EURECOM
Sophia Antipolis

Institut Eurécom
Department of Corporate Communications
2229, Route des Cretes
B.P. 193
06904 Sophia-Antipolis FRANCE

Research Report RR-03-095

How to address secure multicast with a customer perspective?

November 2003

Melek Önen , Refik Molva

Tel : (+33) 4 93 00 26 26
Fax : (+33) 4 93 00 26 27
Email : {onen,molva}@eurecom.fr

[1]Institut Eurécom's research is partially supported by its industrial members : Bouygues Télécom, Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Télécom, Hitachi, ST Microelectronics, Swisscom, Texas Instruments, Thales

Abstract

Even though multicast rekeying is one of the most visited areas in network security, solutions still are severely lacking with respect to reliability and real customer expectations. Since any rekeying operation causes the update of all members' keying material, long-lived members are strongly affected by frequent membership changes.

In this paper, we suggest a new approach that takes into account different recipient categories based on their "loyalty" and that treats each category differently by offering better service to more loyal recipients. In our first solution, we propose to restructure the Logical Key Hierarchy (LKH) scheme by separately regrouping members based on their membership duration aiming at preserving members with long duration membership from the impact of rekeying operations caused by arrivals or departures of short-lived members. We then propose an extensive method for computing system parameters like rekeying intervals based on the customer satisfaction criteria.

1 Introduction

Multicast rekeying is one of the most visited areas in network security. Yet the existing solutions still are severely lacking with respect to reliability and real-life customer expectations. In this paper, we suggest a new approach that takes into account different recipient categories based on the "loyalty" concept and that treats each category differently by offering better service to more loyal recipients. In our first solution as presented here we take a simple definition of loyalty based on the membership duration with respect to the multicast group. Our solution then aims at preserving loyal members with long-duration membership from the impact of rekeying operations caused by less loyal members whose membership is shorter by definition. A typical metaphor for this goal is given in real life in the case of a meeting in a room : the goal of organizers is to preserve the participants of a meeting from frequent openings of the door by other people in search of their meeting door.

Most of existing multicast rekeying solutions require reliable delivery of new keys to members for group rekeying. In particular, in the Logical Key Hierarchy (LKH) scheme proposed by Wong et al. [1] and Wallner et al. [2] and proved to be communication optimal in [3], the key server uses keys of one rekeying interval to encrypt new keys of the subsequent one. Therefore, there is a strong dependency between keys of subsequent intervals. When a new key doesn't reach its intended recipient because of some packet losses, in the following rekeying interval, members affected by these losses will not be able to access future rekeying material.

Recently, some studies [4, 5, 6] have focused on this issue and different reliability schemes using Forward Error Correction (FEC) [7] or retransmission techniques (ARQ) have been proposed aiming at reducing the probability of losses in a rekeying. However, in all proposed solutions, the LKH scheme still suffers from the "one affects all" scalability failure [8] which occurs when the arrival or departure of a member affects the whole group. Each arrival or departure of a single member causes the update of at least one key with all members. Consequently, members who don't leave the group during the entire session can be strongly affected by frequent membership changes. From a commercial point of view it is unfair for a member who'll stay until the end of the session to be equally treated with short-lived members.

In this paper, we investigate how to assure higher reliability for members staying in the group during almost the whole session. To achieve this aim, we propose a restructuring of the key tree and split the group into 2 different sets with regards to members' membership duration. The reliability assurance[1] for members of each

[1]In this paper, we only deal with the reliability of rekey packets; the reliability of data packets

different set will increase proportionally with the membership duration of the corresponding members.

We first briefly describe the LKH scheme and review its failures in terms of scalability and reliability. We then introduce our solution based on a new partitioning scheme. After summarizing the partitioning idea, we describe the proposed protocol and give some results on optimized system parameters aiming at offering the best reliability to members with long duration membership.

2 Problem Statement

The LKH scheme was proposed independently by Wong et al. [1] and Wallner et al. [2] and proved to be communication optimal. After giving a brief description of the scheme, we review its shortcomings in terms of scalability and reliability.

2.1 Logical Key Hierarchy : LKH

In this scheme, the key server constructs and maintains an almost balanced tree with N leaves and N is the group size. A random key is attributed to each node where each leaf node corresponds to a unique member of the group. The key corresponding to the root node is the data encryption key. Each member R_i receives the set of keys corresponding to the path from the root of the tree to its corresponding leaf. Referring to the example in figure 1, R_1 would receive the key set $\{k_0, k_1, k_3, k_8\}$ where k_0 represents the data encryption key.

To remove a member from the group, all keys associated with the vertices of the path from the root to the leaf corresponding to the leaving member are invalidated. The rekey operation then consists of substituting for these invalidated keys with new values and broadcasting the new values in key envelopes encrypted under keying material known by remaining members. As depicted in figure 1, if member R_4 leaves the group, k_4, k_1 and k_0 are updated with k_4', k_1' and k_0', respectively. The key server then broadcasts $E_{k_{10}}(k'_4), E_{k_3}(k'_1), E_{k'_4}(k'_1), E_{k'_1}(k'_0)$ and $E_{k_2}(k'_0)$.

To add a member, the key server extends the tree with an additional leaf. The server marks again all keys associated with the vertices on the path from the leaf to the root as invalid. A random key is assigned to the new leaf and transmitted with a secure unicast channel. All other nodes in the path are updated with the same algorithm as the rekeying operation for a leaving member.

lies beyond our concern.

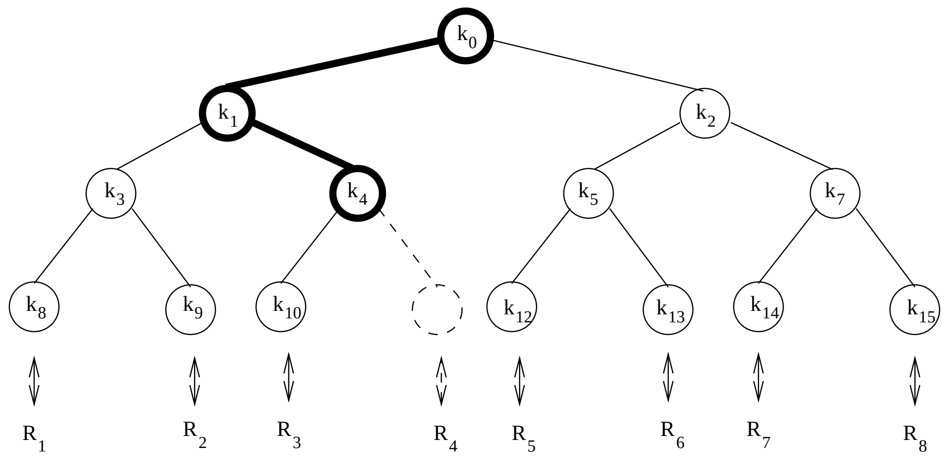

Figure 1: An Example of the LKH scheme

2.2 Shortcomings of LKH

Although the LKH scheme has been proved to be optimal in [3], it still suffers from some drawbacks in terms of scalability and reliability. Hence, when there are frequent arrivals or departures, individual rekeying becomes inefficient and the key server needs a strong reliable key delivery protocol because of the existing dependency between keys of subsequent different intervals.

2.2.1 Individual rekeying

At each arrival or departure of a member, the key server needs to immediately rekey the whole group in order to ensure backward and forward secrecy [8] which respectively prevents a member from accessing the data sent before its arrival or after its departure. However, individual rekeying is relatively inefficient in large groups where join/leave requests happen very frequently. For example, referring to the example in figure 1, if members R_3 and R_4 leave the group one after the other with a very short delay between the two departures, the key server will need to modify twice, the keys located at same vertices in the tree. If on the contrary, the key server had regrouped these two departures in one rekeying operation, the rekeying cost would be reduced by a half.

Batched rekeying algorithms have therefore been proposed in [9] whereby leave and join requests collected during an interval are processed by rekeying operations performed during the subsequent interval. An evaluation of the batch rekeying scheme in [9] shows a clear advantage over individual rekeying. Considering a group of 4096 members regrouped in a key tree of degree 4, in the case of 400 leaving members, batch rekeying requires approximatively 2147 encrypted keys while individual rekeying requires 9600 keys.

2.2.2 Key dependency

Although batch rekeying improves the efficiency of LKH by reducing the rekeying cost, it doesn't completely solve the synchronization problem between each member and the key server [5]. At a new rekeying interval, the key server uses the keys of the previous interval to encrypt new keys. Because of this strong dependency between keys, when a member looses some rekeying packets during a rekeying interval, it needs to contact the key server to refresh its key set, otherwise it will never again be able to decrypt multicast data sent after this rekeying interval even if it still is member of the group. Thus, the key server needs to ensure the receipt of keys by a maximum number of members before the beginning of the next rekeying interval.

Furthermore, since all members need to update their set of keys at every rekeying interval, frequent arrivals and departures should not affect members that are supposed to stay in the group until the end of the session. The key server must thus minimize the impact of rekeying due to the frequent dynamics of short-lived members on members that remain over longer periods of time since the service is offered to them for the entire session. This problem is discussed in the following sections and thanks to the partitioning scheme, the reliability assurance to each partition increases based on members' membership duration.

3 The solution

In our solution, the key server builds a representation that regroups members in different categories based on their membership duration. The degree of reliability assurance for each category increases with their "loyalty" degree. In order to achieve an almost full reliability solution for "loyal" members, the key server will evaluate and adapt system parameters such as rekeying intervals' length.

3.1 Partitioning

Frequent membership dynamics should rarely affect members who are supposed to stay in the group during the whole session. To achieve this aim, the key server needs to separately regroup members in n different sets with regards to the time they have spent in the group and offer a more reliable delivery to those whose membership duration is longer.

In [10], Almeroth et al. observed the group members' behavior during an entire multicast session. The authors realized that members leave the group either for a very short period after their arrival or at the end of the session. Based on these results, we define two real categories to distinguish members :

- short-duration members are supposed to leave the group a very short period after their arrival;
- long-duration members are on the opposite supposed to stay in the group during the entire session.

Since the key server cannot predict the time a member will spend in a multicast session, it cannot decide if a member belongs to the short-duration category or the long-duration one. Thus, we propose to separately regroup members into two monitored categories. In this proposed partitioning, a new coming member is first considered to be **volatile**. If this member spends more than a certain threshold time w in the group, then it becomes **permanent**.

Thanks to this partitioning, **permanent** members will not be affected from departures of **volatile** members but only from departures of members from their subgroup which is supposed to be quasi-static. The reliability processing of each monitored category will be different and the key server must guarantee to almost all **permanent** members the receipt of all necessary keys with a very high probability before the receipt of multicast data encrypted with these keys.

3.2 Rekeying the two monitored sets : Overview

As depicted in the previous section, members are separately regrouped in 2 disjoint sets :

- the set representing **volatile** members whose membership duration is less than w;
- the set representing **permanent** members whose membership duration has exceeded w.

For efficiency reasons, each monitored category is represented by a key tree. However, unlike the classical key tree approach, the key located at the root of each tree is not the data encryption key which is unique for the whole group but it corresponds to a key encryption key which is different for each tree.

Assuming that **volatile** members' departures will happen very frequently, in order to limit the number of leaving members and the extra-time they can stay in the group, the key server sets T_v as short as possible. The common data encryption key will thus be modified while rekeying **volatile** members. On the other hand, since **permanent** members are assumed to stay longer in the group, the key server grants a longer extra-time to these members after their real leaving-time. Thus, the rekeying interval T_p will be set to be longer than T_v. We define T_p as $T_p = kT_v$.

Since the data encryption key is modified every T_v with the rekeying of **volatile** members, **permanent** members still would be affected by losses resulting from this rekeying operation. Thus, during each T_p whereby no rekeying for permanent members takes place, an additional feature of our scheme allows permanent members to retrieve new data encryption keys resulting from rekeying operations at each T_v from their local keying material and without any information from the key server.

After having defined the global rekeying architecture, the key server now needs to adjust T_v and T_p with regards to system parameters. On one hand, to increase the quality of service, the key server needs to increase as much as possible T_v and T_p to be able to offer almost full reliable delivery of necessary keys. On the other hand, increasing these values implies to let more extra-time to leaving members since rekeying is processed in a batch for efficiency reasons. As a result, T_v and T_p should be as small as possible for security reasons but large enough to offer a better service to **permanent** members.

4 Protocol Description and Analysis

In this section, after presenting the rekeying process of our proposed protocol, we describe how to optimize necessary system parameters in order to increase the reliability factor to members with long duration membership.

4.1 Environment

Based on the results in [10], membership duration can be represented by two exponential distributions where the mean duration of membership for short-duration members and long-duration members is denoted by M_s and M_l, respectively. The ratio of short-duration members over N, the total group size, is denoted by γ.

Volatile and **permanent** members are separately regrouped in 2 respective key trees $\mathcal{G}_v$ and $\mathcal{G}_p$ of degree d whose respective root keys are denoted by K_v and K_p. The data encryption key K_{data} is updated every T_v at the same time as the update of $\mathcal{G}_v$. The update process for both trees is presented in section 2.1.

The update of $\mathcal{G}_p$ occurs every T_p where $T_p = kT_v$. During one T_p, at each T_v, **permanent** members automatically retrieve their key without waiting any extra-information from the key server. The key retrieval algorithm is described as follows

:

$$K_{data_{(i+1)}} = PRF(K_p, K_{data_{(i)}}) \tag{1}$$

Here PRF denotes a pseudo-random function (see [11] for further details) and provides forward secrecy for **volatile** members since they don't have the knowledge of K_p.

4.2 Rekeying processing

A new coming member R_i first joins the tree representing **volatile** members $\mathcal{G}_v$ and receives the actual data encryption and its corresponding set of keys. Every T_v, R_i receives the new data encryption key and the necessary information to update its keys like described in section 2.1. When R_i's membership duration reaches w, it is directly transfered to the key tree representing **permanent** members and it receives the new K_{data} and its new set of key encryption keys without waiting the next T_p. After its transfer to $\mathcal{G}_p$, each T_v the data encryption key is automatically computed and $\mathcal{G}_p$ is updated every T_p.

4.3 Optimizing system parameters

In the sequel of this section, we evaluate possible values for T_v and T_p.

To evaluate T_v, the key server computes the average number of leaving members from the **volatile** set and from this information it evaluates an average rekeying cost including the reliability factor which is not very large as for **permanent** members. The reader can refer to [5] for the computation of the average rekeying cost.

Assuming that the system is in a steady state, given all system parameters, the mean number of **volatile** members leaving the key tree every T_v is the sum of the average leaving members from the two real categories, ie. long and short-duration categories. We have :

$$L_v = \gamma N(1 - e^{-T_v/M_s}) + (1-\gamma)N(1 - e^{-T_v/M_l}) \tag{2}$$

Let $cost(L_v)$ be the average rekeying cost based on L_v[2] and the overhead of packets ensuring reliability[3]. T_v must then satisfy the following inequality where

[2]To compute L_v, the key server sums the average leaving members with short-duration and long-duration membership. In the case of an exponential distribution with a mean M the probability that a member leaves at T_v is $p(t \leq T_v) = 1 - e^{-T_v/M}$.

[3]The overhead of packets ensuring reliability has been evaluated and analyzed both with FEC and ARQ techniques.Details of this evaluation is not included in this paper due to space limitations

B is the necessary bandwidth only reserved for the rekeying operation :

$$cost(L_v) < B \times T_v \tag{3}$$

Symmetrically, the key server needs to adjust T_p in order to assure an arbitrarily high degree of reliability to **permanent** members independently of the number of leaving members in this subgroup. The worst rekeying cost corresponds to the case where all keys of $\mathcal{G}_p$ except members' individual keys need to be modified. This case corresponds to the event when for every d members, 1 member leaves $\mathcal{G}_p$, d being the degree of the key tree.

Assuming that N_p is **permanent** members' group size, in the worst case, we have $L_p = \frac{N_p}{d}$ and each of the remaining members needs to receive all keys located on the path from the root to their corresponding leaf except the individual key. Thus the rekeying cost is :

$$cost(L_p) = \frac{(d-1)N_p}{d}([log_d(N_p)] - 1) \tag{4}$$

Given these results the total cost must not exceed the given bandwidth. Thus $T_p = kT_v$ must follow the following inequality which again yields a lower bound on T_p:

$$k \times cost(L_v) + cost(L_p) < B \times T_p \tag{5}$$

Once the value of T_p and T_v are determined, the next important parameter to be estimated is w. The main criterion for the estimation of w is to keep the partitioning of members as perceived by the key server as close as possible to the real categories. However, there exists a tradeoff between w and the rekeying cost of each tree, including the reliability overhead. Hence, if w were too small than the majority of real short-duration members would be identified as **permanent** members and this would again cause further reliability problems. On the other hand, if w were too large, long-duration members would stay longer in the set of **volatile** members and they then would always be affected from frequent membership changes. Thus, the key server needs to adjust w aiming at reducing the number of penalized real long-duration members.

In order to define w, based on γ corresponding to the ratio of short-duration members in the real partitioning, the key server can limit the number of permanent members to $(1-\gamma)N$. The number of **permanent** members in one T_p is thus defined by the following expression[4] :

$$N_p = k\gamma N e^{-w/M_s} + k(1-\gamma)Ne^{-w/M_l} \tag{6}$$

[4]Here, N_p corresponds to the number of members who didn't leave the group during a period w. The probability that a member doesn't leave the group before w where the time is distributed exponentially with a mean M is : $p(t \geq w) = e^{-w/M}$.

Thus, w that achieves the closest identification of real categories, should satisfy $N_p = (1-\gamma)N$.

4.4 Example

We assume that $N = 65536$ where 50% of the group are short-duration members with $M_s = 3$ minutes and $M_l = 3$ hours. The bandwidth reserved for rekeying is limited to 1 Mbps and the loss probability of a rekeying packet for each member is independent and equal to $p = 0.1$. Based on the optimization method, we then compute system parameters for an objective defined by a target probability for the rekeying rate as perceived by a large fraction of permanent members. The following setting for the rekeying intervals assures a quasi-certain rekeying rate for permanent members, that is 99.99 $\%$ of permanent members have 99.99% probability of receiving all rekeying packets :

$$\begin{aligned} T_v &\geq 46s \\ T_p &\geq 3220s \end{aligned}$$

Based on these values, we then are able to compute the threshold value w that would best fit the real partitioning (50% short duration members). Using the resulting value ($w \geq 21000$), the protocol will eventually identify 50% of members as permanent.

5 Conclusion

Most of existing solutions in secure multicast are severely lacking with respect to reliability and real customer expectations. Hence, in the LKH scheme, because of the inherent strong dependency between keys of different subsequent intervals, all members suffer from rekey packet losses regardless of their membership duration. Thus, we propose to separately regroup members into two categories as **volatile** and **permanent** members. A threshold value w sets the time at which a **volatile** member is considered **permanent**. In order to offer higher reliability to permanent members, the key server adjusts the rekeying intervals T_v and T_p of the respective two sets after computing their corresponding rekeying cost. The evaluation of the overhead of rekeying packets ensuring reliability is not included in this paper due to space limitations. Moreover, to keep the partitioning of members as perceived by the key server as close as possible to the real categories, the key server adjusts the threshold value w with regards to the number of permanent members.

Our current work focuses on investigating the suitability of our classification scheme to a range of membership distributions. Further validation of the analytical results will involve trace based experimental evaluations.

References

[1] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. In *ACM SIGCOMM 1998*, pages 68–79, 1998.

[2] Debby M. Wallner, Eric J. Harder, and Ryan C. Agee. Key management for multicast: Issues and architectures. Internet draft, Network working group, september 1998, 1998.

[3] J. Snoeyink, S. Suri, and G. Varguese. A lower bound for multicast key distribution. In *IEEE Infocom*, Anchorage, Alaska, April 2001.

[4] C. Wong and S. Lam. Keystone: a group key management system. In *Proceedings of International Conference in Telecommunications*, 2000.

[5] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam. Reliable group rekeying : A performance analysis. In *ACM Sigcomm*, San Diego, CA, August 2001.

[6] S. Setia, S. Zhu, and S. Jajodia. A comparative performance analysis of reliable group rekey transport protocols for secure multicast. In *Performance*, Rome, Italy, September 2002.

[7] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols. *ACMCCR: Computer Communication Review*, 27, 1997.

[8] Suivo Mittra. Iolus: A framework for scalable secure multicasting. In *Proceedings of the ACM SIGCOMM'97 (September 14-18, 1997, Cannes, France)*, 1997.

[9] Xiaozhou Steve Li, Yang Richard Yang, Mohamed G. Gouda, and Simon S. Lam. Batch rekeying for secure group communications. In *Tenth International World Wide Web conference*, pages 525–534, 2001.

[10] K. Almeroth and M. Ammar. Collection and modelling of the join/leave behavior of multicast group members in the mbone. In *Proceedings of High Performance Distributed Computing Focus Workshop (HPDC'96)*, Syracuse, New York USA, August 1996.

[11] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *CRYPTO*, pages 276–288, 1984.