proc. ACM SIGCOMM '97*, pages 277–288.

[12] R. Molva and A. Pannetrat. Scalable multicast security in dynamic groups. In *Proc. ACM CCS '99*, pages 101–112.

[13] G. I. D. M. M. N. R. Canetti, J. Garay and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proc. INFOCOM '99*, pages 708–716.