EURECOM
Sophia Antipolis

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

Research Report RR-03-085

An Adaptive Hybrid Rekeying Protocol for Dynamic Multicast Groups

August 2003

Melek Önen

Tel : (+33) 4 93 00 26 26
Fax : (+33) 4 93 00 26 27
Email : onen@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

In secure multi-party communications, the Logical Key Hierarchy scheme has been proved to be communication optimal for large groups. However, this scheme still suffers from an expensive rekeying cost when the group is very dynamic. To reduce the rekeying cost, Zhu et al. suggested to partition the logical key tree in two sub-trees based on the duration of each member in the multicast group. Although this scheme reduces the rekeying cost for long-duration members, the key server still needs to send a potentially large number of rekeying messages for the set where member actions are very frequent. Based on the idea of separating long and short-duration members, we propose a hybrid rekeying protocol aiming at reducing the rekeying cost of the set of dynamic members. Thanks to this new protocol, the key server can adapt its rekeying scheme regarding to the frequency of membership operations. For each rekeying interval, the key server will first compute and compare the rekeying cost of three different schemes which differ regarding to the data structure defined for the dynamic set and will choose the scheme with the cheapest cost. Thus, the rekeying cost will always be optimized for each rekeying interval.

1 Introduction

In secure group communications, only members of a defined certain group are allowed to access the content of multicast data. All these data are communicated securely through symmetric encryption algorithms. In the case of large and dynamic groups, the data encryption key should be updated each time a recipient is added to the group or removed from. Consequently, the way of distributing the necessary key material is one of the most important problems. This mechanism of updating keys is also called the **rekey mechanism**.

In [1], Snoeyink et. al. have shown that the *Logical Key Hierarchy (LKH)* scheme proposed independently by Wong et. al. [2] and Wallner et. al. [3] is communication optimal. This scheme reduces the complexity of a group rekeying operation to *O(logN)*, where N represents the number of members of the group. The basic idea of this proposed scheme is to construct a logical key hierarchy represented by a tree where each node represents a random distinct key and every leaf corresponds to one of the member of the group. Each member is associated to the keys on the path from the root to its corresponding leaf and the key represented by the root is the data encryption key. When a member joins or leaves the group, only a a small number of keys of the tree are modified and sent after being encrypted with some other valid keys of the tree.

Although the LKH scheme has been proved to be optimal, it still suffers from some drawbacks in terms of scalability. Hence, if there is a rekey operation after each member action (join or leave), and these requests happen very frequently, individual rekeying can be inefficient. To improve the scalability of the scheme, the use of batched rekeying algorithms have been proposed in [4] whereby requests are collected during a predefined rekey interval and the set of new keys are sent in the next rekeying interval. Thus the number of encrypted keys generated by batch rekeying can be less then the sum of those generated by individual rekeying. Moreover, even if the batch rekeying algorithms outperform the rekeying after each member action in the case of the LKH scheme, the rekeying cost can still stay expensive enough. To reduce even more this cost, Zhu et al. proposed in [5], to regroup members based on the duration of their membership. While separately regrouping long and short duration members into two different subsets, when a short-duration member leaves the group, members from the other partition need only to receive one rekeying message.

Even if the regrouping scheme suggested by Zhu et al. drastically reduces the bandwidth overhead, rekeying short-duration members still requires the exchange of a significant number of messages. We propose to define a hybrid protocol that adaptively combines three different constructions in order to optimize the communication overhead for the short-duration members. At the beginning of each rekey-

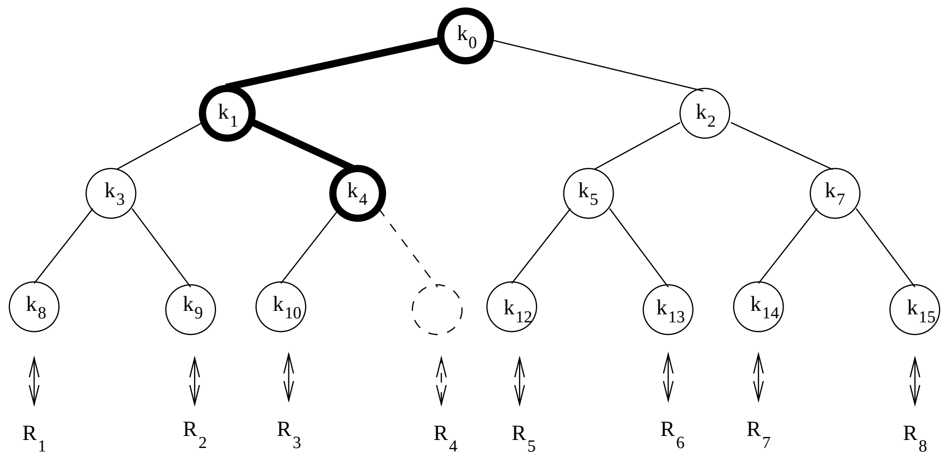

Figure 1: The LKH scheme

ing interval, the key server switches among three mechanism aiming at adapting the best construction reducing the most the rekeying cost.

We first present the LKH scheme and show its effectiveness when the group is considered to be very dynamic. We present existing solutions which reduce the rekeying cost of the scheme and further describe our solution in which the key server adapts the rekeying scheme regarding to the frequency of member actions. Finally, security and cost evaluation of the scheme are detailed in section 4.

2 Related Work

2.1 LKH : The Logical Key Hierarchy

The *Logical Key Hierarchy (LKH)* scheme was proposed independently by Wong et al. [2] and Wallner et al. [3]. The key server which is usually co-located with the source 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, the member R_1 would receive the key set $\{k_0, k_1, k_3, k_8\}$ where k_0 represents the data encryption key and k_8 the individual key of R_1.

To remove a member from the group, all keys representing the path from the root to the leaf corresponding to the leaving member are invalidated. These invalidated keys except the individual key of the leaving member are replaced with new random values and delivered optimally in multicast after being encrypted with some other valid keys of the tree. As illustrated in figure 1, if the member R_4 leaves the group, then the key server needs to broadcast $E_{k_{10}}(k'_4), E_{k_3}(k'_1), E_{k'_4}(k'_1), E_{k'_1}(k'_0)$ and $E_{k_2}(k'_0)$.

To add a member, the key server extends the tree with an additional node. If all leafs in the tree are already attributed to existing members, then the key server takes a leaf and create two children : the left child is assigned to the newly added member. The server makes again all keys in the nodes on the path from the leaf to the root 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 by encrypting the new keys with the corresponding old ones.

2.2 Reducing the rekeying cost

The LKH scheme defines a rekey operation at each membership operation. This scheme could be inefficient when requests happen very frequently. In [6], Yang et al. analyzed the cost of rekeying when requests are regrouped in a batch. To improve scalability, the key server collects requests during a rekey interval and the batched rekeying is done periodically. After evaluating the mean number of messages needed for rekeying the group, they have shown that this number can be much less than the sum of those generated by individual rekeying. Considering a group of 4096 members where 400 members want to leave, using a key tree of degree 4, batch rekeying generates 2147 encrypted keys while individual rekeying generates 9600.

Moreover, Zhu et al. proposed to reduce this rekeying cost after analyzing the group's membership behavior. They proposed to divide the group into two partitions based on the duration of the recipients' membership. A new recipient first joins the partition representing short-duration members and if he's still a member after a predefined threshold time, he is transfered to the other partition representing long-duration members. Based on this partitioning idea, the authors proposed and compared two different constructions for the two-partition algorithm where the data structures defined in each partition are different. Since the long-duration members are supposed to stay in the group during the whole session and therefore membership operations would not happen frequently, the key server defines a balanced key tree for this partition and the two constructions differ with respect to the data structure defined for the partition representing short-duration members.

In their first scheme called the *QT-scheme*, the key server uses a linear queue to represent short-duration members. Every short-duration member is supposed to store only one individual key and the new data encryption key needs to be encrypted with each member's individual key. The alternative scheme called the *TT-scheme* defines a balanced tree for each partition. Consequently, in this scheme, a short-duration member will need to store $log(N_s) + 1$ keys where N_s denotes the size of the corresponding partition and the key server will send enough messages to update the whole tree in a rekey operation. In terms of performance, the authors showed that when N_s is large, the *TT-scheme* is more scalable than the *QT-scheme* in terms of the number of messages to send by the key server for the rekeying. However the rekeying cost still is very expensive for short-duration members.

Therefore, we propose a new rekeying mechanism based on the partitioning of the group into two different subsets, where the key server can adapt the data structure resenting the short-duration members. In the proposed protocol, the key server will adapt the rekeying scheme independently for each interval according to the number of short-duration members' actions. Thus, the rekeying cost will be optimized.

3 A hybrid construction for the partition representing short-duration members

3.1 Motivation

While a key tree for the partition representing short-duration members reduces the bandwidth overhead, the rekeying cost still is very high. For example, if $N_s = 1024$ and the tree is fully balanced with degree 4, the revocation of 250 members implies an average cost of 778 keys to send from the key server. This cost is close to the one of *QT-scheme* where the key server needs to send $1024 - 250 = 774$ keys. Figure 2 represents a comparison between the *QT-scheme* and the *TT-scheme* of the rekeying cost as a function of the number of leaving members with 1024 short-duration members. We observe that the rekeying cost of the *QT-scheme* is lower than the *TT-scheme*'s average cost when the number of leaves is higher than one fifth of the total number of short-duration members.

Therefore, we propose a hybrid construction for the partition representing the short-duration members where the key server can choose independently for each interval which data structure to use for the rekeying mechanism in the aim of optimizing its cost. Moreover, we define a third rekey mechanism which can sometimes outperform the two other existing constructions, based on the revocation scheme proposed by Naor and Pinkas. This revocation scheme is based on threshold cryptography [7] using polynomial interpolation in a reciprocal way : the key server defines in advance the future key which will be used after the revocation of some members and distributes to each actual member a different secret share

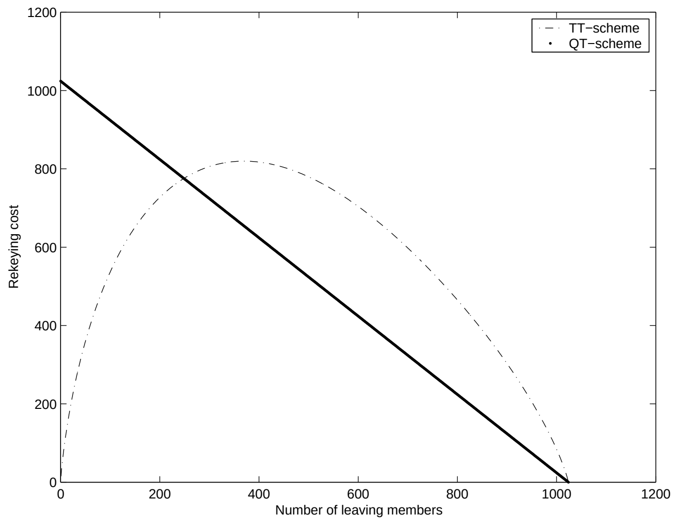

Figure 2: Rekeying cost for the QT-scheme and TT-scheme where $N_s = 1024$

for this key. At the revocation phase, it broadcasts the shares of revoked members and some new ones if needed, such that only the future legitimate members can combine all the shares and recover the data encryption key.

3.2 Member revocation with secret sharing

Definition : *A (k, n) threshold secret sharing scheme is a scheme where a secret K is divided into n shares in such a way that :*

- *any k shares can reconstruct the secret K;*
- *knowledge of $k - 1$ or fewer shares do not disclose any information about the secret.*

In his method [7], Shamir uses the principle of polynomial interpolation to define the key whose shares will be distributed to legitimate members. Formally, let $\mathcal{F}$ be a field such that a random element in $\mathcal{F}$ can be used as an encryption key. In this scheme, the initiator constructs a random polynomial $P(x)$ of degree n over $\mathcal{F}[x]$. The constant term of this polynomial is equal to the secret to be shared : $K = P(0)$. The kth share is defined as $P(k)$. Because of the degree of P, the combination of any $n + 1$ shares can reveal the polynomial by the interpolation technique and consequently K which is equal to $P(0)$. The reader can refer to [7] to verify that any n or fewer shares do not disclose any information about the secret.

The revocation schemes proposed by Naor and Pinkas [8] are based on Shamir's threshold scheme : The key server defines in advance a polynomial $P(x)$ where the future K is equal to $P(0)$. Each member R_i receives in a secure unicast communication its share $(I_i, P(I_i))$ where I_i is a unique identifier over $\mathcal{F}$ for the recipient R_i. The combination of any $n+1$ shares can reveal the polynomial by interpolation. Therefore, at the revocation phase, the key server sends the shares of the revoked users with other shares if needed, to let the legitimate members obtain $n+1$ shares (including their ones) and interpolate P to recover $K = P(0)$. Consequently, the key server can revoke at most n members for one revocation phase.

3.3 The proposed key distribution protocol

3.3.1 Preliminaries : the third construction for short-duration members

We propose to implement the scheme described in the previous section for the rekeying of the partition representing short-duration members and to use it when the LKH scheme and the *QT-scheme* are not efficient. However, since after one revocation phase the polynomial defining the new data encryption key is discovered by the actual authorized members, the same shares cannot be used for a future revocation phase. Thus, the key server needs to define at least one polynomial for each rekeying interval and distribute to the members their corresponding shares. Since the membership duration of these members is bounded, and after this threshold time they are transfered to the other more stable partition, each member needs to receive only the necessary number of shares until their transfer to the new partition. Let w determine this threshold time in terms of intervals. Each member needs then to receive shares from at least w different polynomials in advance.

Moreover, since the number of leaving members is unknown in advance, the key server may define several polynomials for each rekeying interval, in the aim of optimizing the bandwidth overhead. These polynomials will differ regarding to their degrees. The number of polynomials for each interval and their degrees can be determined based on the estimated number of leaving members or the ratio between the LKH batch rekeying cost and the secret sharing scheme's cost. For example, let's take the case where $N_s = 1024$ and the ratio $r = 0.5$. For a unique leaving member, the LKH scheme defines 19 keys to send. Consequently, our first polynomial will be of degree 9. For 10 leaving members, the LKH scheme needs to send on average of 116 keys. Therefore our second polynomial will be of degree 58 and if the number of leaving members is between 10 and 58, the key server will declare 58 shares of this chosen polynomial. From figure 2, we observed that in the case of the *TT-scheme*, after a certain number of leaving members, the rekeying cost for the key server starts to decrease. Therefore our last polynomial's degree for the interval would be equal to or less than the maximum leaving members where the secret sharing scheme is more efficient than the LKH one. After this level, the key server needs to restart to use the LKH scheme or the *QT-scheme*.

Furthermore, figure 2 represents only the average cost of rekeying regarding to the number of leaving members. However, in the real case, this rekeying cost can importantly decrease regarding to the location of leaving members in the key tree. Hence, if all leaving members are regrouped under a subtree of the whole tree, the rekeying cost can even outperform the secret sharing scheme. In our protocol the key server switches among these proposed two schemes and the secret sharing scheme based on the number of leaving members and the real rekey cost corresponding to each of the schemes.

3.3.2 Initialization

The key server defines the key tree for the group where the keys represented by the leaves of this tree are the corresponding members' individual keys. At interval j, the key server needs also to generate m random polynomials for each of the next w rekeying intervals which differ regarding to their degree:

$$\mathcal{P} = \{(P_{j,1}, .., P_{j,m}), ..., (P_{j+w-1,1}, ..P_{j+w-1,m})\}$$

The key server provides then to each member R_i on a private channel:

- $m \times w$ shares from the polynomials in $\mathcal{P}$ starting from interval j in which R_i joined the group : $\{R_i, \{P_{j,k}(R_i)..P_{j+w-1}(R_i)\}_{k\in[1..m]}\}$;
- the keys present in the path of the key tree from the root to the leaf corresponding to R_i.

3.3.3 Rekeying mechanism

Let t_l denote the number of leaving members at the rekeying interval T_l. The key server computes the real rekeying cost of the three schemes and compares them to decide which one to use for this interval :

- If the LKH scheme presents the lowest cost, then the key server will perform the classic rekeying mechanism for this scheme (see section 2.1) and the m shares of the actual interval will never be used.
- If there is a very large number of leaving members such that the *QT-scheme* outperforms the other schemes, then the key server will generate a random secret S and send it independently to each legitimate member after encrypting it with their individual key. The new data encryption key will be the result

of a pseudo-random function over this secret : $K_{new} = PRF(S)$. Since the key tree needs to be updated, all the key encryption keys must be modified as follows : $KEK_{new} = PRF(S, KEK_{old})$. Here, KEK_{old} denotes the key encryption key of the previous rekeying interval.

- If on the contrary, one of the polynomials of degree $n \geq t_{total}$ where t_{total} denotes the sum of the number of members who received one share of this polynomial and who has already left the group in w previous intervals and the number of actual leaving members denoted by t_l, than the key server will broadcast n values of this chosen polynomial including the t_{total} shares corresponding to all members who have left or who are leaving. Let $P_{l,k}$ denote the chosen polynomial. After having received and decrypted all these values, every legitimate member will be able to retrieve the secret $S = P_{l,k}(0)$ and compute the new data encryption key which is the result of a pseudo random function over this secret : $K_{new} = PRF(P_{l,k}(0))$. Furthermore, in order to assure forward secrecy, as with the *QT-scheme*, the key server needs to change all the values of the key tree. Thus, all key encryption keys except individual keys represented by the leaves of the tree are modified as follows : $KEK_{new} = PRF(P_{l,k}(0), KEK_{old})$.

Moreover, if there is no leaving member for a particular rekeying interval, then the corresponding shares are out of use, the new data encryption key, and all the key encryption keys except the members' individual keys are derived from the result of a pseudo-random function applied over previous keys as follows : $K_{new} = PRF(K_{old})$ and $KEK_{new} = PRF(KEK_{old})$. Therefore, when a member joins the group it will only have access to the actual data encryption key, its corresponding key encryption keys from the key tree including its individual key, and $w \times m$ shares of the polynomials defined for the corresponding interval and the $w - 1$ next ones. Furthermore, when a member reaches w intervals, he becomes a long-duration member and is transfered to its corresponding partition.

4 Security and Cost Evaluation of the proposed protocol

4.1 Security issues

Backward secrecy : Backward secrecy guarantees that a new member cannot have access to any secret data sent before this member joined the group. In the proposed hybrid protocol, since the key server can implement each of the three different schemes independently for each rekeying interval, we need to analyze the security of each of them :

- If the rekeying mechanism is based on the LKH scheme, then backward secrecy is obviously guaranteed by the scheme itself and a new member can only have access to the data sent after its addition to the group;
- Using the *QT-scheme* or the secret sharing scheme, all the keys of the key tree are updated. The current legitimate members retrieve respectively a secret S by decrypting it with their individual key or by interpolating one of the actual polynomial with the help of the shares they have received. Since the new data encryption key and the new key encryption keys are the result of a pseudo-random function over this secret and their corresponding previous keys, a new member who doesn't have any knowledge about S cannot retrieve any of these previous keys thanks to the one-way property of a pseudo-random function.

Forward secrecy : Symmetrically, forward secrecy prevents members from accessing secret data transmitted after their removal from the group.In this case, we again show with the 3 schemes that the revoked users can't have access to the future sent data :

- First of all, the LKH scheme also guarantees forward secrecy.
- Using the *QT-scheme*, the new data encryption key is encrypted with the individual keys of legitimate members. Hence, a revoked member will not have the possibility to retrieve neither the data encryption key nor the key encryption keys.
- While secret sharing scheme is used, the key server chooses the most efficient polynomial whose degree is greater than the number of all revoked members knowing a share of this polynomial. Thus, a coalition of any of these revoked users including those who left the group previously does not give any information about the new secret S, or any other key that is updated.

4.2 Cost evaluation of the scheme

With our hybrid protocol, since the key server decides which data structure to use after having compared the rekeying cost of each scheme, we offer the best optimized rekeying cost. However, in terms of storage overhead, each member needs to keep in addition to their keys from the key tree, the $w \times m$ shares. In [9], while studying the multicast group behavior of the Internet's multicast backbone Almeroth et al. have shown that group members either join for a very short period of time or stay for the entire session. Consequently, the threshold time w will be very low. Furthermore, from the figure 2, we observe that the average rekeying cost for the LKH scheme increases very fast. Thus, the key server may not define too many polynomials for one rekeying interval. Finally we offer an optimized rekeying cost for the two-partition scheme at the cost of a slight increase on the storage overhead for short duration members.

5 Conclusion

Although the LKH scheme has been proved to be optimal in terms of communication overhead, it's not always adaptable to groups where member actions happen very frequently. Thus, to reduce the rekeying cost, Zhu et al. proposed in [5] to regroup members based on their membership duration and to adapt the LKH scheme for long-duration members. However, the set representing short-duration members is the set which creates an expensive rekeying cost.

In this paper, we proposed hybrid rekeying protocol which initially defines two sets of members which differ regarding to their membership duration. Thanks to this adaptive scheme, the key server chooses the best rekeying mechanism among the described three data structures for the set where member actions happen very frequently. Thus, the communication overhead is optimized for the set representing short-duration members with a slight increase on the storage overhead at the side of the members.

References

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

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

[3] 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.

[4] 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.

[5] S. Zhu, S. Setia, and S. Jajodia. Performance optimizations for group key management schemes for secure multicast. In *The 23rd IEEE International Conference on Distribued Computing (ICDCS)*, May 2003.

[6] 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.

[7] A. Shamir. How to Share a Secret. *Communication of the ACM*, 22(11):612–613, November 1979.

[8] M. Naor and B. Pinkas. Efficient trace and revoke schemes. *Lecture Notes in Computer Science*, 1962:1–20, 2001.

[9] 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.