International Journal of Networked and Distributed Computing, Vol. 3, No. 1 (January 2015), 21-30

An Efficient Structure for LKH Key Tree on Secure Multicast Communications

Naoshi Sakamoto

Department of Information and Communication Engineering, Tokyo Denki University,
5 Senju-Asahi-cho,
Adachi-ku, Tokyo, 120-8551, Japan
E-mail: sakamoto@c.dendai.ac.jp

Abstract

In order to communicate in cipher over IP multicast, each of joining and leaving participants causes renewing keys. Moreover, the number of renewed keys depends on the key management system. LKH, one of the key management systems, uses a tree structure to manage keys to share with participants. Every node of the tree is given a key, and each leaf of the tree is corresponding to a participant. If all members are handled equally, by using a balanced binary tree, the average number of renewed keys per join and leave is estimated at $\lceil \log_2 n \rceil$, where n denotes the number of participants. In this study, we introduce a scenario that the key management system can distinguish between inconstant members and stable members, instead of handling members equally. Under this scenario, our system improves the number of renewing keys efficiently by considering another tree structure against the balanced binary tree structure.

Keywords: IP Multicast, IPsec, GSAKMP, LKH, Pareto Principal

1. Introduction

Some video conferences using multicast communication require avoiding a wire tapping. That is, they must communicate in safe. IPsec is a framework of secure communications on IP. In order to manage keys for secure communications on IP, IKE for unicast and GSAKMP for multicast are used.

For GSAKMP, LKH is proposed as a method to manage keys[1]. In LKH, the key management is based on a key tree, where each of all its nodes corresponds to a key, and each of all its leaves is corresponded to a participant. It is reported that the balanced binary tree is efficient for a key tree when every member would be handled equally[2].

Meanwhile, it is reported that for some services on the Internet, the fluctuation of the frequency in use strongly depends on days of the week[3]. According to such investigations, we are expecting that the good user model will be found in future so that this will be able to estimate the load of the service well. In this study, we propose an efficient method to manage key for multicast communication in cipher by assuming that the administrator can obtain the information of users' behavior. While the users are handled equally in the former studies, we take special care with the users that frequently cause renewing keys. Then, we show the condition that our method is more efficient than former ones.

In this paper, in section 2, we explain the secure multicast communication and the related studies. Then, we show and analyze our method in section 3. Finally, in section 4, we conclude our study.

Published by Atlantis Press

2. Secure communication and Key management

2.1. Basic concept

We assume that the base of logarithms is two.

The system of multicast communication in cipher requires not only the same join and leave management as normal multicast communication, but also the key management. In this paper, we assume that this can manage the users such that:

- the users are registered in advance,
- once a user declares her join, she is enabled to receive data (we call a joining user a participant),
- then, after she declares her leave, she can receive nothing anymore.

Then, this yields that this system must use the keys where

- those who have already left can not decrypt, and
- whoever joins can decrypt.

Moreover, the key to decrypt data should be shared by the all participants. Thus, the keys must be renewed in safe for every joins and leaves. In order to manage joins and leaves and rekey, we assume a key server is installed in the system.

Moreover, we assume that the key server creates log files so that we can obtain the statistical information with respect to users' behavior.

2.2. Framework of secure multicast communication over the Internet

Security Architecture for Internet Protocol (IPsec RFC4301) basically employs a shared key cryptosystem. For safety's sake, the key management protocol renews keys whenever a period of time elapses, and the group of participants is renewed.

Group Secure Association Key Management Protocol (GSAKMP RFC4535) is a key management protocol for secure multicast communication. This organizes and manages a group to be protected by secure communication. This assumes the key server to manage participants and keys. GSAKMP uses the following two types of key. A traffic encryption key (TEK) denotes the encryption key to protect data communication channels. On the other hand, a key encryption key (KEK) denotes the encryption key to encrypt TEKs in order to distribute TEKs in secure. GSAKMP can apply LKH to rekey efficiently.

2.3. LKH

LKH(Logical Key Hierarchy [1]) is a method to distribute keys efficiently via multicast for GSAKMP. In LKH, the key server manages keys by using a tree structure(see Figure 1.)

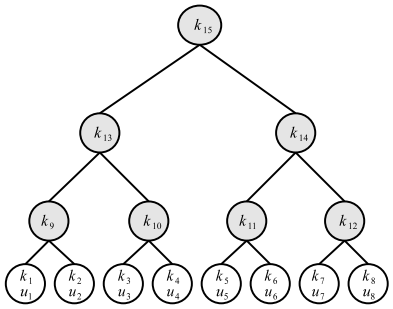

Fig. 1. Key tree of LKH

We call such tree in Figure 1 a key tree. Firstly, the key server generates a tree so that each leaf is corresponding to a participant each. Secondly, it generates KEKs and corresponds each KEK to a node of the tree each. These KEKs are different from each other. Finally, for each participant p, the key server distributes a key set that consists of any KEKs over the path from the root to the leaf corresponding to p in the key tree.

For instance, in Figure 1, u_1, the most left participant, has KEK k_1, k_9, k_{13}, and k_{15}. Note that though the key tree is a binary tree in Figure 1, any ordinal trees are also available for a key tree. Nevertheless, we assume that any key tree are all binary trees in this paper.

Now, we show how the key server rekeys for the key tree in Figure 1 as follows. Firstly, we explain the action of the periodic rekeying that is independent of a join and a leave. The periodic rekeying does not need to exclude the particular members.

Thus, the key server encrypts a new TEK with the root key k_{15}, and distributes it to all participants via multicast. Since the all participants share k_{15}, they obtain the new TEK. This requires only one new TEK.

Secondly, we explain the rekey process for a join and a leave. When a participant leaves, the key server must renew not only the TEKs, but also the KEKs for all of the left participants. Suppose that participant u_1 leaves. Then, k_1 must be discard, and k_9, k_{13} and k_{15} must be renewed since u_1 has owned them. Let new KEKs with respect to the remaining k_9, k_{13} and k_{15} be k'_9, k'_{13} and k'_{15} respectively. The key server distributes these new keys, k'_9, k'_{13} and k'_{15}, to the participants by encrypting with KEKs. Note that the key server uses multicast communication so that it reduces the number of communications as follows:

(i) For u_5, u_6, u_7 and u_8, it is necessary to renew only k_{15}. Since they share k_{14}, the key server encrypts the new key k'_{15} with k_{14}, and distributes it via multicast.

(ii) For u_3 and u_4, it is necessary to renew KEK k_{15} and k_{13}. Since they share k_{10}, the key server encrypts the key set that consists of the new key k'_{15} and k'_{13} with k_{10}, and distributes it via multicast.

(iii) Finally, since u_2 has KEK k_2, the key set that consists of the new key k'_{15}, k'_{13} and k'_9 encrypted with k_2, are delivered to u_2 via unicast.

On the other hand, when a participant joins, all keys on the path from the root to the node corresponding to the participant are renewed and distributed by the same way for a leave.

Therefore, for every rekey, the sum of the number of either created or discarded KEKs and renewing KEKs is equal to the depth of the position of the node for the participant. When the key tree is balanced, the number of renewed keys per a rekey is at most $\lceil \log n \rceil^2$. Moreover, we can estimate the number of distributed keys as follows. Let the depth of the root be one. For a participant such that in her own KEKs, KEKs in depth up to d are being renewed, d KEKs are sent to her by encrypted with the unrenewing KEK in depth $d+1$ that she owns. Thus, since KEKs on the path from the root to a node in depth up to $\lceil \log n \rceil - 1$ are entirely renewed, the number of distributed keys is given by (1).

$$1 + 2 + \ldots + \lceil \log n \rceil - 1 = \sum_{i=1}^{\lceil \log n \rceil - 1} i = \frac{\lceil \log n \rceil^2 - \lceil \log n \rceil}{2}. \quad (1)$$

2.4. Related works

On pay TV, it is ordinary that a contract happens to be concluded and canceled not anytime, but for each program. Therefore, we can assume that we administer the process for a join and a leave only at some intervals. Wong, Gouda and Lam provided Batch LKH, a method of key administration for this assumption[4]. That is, in Batch LKH, the process for a join and a leave within a period is handled all at once. Since the root KEK key is renewed every rekey, rekeying for more than two of joins and leaves all at once in a period reduces the number of renewed keys from rekeying every a join and a leave. While our method also reduces the number of times of renewing keys, our method and Batch LKH can be used together. Moreover, it is expected that using them together results more efficient.

Now, according to the former subsection, in LKH, the number of keys to renew per rekey depends on the depth of a key tree. Thus, a balanced tree is considered to be most efficient for the form of the key tree since the depth of the balanced tree is shallowest. However, joining and leaving may make the key tree unbalance gradually. Pegueroles and Rico-Novella provided the Balanced Batch LKH[5]. This enables a key tree to keep balanced by reconstructing the form of the key tree every rekeying of Batch LKH. Since this method improves LKH independently of our method, it is expected that this method can be applied to our method effectively.

On the other hand, the broadcast encryption[6] is connected with multicast communication in cipher. In this method, a sender transmits a bunch of keys to each participant in advance so that sender enables

some group of participants to communicate. LKH is also one of the broadcast encryption systems. Abdalla, Shavitt and Wool investigates about the number of key transmissions for the broadcast encryption [7]. They estimated lower bound of the number of key transmissions theoretically, and proposed an algorithm for several type of relationships between participants. Particularly, in Section 6.B, they evaluated it for tree structure. This is similar to ours. However, while they show the upper-bound by using simulation, we show the upper-bound theoretically.

3. Proposal method

In this section, we propose our method. Despite other related works, our method assumes that the key server can obtain some special information of users in advance.

3.1. Basic Concepts

When the key tree is managed in the form of a balanced binary tree, the number of renewed keys per each of a join and a leave is $\lceil \log M \rceil$, where M stands for the average number of participants. This is considered to be the most efficient, when each user joins and leaves independently with the same probability.

Now, consider an unbalanced key tree like Figure 2.

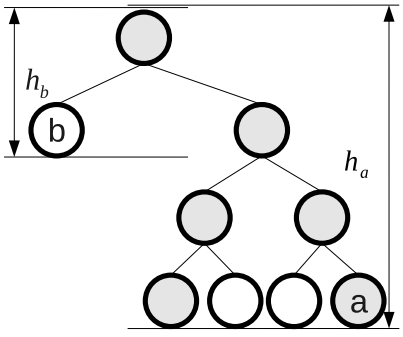

Fig. 2. Unbalanced key tree

In the tree in this figure, when the participant a leaves, the number of renewed keys is equal to its depth h_a. This is greater than the number when the key tree is managed in the form of a balanced tree. On the other hand, when the participant b leaves, the number of renewed keys is equal to its depth h_b. This is less than the number when the key tree is managed in the form of a balanced tree. If the key server could know that a certain participant leaves earlier than the other participants, it could reduce the number of renewed keys by placing her node to the shallower part of the key tree.

From this idea, we propose more efficient structure of the key tree then a balanced tree, when the key server can know the average number of participants and the join and leave rate.

3.2. How to construct a key tree

Our method requires to grasp the trend of users. Since the trend might be affected by the individual characters of the users and the contents of the service, it may not necessarily possible to predict the users' behavior by modeling the trend.

However, Pareto principle usually holds. This is a rule of thumb; e.g., "20% of participants occupy 80% of resources".

We assume that we can classify users into more than two groups where one group contains users that join and leave frequently and another group contains users that join and leave rarely. If Pareto principle holds with respect to the users, this assumption can be realized by analyzing the logs of the communications of the users.

On the other hand, in administrating the multicast groups, users are often divided into plural groups in advance for the network topology. Since for each group a key tree is prepared independently, our method can apply to each key tree separately.

Now, let G_1, G_2,... be user subgroups that are obtained. For each subgroup G_1, G_2, ..., let M_1, M_2, ... be the average number of participants in the subgroup respectively. Moreover, let λ_1, λ_2, ... be the join and leave rate of the subgroup respectively.

Note that both M_i and λ_i have additivity. That is, for groups $G_0 = G_1 \cup G_2$, $G_1 \cap G_2 = \emptyset$, we have the property as (2).

$$\begin{aligned} M_0 &= M_1 + M_2, \text{ and} \\ \lambda_0 &= \lambda_1 + \lambda_2. \end{aligned} \tag{2}$$

After this, we denote $G=(M,\lambda)$ when we pay attention to only M and λ in the discussion about group G.

3.3. Division into two groups

At first, we discuss that we divide user group into two. For group $G=(M,\lambda)$, we discuss the condition how G is divided into $G_1=(M_1,\lambda_1)$ and $G_2=(M_2=M-M_1,\lambda_2=\lambda-\lambda_1)$ in order to reduce the number of rekeys.

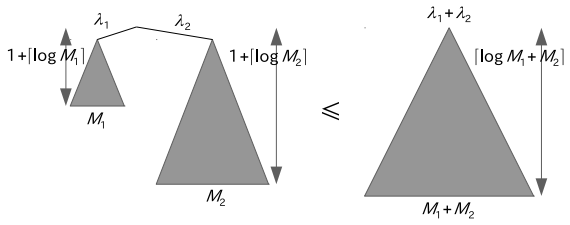

Fig. 3. Separability

Definition 1. $G=(M,\lambda)$ is separable into $G_1=(M_1,\lambda_1)$ and $G_2=(M_2=M-M_1,\lambda_2=\lambda-\lambda_1)$ iff $\lambda_1(1+\lceil\log M_1\rceil)+(\lambda-\lambda_1)(1+\lceil\log M-M_1\rceil)\leqslant\lambda\lceil\log M\rceil$.

Theorem 1. *For $G=(M,\lambda)$, let $M=2^{m-1}+r$ and $0\leqslant r<2^{m-1}$, where m is a positive integer and r is a positive real number. G is separable into $G_1=(M_1,\lambda_1)$ and $G_2=(M-M_1,\lambda-\lambda_1)$ if condition of G_1 given by (3) holds:*

$$
\begin{aligned}
G_1 &\in S_1\cup S_2\cup S_3, \textit{where}\\
S_1 &= \{(M_1,\lambda_1)\mid 0<M_1\leqslant\min\{r,2^{m-2}\},\\
&\quad \wedge\,\lambda\frac{1}{m-\lceil\log M_1\rceil}\leqslant\lambda_1\leqslant 1\},\\
S_2 &= \{(M_1,\lambda_1)\mid r\leqslant M_1\leqslant 2^{m-1}\ \wedge\ 0\leqslant\lambda_1\leqslant 1\},\\
&\quad \textit{and}\\
S_3 &= \{(M_1,\lambda_1)\mid\\
&\quad \max\{2^{m-1},2^{m-2}+r\}<M_1\leqslant M\\
&\quad \wedge\,0\leqslant\lambda_1\leqslant\lambda(1-\frac{1}{m-\lceil\log M-M_1\rceil})\}. \quad (3)
\end{aligned}
$$

We show the area of these sets with Figure 4 and 5. In the case of $0\leqslant r<2^{m-2}$, S_1, S_2 and S_3 are separate from each other as in Figure 4. On the other hand, in the case of $2^{m-2}\leqslant r<2^{m-1}$, S_2 is connected to both S_1 and S_3 as in Figure 5.

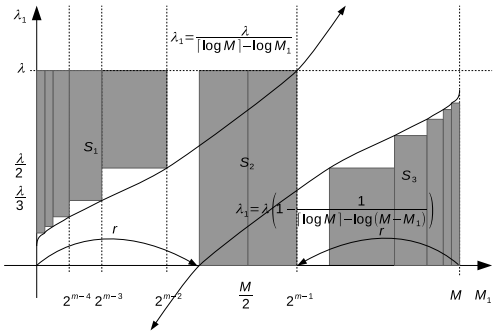

Fig. 4. In the case of $0\leqslant r<2^{m-2}$

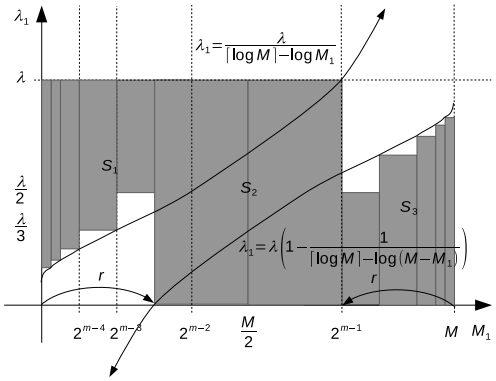

Fig. 5. In the case of $2^{m-2}\leqslant r<2^{m-1}$

Proof. We show the condition of M_1 and λ_1 bu considering separately the cases with respect to M_1. Assume that $M_1<M$ and $\lambda_1<\lambda$.

G is separable into G_1 and G_2, iff inequality (4) holds.

$$
\begin{aligned}
\lambda_1(1+\lceil\log M_1\rceil) &+ (\lambda-\lambda_1)(1+\lceil\log M-M_1\rceil)\\
&\leqslant \lambda\lceil\log M\rceil. \quad (4)
\end{aligned}
$$

(i) When $0<M_1<r$, By $\lceil\log M\rceil=\lceil\log M-M_1\rceil=m$, inequality (4) can be transformed to (5).

$$
\begin{aligned}
&\lambda_1(1+\lceil\log M_1\rceil)+(\lambda-\lambda_1)(1+m)\\
&\qquad\leqslant\lambda m. \quad (5)
\end{aligned}
$$

Thus, we obtain the condition of (6).

$$\lambda_1 \geqslant \lambda \frac{1}{m - \lceil \log M_1 \rceil}. \quad (6)$$

Note that from $\lambda_1 < \lambda$, $\lceil \log M \rceil - \lceil \log M_1 \rceil > 1$ must hold. Therefore, condition $M_1 \leqslant 2^{m-2}$ must also hold.

(ii) When $r \leqslant M_1 \leqslant M/2$, by applying $\lceil \log M - M_1 \rceil = m-1$, inequality (4) can be transformed to (7).

$$\begin{aligned} \lambda_1(1+\lceil \log M_1 \rceil) & \\ +(\lambda-\lambda_1)(1+m-1) &\leqslant \lambda m \\ \lambda_1(1+\lceil \log M_1 \rceil - m) &\leqslant 0. \end{aligned} \quad (7)$$

From $\lambda_1 > 0$, we have $m - \lceil \log M_1 \rceil - 1 \geqslant 0$. We also have $M_1 \leqslant 2^{m-1}$. However, since $M_1 < M/2 = 2^{m-2} + r/2 < 2^{m-1}$, inequality (4) always holds for any λ_1.

(iii) When $M/2 < M_1 \leqslant 2^{m-1}$, then $\lceil \log M_1 \rceil = m-1$, by applying $M/2 > M - M_1 \geqslant r$. Then, inequality (4) can be transformed to (8).

$$\begin{aligned} \lambda_1(1+m-1) & \\ +(\lambda-\lambda_1)(1+\lceil \log M - M_1 \rceil) &\leqslant \lambda m \\ (\lambda-\lambda_1)(m-\lceil \log M - M_1 \rceil - 1) &\geqslant 0. \end{aligned} \quad (8)$$

From $\lambda > \lambda_1$, we have $m - \lceil \log M - M_1 \rceil - 1 \geqslant 0$. Then this can be transformed to $M - M_1 \leqslant 2^{m-1}$. However, since $M - M_1 < M/2 = 2^{m-2} + r/2 \leqslant 2^{m-1}$, inequality (4) always holds for any λ_1.

(iv) When $2^{m-1} < M_1 < M_2$, by applying $\lceil \log M_1 \rceil = m$, inequality (4) can be transformed to (9).

$$\begin{aligned} \lambda_1(1+m) & \\ +(\lambda-\lambda_1)(1+\lceil \log M - M_1 \rceil) &\leqslant \lambda m \\ \lambda_1 \leqslant \lambda \left(1 - \frac{1}{m - \lceil \log M - M_1 \rceil}\right). & \end{aligned} \quad (9)$$

However, $m - \lceil \log M - M_1 \rceil > 1$ must hold. Since this can be transformed to $2^{m-2} \geqslant M - M_1$, we have the condition that $M_1 \geqslant 2^{m-2} + r$.

□

While this theorem gives the condition of parameters to divide a group, this can also apply to decide for given two groups, which management is better, ether managing them separately or managing them with mixing into one. Moreover, we can seek a suitable set of parameters for this theorem. Then, we can try to divide the user group into two to adapt the set of parameters. This problem can be formalized that:

- Instance: for given a group that consists of elements with two kind of parameters and two constraints,
- Question: divide a group into two in order that for each kind, the sum of parameters in one subgroup satisfies the constraint respectively

However, this problem is called the Bounded Knapsack Problem, which is NP-hard. Though some algorithms for this have been proposed[8], they are not polynomial time computable.

Example 1. Now, when the users are obeyed Pareto principal, we compute how much our method can reduce the number of rekeys. That is, when $G = (M, \lambda)$ is separable into $G_1 = (0.2M, 0.8\lambda)$ and $G_2 = (0.8M, 0.2\lambda)$, we evaluate how much our method reduces the average number of rekeys by function (10).

$$\begin{aligned} f(M) &= \lambda \lceil \log M \rceil \\ &\quad -(0.8\lambda(1+\lceil \log 0.2M \rceil) \\ &\quad +0.2\lambda(1+\lceil \log 0.8M \rceil)). \end{aligned} \quad (10)$$

We assume $M > 4$.

(i) When $2^m < M \leqslant 5 \cdot 2^{m-2}$, by applying $\lceil \log M \rceil = m+1$, $\lceil \log 0.2M \rceil = m-2$, and $\lceil \log 0.8M \rceil = m$, we have (11).

$$\begin{aligned} f(M) &= \lambda(m+1) - (0.8\lambda(1+(m-2)) \\ &\quad +0.2\lambda(1+m)) \\ &= 1.6\lambda. \end{aligned} \quad (11)$$

(ii) When $5 \cdot 2^{m-2} < M \leqslant 2^{m+1}$, by applying $\lceil \log M \rceil = m+1$, $\lceil \log 0.2M \rceil = m-1$, and

$\lceil \log 0.8M \rceil = m+1$, we have (12).

$$\begin{aligned} f(M) &= \lambda(m+1) - (0.8\lambda(1+(m-1)) \\ &\quad +0.2\lambda(1+(m+1)) \\ &= 0.6\lambda. \end{aligned} \tag{12}$$

Thus, by the value of M, our method reduces the number of rekeys by 0.6λ or 1.6λ.

3.4. Management of the key tree with plural subgroups

Next, we propose the method to construct the key tree with connecting balanced binary trees corresponding to the plural groups G_1, G_2, ..., G_k each other.

As we saw in the former section, it is complicated to decide which method reduces the number of rekeys more, ether managing two subgroups separately, or managing them together by mixing them. Moreover, it should be naturally more complicated to decide which method reduces the number of rekeys more, ether managing plural subgroups separately, or managing them by selecting some set of subgroups and mixing subgroups in the set, since we must consider all combinations of selection from the subgroups.

However, if we must manage the plural subgroups by each separate balanced binary tree, we can have the optimal method by constructing the key tree according to Huffman coding[9] as follows.

Theorem 2. *When G_1, G_2, ..., and G_k must be managed by each separate balanced binary tree, it is optimal to apply the algorithm of Huffman coding for λ_1, ..., λ_k to construct the key tree.*

Proof. We assume that each group G_i is managed the corresponding balanced binary tree of depth $\lceil \log M_i \rceil$, for $i = 1$, ..., k. Let p_i denote the depth from the root node of the key tree to the root node of the balanced binary tree corresponding to G_i. Then, the average number of rekeys can be estimated by (13).

$$\sum_{i=1}^{k} \lambda_i (p_i + \lceil \log M_i \rceil) = \sum_{i=1}^{k} \lambda_A P_i + \sum_{i=1}^{k} \lambda_i \lceil \log M_i \rceil. \tag{13}$$

In this formula, the method to construct the key tree only affects the value of the term $\sum_{i=1}^{k} \lambda_i p_i$. Since the algorithm of Huffman coding can construct a binary tree to minimize this, we can apply this algorithm for λ_1, ..., λ_k. □

Now, we discuss the validity for managing more than one group separately or managing them with mixing some subgroups. Then, we define "valid-separation" as the transformed notation of "separable".

Definition 2. G_2 is a valid-separation for G_1 iff $G_1 \cup G_2$ is separable into G_1 and G_2.

G_1 and G_2 are valid-separation iff $G_1 \cup G_2$ is separable into G_1 and G_2.

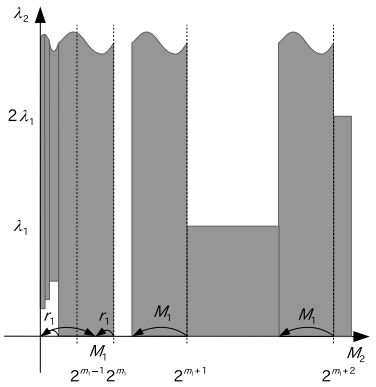

Fig. 6. Valid-separation

Theorem 3. *$G_2 = (M_2, \lambda_2)$ is a valid-separation for $G_1 = (M_1, \lambda_1)$, if one of the following conditions holds where $M_1 = 2^{m_1} - r_1$ for a positive integer $m_1 \geqslant 1$, a positive real number $r_1 \geqslant 0$, and a positive integer $l \geqslant 1$ (Figure 6):*

(i) When $0 < M_2 \leqslant r_1$, then

$$\lambda_2 \geqslant \frac{\lambda_1}{m - 1 - \lceil \log M_2 \rceil};$$

(ii) When $r_1 < M_2 \leqslant 2^{m_1} + r_1 = 2^{m_1+1} - M_1$, then $r_2 < M_2 \leqslant 2^{m_1}$;

(iii) When $2^{m_1+l} - M_1 < M_2 \leqslant 2^{m_1+l}$, then any G_2 is valid;

(iv) When $2^{m_1+l} < M_2 \leqslant 2^{m_1+l} - M_1$, then $\lambda_2 \leqslant l\lambda_1$.

Since the proof of this theorem is similar to the proof of Theorem 1, we omit the proof.

By using this theorem, we can evaluate whether one combination of groups is a valid-separation for another combination of groups. However, it is still complicated to decide whether more than two groups are valid-separation. That is, it is complicated to decide whether we can not reduce the average rate of rekeys anymore whatever we mix some combination of groups. In the next section, we discuss valid-separateness for three groups.

3.5. Valid-separateness for three groups

We yield the condition as valid-separateness for three groups so that the structure of the key tree in figure 7 is optimal.

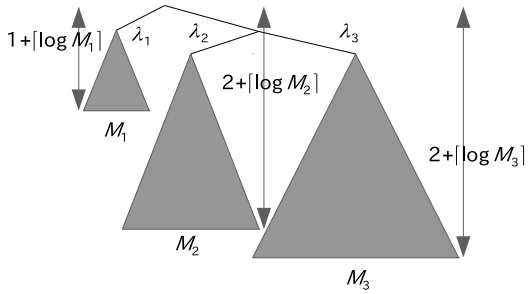

Fig. 7. three groups

Dividing $G = (M, \lambda)$ into three groups G_1, G_2, G_3 is suitable, if $\lambda_1(1+\lceil \log M_1 \rceil) + \lambda_2(2+\lceil \log M_2 \rceil) + \lambda_3(2+\lceil \log M_3 \rceil)$ is smaller than each of the following formulas:

$$(\lambda_1+\lambda_2+\lambda_3)\lceil \log M_1+M_2+M_3 \rceil, \quad (14)$$

$$\lambda_1(1+\lceil \log M_1 \rceil) + (\lambda_2+\lambda_3)(1+\lceil \log M_2+M_3 \rceil), \quad (15)$$

$$\lambda_2(1+\lceil \log M_2 \rceil) + (\lambda_1+\lambda_3)(1+\lceil \log M_1+M_3 \rceil), \text{ and} \quad (16)$$

$$\lambda_3(1+\lceil \log M_3 \rceil) + (\lambda_1+\lambda_2)(1+\lceil \log M_1+M_2 \rceil). \quad (17)$$

(i) On the average number of rekeys $\leqslant$ formula (15) $\leqslant$ formula (14), it is not trivial that formula (15) $\leqslant$ formula (14). Thus, this inequality is a sufficient condition.

In order to satisfy this inequality, we may show that G_1 is a valid-separation for $G_2 \cup G_3$, G_2 is a valid-separation for G_3, and G_1 is a valid-separation for both G_2 and G_3. Then, once G_1 is selected, we select G_2 and G_3 to satisfy both the condition whether G_2 and G_3 are valid-separation, and the condition whether both G_2 and G_3 are valid-separations for G_1. We can verify the valid-separateness by verifying whether the selected G_2 and G_3 are contained in the area yielded by superimposing the area like Figure 6 on the area like ether Figure 4 or Figure 5.

(ii) For the average number of rekeys $\leqslant$ ether formula (16) or formula (17);
The inequality that the average number of rekeys $\leqslant$ formula (17) is transformed to (18).

$$\begin{aligned} &\lambda_1(1+\lceil \log M_1 \rceil) \\ &\quad +\lambda_2(2+\lceil \log M_2 \rceil)+\lambda_3(2+\lceil \log M_3 \rceil) \\ \leqslant\ &\lambda_3(1+\lceil \log M_3 \rceil) \\ &\quad +(\lambda_1+\lambda_2)(1+\lceil \log M_1+M_2 \rceil). \end{aligned} \quad (18)$$

We consider the condition for this inequality in the following lemma. And we also consider it for formula (16) similarly.

Lemma 4. *The average number of rekeys $\leqslant$ formula (17), if for $M_1 = 2^{m_1} - r_1$ where positive integer $m_1 \geqslant 1$ and positive real number $r_1 \geqslant 0$, and positive integer $l \geqslant 1$, one of the following conditions satisfies(Figure 8).*

(i) When $0 < M_2 \leqslant r_1$, then

$$\lambda_2 \geqslant \frac{\lambda_3}{m - 1 - \lceil \log M_2 \rceil};$$

(ii) When $r_1 < M_2 \leqslant 2^{m_1} + r_1 = 2^{m_1+1} - M_1$, then

(a) if $\lceil \log M_2 \rceil < m$, then

$$\lambda_2 \geqslant \frac{\lambda_3 - \lambda_1}{m_1 - \lceil \log M_2 \rceil};$$

(b) if $\lceil \log M_2 \rceil = m$, then $\lambda_3 \leqslant \lambda_1$

(c) if $\lceil \log M_2 \rceil = m+1$, then $\lambda_2 \leqslant \lambda_1 - \lambda_3$;

(iii) When $2^{m_1+l} - M_1 < M_2 \leqslant 2^{m_1+l}$*, then* $\lambda_3 \leqslant \lambda_1(l+1)$*;*

(iv) When $2^{m_1+l} < M_2 \leqslant 2^{m_1+l} - M_1$*, then* $\lambda_2 \leqslant \lambda_1(l+1) - \lambda_3$.

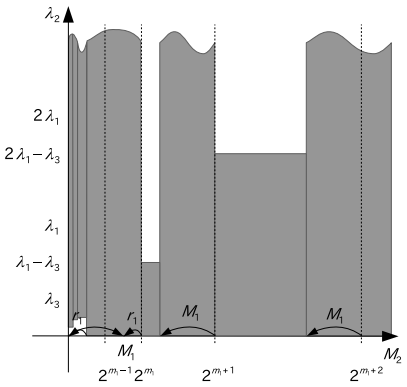

Fig. 8. In the case of $\lambda_2 + \lambda_3 \leqslant \lambda_1$

Since the proof of this lemma is similar to the proof of Theorem 1, we omit the proof.

If we have an algorithm A such that it induces a subgroup which satisfies the given constraint, we can induce three subgroups that are valid-separations by executing the following procedure:

(i) Let $G = (M, \lambda)$ be the given group.

(ii) By giving the constraint that consists of $1/2M$ as the maximum number of participants, $1/2\lambda$ as the minimum average rate of rekeys, and the condition of Theorem 1 to the algorithm A, we try to obtain $G_1 = (M_1, \lambda_1)$ from the algorithm A. If we can not obtain it, we terminate this procedure.

(iii) Let $\overline{G} = G \backslash G_1$, $\overline{\lambda} = \lambda - \lambda_1$, $\overline{M} = M - M_1$. By giving the constraint that consists of the condition of Theorem 1 and the condition of Theorem 3 for G_1 to the algorithm A, we try to obtain G_2, G_3 from the algorithm A such that both G_2 and G_3 satisfy the constraint. If we obtain them, we have G_1, G_2, and G_3 as valid-separation. Otherwise, we can not obtain them, we have G_1 and $\overline{G}$ as valid-separation.

Note that the reason why let $\lambda_1 \geqslant 1/2\lambda$ in the first step, is that since $\lambda_1 > \lambda_2 + \lambda_3$ always holds whatever G_2 and G_3 are selected, G_1 must be placed at the most upper level of the key tree.

3.6. Convenient method for selection of three subgroups

In the former section, we propose the method to induce three subgroups. However, the used constraint in the method is very complicated. Then, in this section, we propose another method whose constraint is simpler.

First, in Theorem 1, the condition that $M_1 \leqslant 1/4M$ and $\lambda_1 \geqslant 1/2\lambda$ always holds.

By applying this condition for G_1, G_2 and G_3, we have $3M_1 \leqslant M_2$, $3M_1 \leqslant M_3$, $3M_2 \leqslant M_3$ and $\lambda_1 \geqslant \lambda_2 \geqslant \lambda_3$. This induces $12M_1 \leqslant M_2 + M_3$. Finally, we let the condition in the first step be $M_1 \leqslant 1/13M$, $\lambda_1 \geqslant 1/2\lambda$. If we can not have G_1 satisfies the above condition, we try to consider to find valid-separation of two groups.

If we have G_1 where $M_1 \leqslant 1/13M$, we let the condition in the second step be $3M_1 \leqslant M_2 \leqslant 1/4(M - M_1)$ and $\lambda_2 \geqslant 1/2(\lambda - \lambda_1)$. If we get G_2 satisfying the above condition, G_3 also satisfies it. Moreover, G_1, G_2 and G_3 obtained for these constraints, also satisfy the condition of Theorem 3 and the condition of Lemma 4. Therefore, G_1, G_2 and G_3 are valid-separation.

4. Conclusion

For LKH, which is the key management method for IP Multicast communication in cipher, the former study proposed the method that managing the key tree by using a balanced binary tree. In this paper, we proposed a method that managing the key tree by separate user group from the behavior of users. Then, we show the condition that we reduce the average number of rekeys. Moreover, we can improve our method by applying the method of Batch LKH.

We are improving our method in efficiency in future.

References

1. D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. RFC 2627 (Informational), June 1999.
2. Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. In *IEEE/ACM Transactions on Networking*, volume 8, pages 16–30, 2000.
3. Pin-Yun Tarng, Kuan-Ta Chen, and Polly Huang. An analysis of WoW players' game hours. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 47–52, 2008.
4. Xiaozhou Steve Li, Yang Richard Yang, Mohamed G. Gouda, and Simon S. Lam. Batch rekeying for secure group communications. In *Proceedings of Tenth International World Wide Web Conference*, pages 525–534, 2001.
5. Josep Pegueroles and Francisco Rico-Novella. Balanced batch LKH: New proposal, implementation and performance evaluation. In *Eighth IEEE Symposium on Computers and Communications*, page 815, 2003. iscc.
6. M. Baugher, R. Canetti, L. Dondeti, and F. Lindholm. Multicast Security (MSEC) Group Key Management Architecture. RFC 4046 (Informational), April 2005.
7. Michel Abdalla, Yuval Shavitt, and Avishai Wool. Key management for restricted multicast using broadcast encryption. *IEEE/ACM Trans. Netw.*, 8(4):443–454, 2000.
8. D. Pisinger. A minimal algorithm for the bounded knapsack problem. In J. Clausen E. Balas, editor, *Integer Programming and Combinatorial Optimization, Fourth IPCO conference*, volume 920 of *Lecture Notes in Computer Science*, pages 95–109. Springer, Berlin, 1995.
9. David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, September 1952.