A Constant Size Rekeying Message Framework for Secure Multicasting

Chih-Kuang Tseng[1], Kuen-Pin Wu[1], Jen-Chiun Lin[1], Chun-Yen Chou[2], and Feipei Lai*[3]

Abstract

We propose a new framework for key management to construct a secure multicasting environment. When rekeying, only one message needs to be generated, and the message size is a constant. Especially when the group size is large, our framework largely reduces the traffic in the network. In this paper, two solutions are given to carry out this framework, which are closed curve solution (CCS) and perpendicular space solution (PSS), respectively. One is based on geometric approach and the other on linear algebraic approach. Our framework is also compatible to other group communication protocols.

1 Introduction

Multicasting is now considered an attracting and challenging paradigm in the area of communication. It avoids transmitting packets from one sender to each recipient separately to save the network bandwidth. Such applications include pay-per-view systems, where a video server delivers a movie to all subscribers through multicasting; or video conferencing systems, where messages are shown to all remote members. The movie and messages should be only available to authorized subscribers and conference members, respectively. Therefore, secure multicasting becomes an important design issue in the area of communication.

We assume all members in a group share a group key. Messages should be encrypted by the key before they are multicast. On receiving an encrypted message, group members can extract the message using the group key, while others cannot.

For the sake of security, we need a new group key when someone joins or leaves the group. A new member can only access the multicast messages sent after it joins the group, as illustrated in Figure 1. Therefore, the group key has to be changed when someone joins the group. Similarly, the group key should also be changed when someone leaves the group, because the user is no longer authorized to access the multicast messages after leaving the group. Consequently, distributing new group keys to group members will take place often in dynamic groups in which membership changes frequently. Our goal is to have a way to distribute the new group key securely and efficiently.

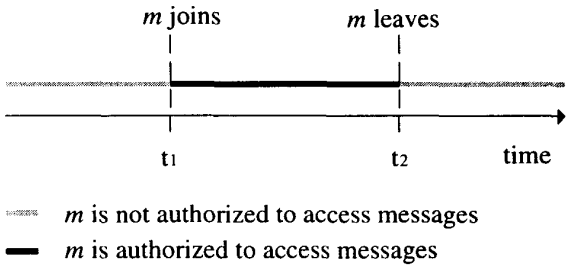

Figure 1. A requirement for a secure multicasting environment

In recent years, some efforts have been done to achieve secure multicasting. In some static key management protocols [1, 2, 3], all members obtain a permanent 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 [5] divides a large dynamic group into subgroups. Each subgroup requires an extra agent to deal with"key translation." Moreover, Iolus needs some intermediate nodes, say GSIs, GSAs, GSCs, to support its framework. The key management schemes for secure multicasting in [6, 7, 9] require each of the N members to store $\log(N) + 1$ keys. As the group size grows, the system requires a large amount of keys and it is difficult to manage so many keys while

*Corresponding author. Tel: +886-2-23914116 Fax: +886-2-23637204.

[1]Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, {*tck, kpw, simon*}*@orchid.ee.ntu.edu.tw*

[2]Department of Mathematical Education, National Hualien Teachers College, Hualien 970, Taiwan, *choucy@sparc2.Nhltc.edu.tw*

[3]Department of Electrical Engineering & Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan, *flai@cc.ee.ntu.edu.tw*

0-7695-1321-2/01 $10.00 © 2001 IEEE

keeping the system secure. Cliques [8] provides a way to distribute group session keys in dynamic groups. However, it does not scale well to a large group. Molva et al. [10] proposed a scalable solution for dynamic groups. Nevertheless, the scheme has to modify the structure of intermediate components of the multicast communication such as routers or proxies. Waldvogel et al. [11] proposed a scalable scheme for dynamic groups such that each member holds fixed number of keys even if the number of members is altered. 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 can have 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. Besides, the methods in [10, 11] may suffer from the collusion attack. However, a very low-overhead operation exists to shrink the key space as the system grows. Wu et al. [13] proposed a scheme to distribute the new group key through a special function called secure filter. While rekeying, it largely reduces the number of multicast messages. However, it needs a great deal of computation when the group size becomes large.

We present a framework using one multicast rekeying message and the message size will not grow with the group size. Especially when the group size is large, our framework largely reduces the traffic in the network. Besides, no intermediate nodes will be employed in our framework. And our framework can be easily applied to the multicasting environments and other group communication protocols without any modification of network hardware. Above all, our framework can resist the collusion attack.

Generally speaking, most solutions to the key management problems in the secure multicasting environment are based on number theory and basic logical operations. In this paper, we propose a new framework for key management in a secure multicasting environment, in which each member is assigned a private key. When a user joins or leaves the group, the group manager will multicast only one message to all members. On receiving it, members can use their own private keys to retrieve the new group key from the message.

Below we give a detailed description of the key distribution problem in secure multicasting environment. We will then define our framework formally. Two concrete solutions are given in section 4, one is based on geometry and the other on linear algebra. To our best knowledge, they are the first one that employs geometry and linear algebra, respectively, to solve these kinds of problems. Section 5 concludes this paper.

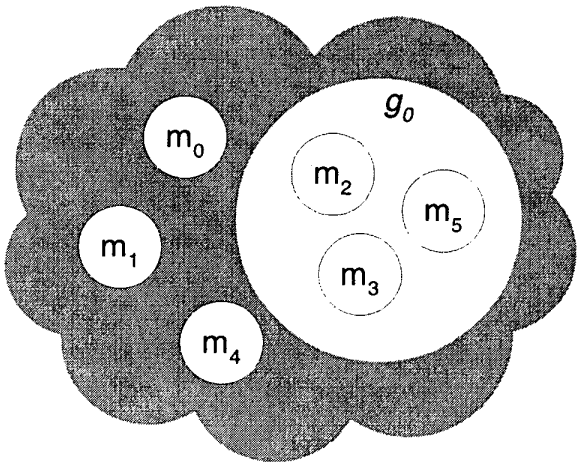

Figure 2. A secure multicasting environment.

2 Key management in secure multicasting

A secure multicasting environment consists of users and secure multicasting groups. Each user m_i has a private secret key k_i to perform secure point-to-point communication. A secure multicasting group consists of more than one user. Each secure multicasting group g_j is associated with a group key Kg_j. Members of g_j are capable of using Kg_j to perform secure multicasting. Note that a user can be a member of many secure multicasting groups.

There is an example illustrated in Figure 2. In the figure the secure multicasting environment has six users m_0, m_1, ..., m_5. A secure multicasting group g_0 with three members m_2, m_3 and m_5 is also defined. Each user m_i has a private secret key k_i and the secure multicasting group g_0 has a group key Kg_0. To perform secure multicasting, m_2, without loss of generality, first encrypts the multicast messages using Kg_0, and then sends them to m_3 and m_5. On receiving the encrypted messages from m_2, m_3 and m_5 use Kg_0 to decrypt them.

Problems arise if the membership of a secure multicasting group is changed. When a user joins a secure multicasting group, it has to acquire the associated group key to further deal with multicast messages. However, it is possible that the user uses the group key to access past messages that were sent before joining the group. This is not allowed since the user is not an authorized recipient of the old messages as they were sent. Similarly, if a member leaves a secure multicasting group, it is no longer authorized to access the multicast messages. It is also possible that the member uses the group key to access the messages that are sent after it leaves the group. Again, this is not allowed since it is no longer a member. This is illustrated in Figure 1.

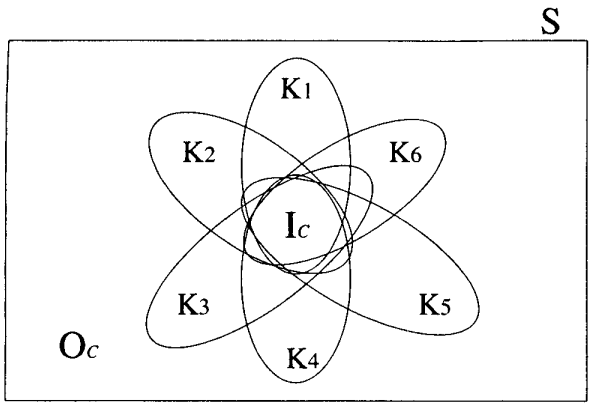

Figure 3. A group configuration $\mathcal{C} = \{K_1, K_2, K_3, K_4, K_5, K_6\}, U_{\mathcal{C}} = \bigcup_{i=1}^{6} K_i$

Therefore, group keys should be changed if some members join or leave the secure multicasting groups. How to distribute new group keys to group members is therefore an important design issue in dynamic groups whose membership changes frequently. Our goal is primarily aimed at finding a way to distribute the new group key securely and efficiently. Moreover, the scheme should be scalable in large dynamic groups.

Our framework to the key distribution problem is presented in the next section. The scheme scales and works well without the help of the trusted third-party [5, 10]. Moreover, the scheme is independent of physical communication environment. We will introduce two solutions of this framework, namely, closed curve solution (CCS) and perpendicular space solution (PSS).

3 A key management framework

The design goal for our framework is that authorized members can obtain the group key efficiently, while the others cannot. We assume that anyone can access the data of the multicasting group only by use of the group key.

In our framework, the private key of a group member is represented as a subset of S, where S is the underlying space. That is, each member in the multicast group is assigned a private key K_i, where $K_i \in 2^S$.[4] A group configuration $\mathcal{C} = \{K_1, K_2, \ldots, K_n\}$ consists of the private keys for each member as illustrated in Figure 3, where $n \in \mathbf{N}$ is the number of group members. The group manager computes $I_{\mathcal{C}}$, $U_{\mathcal{C}}$, and $O_{\mathcal{C}}$ for the generation of the group key:

[4] 2^S is the set of all subsets of S.

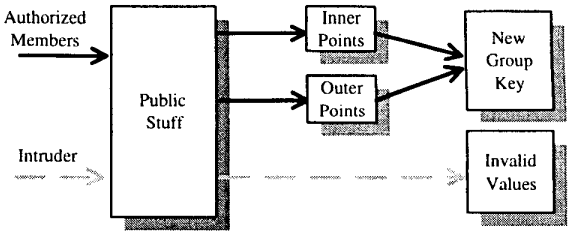

Figure 4. Abstraction of our framework

$$
\begin{aligned}
I_{\mathcal{C}} &= \bigcap_{K_i \in \mathcal{C}} K_i \\
U_{\mathcal{C}} &= \bigcup_{K_i \in \mathcal{C}} K_i \\
O_{\mathcal{C}} &= (U_{\mathcal{C}})^c = S - U_{\mathcal{C}}
\end{aligned}
$$

Thus, we can define the rekeying history $\mathcal{R}$ as a sequence of pairs of configurations $\mathcal{C}_i$ and public sets G_i used to generate group keys: $\mathcal{R} = ((\mathcal{C}_1, G_1), (\mathcal{C}_2, G_2), \ldots, (\mathcal{C}_i, G_i), \ldots)$, where $|G_i| < \infty$. At each time of rekeying, the group manager will multicast only a public set G_i to all members. Here G_i should satisfy the following requirements:

$$
\begin{aligned}
G_i \cap I_{\mathcal{C}_i} &\neq \emptyset \\
G_i \cap (U_{\mathcal{C}_i} - I_{\mathcal{C}_i}) &= \emptyset \\
G_i \cap O_{\mathcal{C}_i} &\neq \emptyset.
\end{aligned}
$$

When a user receives the public set G_i, a point of G_i will be regarded as an *inner point* if it belongs to its private key. All others are *outer points*. Thus, a member can utilize its private key to partition the elements in G_i into two sets: the *inner-point set* and the *outer-point set*. These two sets can then be used to generate the new group key according to different implementations. Without knowing these two sets, any unauthorized user should not know the group key. The security of the framework is based on the difficulty of finding the two sets, which is quite implementation-dependent. The notion of the framework is illustrated in Figure 4, and we will give two solutions in the next section.

Moreover, the size of G_i does not depend on the group size except at reorganizing. The size of our rekeying message will not increase with the growing of the group size. Therefore, our framework saves more bandwidth as the group size becomes larger. The group manager has to decide the size of the public set G_i. Especially, for example, for an environment requiring higher security, it is not appropriate to have a small G_i. Otherwise, it is likely to be vulnerable to brute force attacks.

For the sake of security, special attentions must be paid on the choice of the group keys. For a rekeying history

$\mathcal{R} = ((\mathcal{C}_1, G_1), (\mathcal{C}_2, G_2), \ldots, (\mathcal{C}_i, G_i), \ldots)$, where $(\mathcal{C}_i, G_i)$ corresponds to time period T_i, we must have

$$G_i \cap (U_{\mathcal{C}_j} - I_{\mathcal{C}_j}) \neq \emptyset \quad \text{for all } i \neq j.$$

Otherwise, all legal members at time T_j can access the messages multicast during T_i, while not all of them are legal members at time T_i.

Actually, if the membership of any two time periods differ for more than one member, we further require the following: for any ex-member or any new member with private key K, if the membership is not valid at time T_i, then the inner point set of G_i should not be a subset of K. Otherwise, the ex-member or the new member can access some multicased messages of T_i for which it is not authorized.

It is worth mentioning that our framework resist collusion attacks under proper implementations. For a multicast message at a fixed time T_i with a corresponding $\mathcal{C}_i$, because the private keys held by unauthorized users are not in $\mathcal{C}_i$ and the choice of G_i is based on $\mathcal{C}_i$, they cannot collude to get the group key through the collection of their private keys.

4 Solutions

We now show two concrete solutions to illustrate the framework.

4.1 Closed Curve Solution (CCS)

In the solution, each private key is represented as a closed curve on a fixed underlying plane. The group manager holds all closed curves.

Let $\mathcal{C}$ denote the set of private closed curves held by all members in this group. Let I denote the area bounded by all the closed curves c's in $\mathcal{C}$. We require that I cannot be an empty set. Let U denote the area of the union of all areas by all the closed curves c's in $\mathcal{C}$. Let O denote the complement of U on the plane.

Thus, for a multicasting group of n members,

$$\mathcal{C} = \{c_1, c_2, \ldots, c_n\},$$

where c_i is a closed curve, $1 \leq i \leq n$, such that

$$I = c_1 \cap c_2 \cap \cdots \cap c_n \neq \emptyset.^5$$

When rekeying, the group manager chooses a public set of points $G = \{p_1, p_2, \ldots, p_{\alpha+\beta}\}$ such that there are α points in I and β points in O. The group manager then sends G to all the group members.

When an authorized member m_i with private key c_i receives G, m_i can separate the points in G into two sets by the following rule. If a point is inside c_i, it will be put into the *inner-point set* P_I. Otherwise it will be put into the *outer-point set* P_O. The user then uses the results to generate the correct group key.

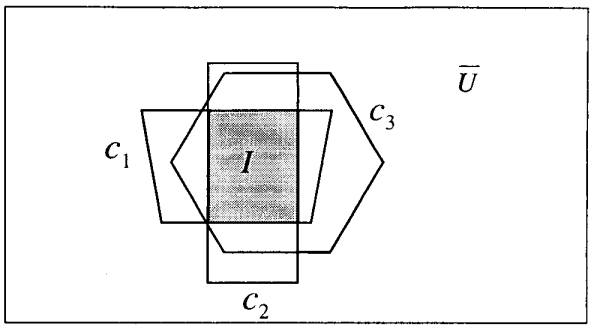

Figure 5. An example of CCS with its $\mathcal{C} = \{c_1, c_2, c_3\}$

The security of this solution is based on the difficulty of separating the points in G into correct P_I and P_O. There are several ways to ensure this. First of all, the number of inner points can be randomly decided and all the points are equally distributed. Secondly, the intersection I may be disconnected and have many components, which means that inner points and outer points can be mixed on the plane. Thus we can have the inner points not clustering together. This makes brutal attacks not feasible. If an intruder u tries to find the Kg in a brute-force way, u needs $O(2^{\alpha+\beta})$ times of trials. For example, if G has 100 points, an intruder needs $O(2^{100})$ times of trials.

Now suppose that a user m_j with private key c_j leaves the group, the group manager will choose some points located inside c_j but not inside any other c_i for $i \neq j$. Hence these point are in the new O and thus are outer points. The system manager will also choose some points outside c_j but inside any other c_i for $i \neq j$. Hence these points are in the new I and thus are inner points. This makes m_j not capable of using c_j to distinguish the correct inner points and outer points.

On the other hand, suppose that m_j joins the group. The system manager assigns a private key c_j to m_j with the following consideration: c_j can not be used to correctly separate the inner points and outer points for any previous public set G.

We now give formal procedures of CCS as follows.

Procedure CCSJOIN(m_i) for **Joining Operation**
begin
m_i is assigned a closed curve c_i;
$\mathcal{C} = \mathcal{C} \cup \{c_i\}$;
$I = I \cap c_i$;

[5] For the sake of brevity, we also use c to denote the region it bounds. For example, $(c_1 \cap c_2)$ means the area intersected by c_1 and c_2.

$U = U \cup c_i$;
$O = U^c$;
select $\alpha + \beta$ points to form a point set G, such that
$|G \cap I| = \alpha$ and $|G \cap O| = \beta$;
send G to all group members;
end

Procedure CCSLEAVE(m_j) for **Leaving Operation**
begin
$\mathcal{C} = \mathcal{C} \setminus \{c_j\}$;
$I = \bigcap_{c_k \in \mathcal{C}} c_k$;
$U = \bigcup_{c_k \in \mathcal{C}} c_k$;
$O = U^c$;
select $\alpha + \beta$ points to form a point set G, such that
$|G \cap I| = \alpha$ and $|G \cap O| = \beta$;
send G to all group members;
end

Note that we do not specify a way to construct group keys using the inner points and outer points. There are many possible approaches, such as XOR all inner points, or put all inner points into an one-way function.

4.2 Perpendicular Space Solution (PSS)

In this solution, each private key is represented as a $k \times 1$ vector $\vec{t}$ over $\mathbf{Z}_p^k$, where p is a large prime and k is an integer. The dimension of its corresponding perpendicular space[6] $t^\perp$ over $\mathbf{Z}_p^k$ is $k - 1$.

Note that *a vector $\vec{v} \in t^\perp$ if and only if $\vec{v}$ is orthogonal to $\vec{t}$.* This means that that *the inner product of $\vec{v}$ with $\vec{t}$ is zero.* We can therefore use this property to separate the vectors.

Let $\mathcal{C}$ denote the set of the perpendicular spaces corresponding to the vectors held by each member. Let I denote the intersection space by all the perpendicular spaces of the members. Let U denote the union by all the perpendicular spaces of the members. Let O denote the complement of U.

Thus, for a group of n members,

$$\mathcal{C} = \{t_1^\perp, t_2^\perp, \ldots, t_n^\perp\},$$

where $t_i^\perp$ is the perpendicular space with respect to $\vec{t_i}$ such that

$$I = t_1^\perp \cap t_2^\perp \cap \cdots \cap t_n^\perp \neq \emptyset.$$

When rekeying, the group manager finds a set of vectors $G=\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_{\alpha+\beta}\}$ such that there are α vectors in I and β vectors in O. The group manager then sends G to all the group members.

[6] A perpendicular space $t^\perp$ is a collection of all the vectors perpendicular to $\vec{t}$. Since $t^\perp$ comes with $\vec{t}$, we can hence take $t^\perp$ as user's private key abstractly for consistency with our framework.

When an authorized member m_i receives this public set G, it uses its vector $\vec{t_i}$ to perform the inner product of $\vec{t_i}$ with each vector in G. If the result equals zero, the vector then belongs to $t^\perp$. m_i then put this kind of vectors into the *inner-vector set* while other vectors are put into the *outer-vector set.* The user then uses the results to generate the correct group key.

The analysis of security for the PSS is the same as for the CCS. We will now only give formal procedures for the PSS here.

Procedure for PSSJOIN(m_i) for **Joining Operation**
begin
m_i is assigned a $k \times 1$ vector $\vec{t_i}$ over Z_p^k, where
p is a large prime; k is an integer;
$\mathcal{C} = \mathcal{C} \cup \{t_i^\perp\}$;
$I = I \cap t_i^\perp$;
$U = U \cup t_i^\perp$;
$O = U^c$;
select $\alpha + \beta$ vectors to form a vector set G, such that
$|G \cap I| = \alpha$ and $|G \cap O| = \beta$;
send G to all group members;
end

Procedures PSSLEAVE(m_j) for **Leaving Operation**
begin
$\mathcal{C} = \mathcal{C} \setminus \{t_j^\perp\}$;
$I = \bigcap_{t_i^\perp \in \mathcal{C}} t_i^\perp$;
$U = \bigcup_{t_i^\perp \in \mathcal{C}} t_i^\perp$;
$O = U^c$;
select $\alpha + \beta$ vectors to form a vector set G, such that
$|G \cap I| = \alpha$ and $|G \cap O| = \beta$;
send G to all group members;
end

5 Conclusions

In this paper we have proposed a novel framework for key management in the secure multicasting environments. When rekeying, the group manager will multicast a public set to all the members. On receiving the public set, a member can use its private key to separate the elements in the public set and use the results to generate the new group key. We also give two solutions of the framework, which have different properties, but rely on the same basic principle. One of which is closed curve solution (CCS), and the other is perpendicular space solution (PSS).

One benefit of our framework is that we need only one multicast rekeying message, say the public set, and hence the message size can be a constant. Furthermore, compared with the related work, our framework applies well to other

group communication protocols because it does not need a third trusted party (such as proxy, router, or some intermediate nodes). Above all, this methodology shown in our framework is useful and meaningful. We give solutions using the concept of geometry and linear algebra. By this, we believe it is worth trying to find other interesting mathematical properties for solving the secure multicasting problem.

6 Acknowledgement

We would like to thank Chun-Hsiung Hsia in Indiana University for his discussion on Perpendicular Space Solution and the anonymous reviewers for the valuable feedback in improving our paper.

References

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

[2] H. Harney, and C. Muckenhirn, "Group key management protocol (GKMP) specification," *RFC* 2093, July, 1997.

[3] H. Harney, and C. Muckenhirn, "Group key management protocol (GKMP) architecture," *RFC* 2094, July, 1997.

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

[5] S. Mittra, "Iolus: A framework for scalable secure multicasting," *Proc. ACM SIGCOMM '97*, pp. 277-288.

[6] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *Proc. ACM SIGCOMM '98*, pp. 68-79.

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

[8] M. Steiner, G. Tsudik, and M. Waidner, "Cliques: A protocol suite for key agreement in dynamic groups," *IBM Zurich Res. Lab, Switzerland, Res. Rep.* RZ 2984 No. 93030, Dec. 1997.

[9] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas "Multicast security: A taxonomy and some efficient constructions." *Proc. INFOCOM '99*, pp. 708-716.

[10] R. Molva, and A. Pannetrat, "Scalable multicast security in dynamic groups," *Proc. ACM CCS '99*, pp. 101-112.

[11] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: Versatile group key management," *IEEE J. Selected areas in communication*, vol. 17, pp. 1614-1631, Sept. 1999.

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

[13] K.P. Wu, S.J. Ruan, F. Lai, C.K. Tseng, "On Key Distribution in Secure Multicasting," *Proc. of 25th IEEE Conf. Local Computer Networks*, pp. 208-212, Nov. 2000.