Reliable Group Rekeying with a Customer Perspective

Melek Önen, Refik Molva
Institut Eurécom
Sophia-Antipolis - France
Email: {Melek.Onen,Refik.Molva}@eurecom.fr

***Abstract*— Even though group rekeying is one of the most visited areas in network security, solutions still are severely lacking with respect to reliability and real customer expectations. In this paper, we first classify secure multicast applications with regards to these expectations and suggest a new approach that defines different recipient categories based on their "loyalty" and that treats each category differently by offering better service to more loyal recipients. 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 describe an extensive method for computing system parameters like rekeying intervals based on the customer satisfaction criteria.**

I. Introduction

In secure multi-party communications, several solutions have been proposed and even adopted in order to deal with group rekeying. Yet, existing solutions still are severely lacking with respect to reliability and real-life customers expectations. Indeed, since in these solutions each rekeying operation requires the update of the keying material of all members alike, frequent rekeying caused by volatile members would strongly affect long-lived members. In this paper, we take a simple definition of loyalty and based on this definition, we classify members with regards to their membership duration. Our solution aims at preserving loyal members with long-duration membership from the impact of rekeying operations caused by less loyal members.

Recently, some studies [1], [2], [3] have focused on the shortcomings of the most efficient rekeying protocol, that is the Logical Key Hierarchy (LKH) [4] with respect to reliability and different solutions using Forward Error Correction (FEC) [5] 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 [6] which occurs when the arrival or departure of any single member causes the update of at least one key with all members. Consequently, members who do not leave the group during an entire session, can be strongly affected by frequent membership changes. From a commercial point of view, it is unfair for a member who is supposed to 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 for members of each different set will increase proportionally with the membership duration of the corresponding members.

We first overview existing multicast applications in order to classify them with respect to security and customer expectations. We then briefly describe the LKH scheme and review its failures in terms of scalability and reliability. Finally, we introduce our rekeying protocol based on a new partitioning scheme and give some results on optimized system parameters aiming at offering the required reliable delivery to members with long duration membership.

II. Classification of Secure Multicast Applications

In [7], whereby a taxonomy of multicast applications is presented, the authors define five categories for one-to-many multicast applications which we target with our protocol:

- Scheduled Audio/Video distributions: lectures, meetings, ... ;
- push-media: news headlines, weather updates, ... ;
- file distribution and caching: web site content, executable binaries, ... ;
- announcements: network time, multicast session schedules, ... ;
- monitoring: sensor equipments, manufacturing, ...

This taxonomy is very general in nature and does not highlight the differences between applications with respect to security. We thus propose a more specific classification of multicast applications based on their security requirements and pricing models. We first briefly overview these characteristics and then describe our classification.

A. Security Requirements and Pricing Models

The first and mostly common requirement needed in multicast security is confidentiality which offers, via cryptographic encryption algorithms, a restricted access to multicast content for only members of a defined multicast group. In the case of dynamic multicast groups where entities are supposed to

This work is partially supported by Alcatel Space Industries, Toulouse, France.

join/leave the group frequently we come up with two security requirements that are not found in traditional secure unicast communications: backward and forward secrecy.

Backward secrecy defines the requirement that an added member should not be able to access the multicast content transmitted before its addition to the multicast group. Symmetrically, forward secrecy defines the requirement that a member leaving the group should not be able to further access multicast content.

Multicast applications may or may not require backward and/or forward secrecy. This requirement will depend on the pricing model of the application. Indeed, in some actual broadcast applications, like current digital TV platforms, customers first subscribe to the service and then pay a fixed fee periodically. In this case, the service provider does not know any customer's behavior and let the clients pay for the whole application. On the other hand, some other applications are not subscription based and offer to clients the possibility to pay only the service they really had access to. This kind of pricing model better fits customers' needs.

B. The Proposed Classification

By combining the security requirements and the pricing models, we end up with 3 classes of secure multicast applications:

- Subscription based applications whereby clients pay a fixed fee for the entire session no matter its membership duration or arrival time. In this type of application, the service provider needs to ensure neither forward nor backward secrecy and rekeying only occurs when there is a need for updating keys to prevent cryptanalysis.
- Applications where the pricing mechanism is based on members' arrival time where clients pay at their arrival for the remaining time until the end of the session. Here, the service provider does not need to ensure forward secrecy, since clients pay for the remaining whole session. However, it needs to provide backward secrecy since the client did not pay for the time before its arrival.
- Applications where the pricing mechanism is based on members' membership duration where clients only pay for the service they really had access to. In this kind of applications, the service provider needs to ensure both backward and forward secrecy.

C. Our Problem Statement

In the remaining of the paper, we only deal with applications whereby the service provider must ensure backward and forward secrecy (ie. applications of the third type). Consequently, the degree of "loyalty" of any member depends on its membership duration. The service provider should offer a better reliability assurance for long-lived members.

In order to provide backward and forward secrecy, Wong et al. [4] and Wallner et al. [8] independently proposed the LKH scheme. Since LKH is proved to be communication optimal in [9], our solution is based on this scheme. Thus, we first give a brief description of LKH and review its failures in terms of scalability and reliability, and then introduce our protocol dealing with members' membership duration.

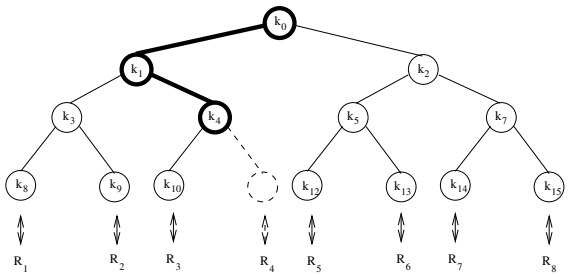

Fig. 1. An Example of the LKH scheme

III. Logical Key Hierarchy: LKH

A. Protocol Description

In this scheme, the key server constructs and maintains an almost balanced tree with N leaves where 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 rekeying 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 $E_{k_2}(k'_0)$ with $E_{k_x}(k_y)$ denoting the encryption of k_y with k_x.

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.

B. Shortcomings of LKH

Although the LKH scheme has been proved to be optimal in [9], 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.

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 [6] 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 [10] 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 [10] 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) Key Dependency: Although batch rekeying improves the efficiency of LKH by reducing the rekeying cost, it does not completely solve the synchronization problem between each member and the key server [2]. 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 reception of keys by a maximum number of members before the beginning of the next rekeying interval.

In order to deal with this problem, the authors in [3] have designed the WKA-BKR protocol which exploits the property that some keys are more valuable than others and defines the replication degree of a key based on its localization in the key tree. Moreover, Yang et al. [2] have proposed a reliable rekeying protocol based on the use of proactive-FEC in order to optimize bandwidth utilization.

3) The "one affects all" Failure: The LKH scheme inherently suffers from the "one affects all" failure. Indeed, 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. This problem is discussed in the following sections and thanks to the proposed partitioning scheme, the reliability assurance to each partition increases based on members' membership duration.

IV. Restructuring LKH with regards to Members' Membership Duration

In order to offer a better service to "loyal" members, 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.

A. 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 [11], 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 by 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.

B. Rekeying the Two Monitored Sets

As depicted in the previous section, members are separately regrouped in two 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, **volatile** and **permanent** members are respectively regrouped in two key trees denoted by $\mathcal{G}_v$ and $\mathcal{G}_p$, with K_v and K_p being keys located at the root of each tree. Unlike the classical key tree approach, K_v and K_p

are different from K_{data} which represents the data encryption key.

A new coming member R_i first joins the tree representing **volatile** members $\mathcal{G}_v$ and receives the actual data encryption key and its keying material. When R_i's membership duration reaches w, it is directly transfered to the key tree representing **permanent** members and it receives its new set of key encryption keys.

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 their rekeying interval T_v to a value as short as possible. 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 $T_p = k \times T_v$ where k is a positive integer needed to be determined.

However, since the key tree $\mathcal{G}_v$ representing **volatile** members is updated every T_v, the data encryption key (K_{data}) needs to be modified at the same time. In this case, **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. The key retrieval algorithm at each T_v during one T_p is described as follows:

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

Here PRF denotes a pseudo-random function (see [12] for further details) and provides forward secrecy for **volatile** members since they do not have the knowledge of K_p.

V. Protocol Analysis

The global rekeying architecture being defined, we now come to the crucial problem of determining the values of T_v, T_p and w. 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 an 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. In order to offer to them an almost fully reliable delivery of keying material, the key server needs to adjust these parameters by computing the rekeying cost of each category.

A. Optimizing System Parameters

In 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 computes an average rekeying cost including the reliability factor which is not as large as for **permanent** members. The reader can refer to [2] for the computation of the average rekeying cost.

Based on the results in [11], 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.

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[1] and the overhead of packets ensuring reliability[2]. T_v must then satisfy the following inequality where B is the necessary bandwidth reserved for the rekeying operation:

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

Symmetrically, the key server needs to adjust T_p (and thus k) in order to assure an arbitrarily high degree of reliability to **permanent** members independently of the number of leaving members in this subgroup. Hence, the key server must ensure the delivery of all keys even in the worst 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 and h_p is the depth of the corresponding key tree $\mathcal{G}_p$ ($h_p = \lceil log_d N_p \rceil + 1$), the worst rekeying cost without any overhead for reliable delivery is:

$$cost_{init}(L_p) = \sum_{i=1}^{h_p - 1} d^i \tag{4}$$

Given these results the total cost must not exceed the given bandwidth. Thus, $T_p = k \times T_v$ 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

[1]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 $p(t \leq T_v) = 1 - e^{-T_v/M}$.

[2]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.

categories. However, there exists a tradeoff between w and the rekeying cost of each tree, including the reliability overhead. Hence, if w was too small, then 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 was 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[3]:

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

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

B. 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 settings for the rekeying intervals assure a quasi-certain rekeying rate for permanent members, that is 99.99 $\%$ of permanent members have 99.99% probability of receiving all rekeying packets:

$$\begin{aligned} T_v &\geq 46s \\ T_p &\geq 4002s \end{aligned}$$

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

VI. 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 proposed protocol fits well for applications where there exists a strong requirement for backward and forward secrecy and where clients pay only for the amount of time they were present in the multicast group. A typical example for such applications is the "pay_as_you_watch" application. Further validation of the analytical results will involve trace based experimental evaluations.

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

Acknowledgment

The authors would like to thank Laurence Duquerroy and Sébastien Josset for their helpful comments on the final revision of this paper.

References

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

[2] 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.

[3] 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.

[4] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," in *ACM SIGCOMM 1998*, 1998, pp. 68–79.

[5] L. Rizzo, "Effective erasure codes for reliable computer communication protoco ls," *ACMCCR: Computer Communication Review*, vol. 27, 1997.

[6] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proceedings of the ACM SIGCOMM'97 (September 14-18, 1997, Cann es, France)*, 1997.

[7] B. Quinn and K. Almeroth, "IP Multicast Applications : Challenges and Solutions," RFC 3170, september 2001.

[8] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key management for multicast: Issues and architectures," Internet draft, Network working group, september 1998, 1998.

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

[10] X. S. Li, Y. R. Yang, M. G. Gouda, and S. imon S. Lam, "Batch rekeying for secure group communications," in *Tenth International World Wide Web conference*, 2001, pp. 525–534.

[11] K. Almeroth and M. Ammar, "Collection and modelling of the join/leave behavior of multicast grou p members in the mbone," in *Proceedings of High Performance Distributed Computing Focus Works hop (HPDC'96)*, Syracuse, New York USA, August 1996.

[12] O. Goldreich, S. Goldwasser, and S. Micali, "On the cryptographic applications of random functions," in *CRYPTO*, 1984, pp. 276–288.