Simulation based performance evaluation of a user-centric group rekeying protocol

Melek Önen, Jussi Kyröhonka
Institut Eurécom
Sophia-Antipolis - France
Email: {Melek.Onen,Jussi.Kyrohonka}@eurecom.fr

***Abstract*— Most of the actual group rekeying solutions only deal with security and scalability issues and are severely lacking with respect to reliability and customer satisfaction. We suggested a reliable group rekeying protocol whereby the key server first partitions members with respect to their membership duration and offers a strongly reliable delivery for long-lived members. In this paper, we review this protocol that combines proactive FEC and replication techniques, describe a user-oriented key assignment scheme in order to define FEC blocks. We then analyze the efficiency of the protocol based on different simulations and show that thanks to this protocol, the number of long-lived members losing their keying material is significantly reduced while the communication overhead has slightly increased.**

I. Introduction

Group rekeying is one of the most visited areas in network security. Yet existing solutions remain to be implemented because they still are severely lacking with respect to real-life requirements such as reliability and customer satisfaction. When a new key does not reach its intended recipient because of some packet losses, members affected by these losses will not be able to access future rekeying material and the multicast data. Moreover, the addition or removal of a single member provoke the update of the keying material of all members alike.

In order to reduce the impact of any rekeying operation on at least a certain set of privileged members, we recently suggested in [1] a new approach whereby the service provider partitions recipients with respect to their membership duration, defines a set of privileged members (those with long membership duration) and offers them a strongly reliable delivery of the keying material. In this paper, we review the proposed protocol that is based on the Logical Key Hierarchy (LKH) scheme described in [2] and give a performance evaluation of this protocol based on simulation in order to understand that it offers an optimization in terms of scalability, reliability and answers to the requirement of customer satisfaction.

Section II gives a brief description of LKH and summarizes its requirements in terms of reliability and customer satisfaction. We then review the proposed rekeying protocol that first partitions members with respect to their membership duration and offers a strongly reliable delivery to long-duration members. We also describe a new user oriented key assignment algorithm that provides group members the ease of recovering their keying material from a single block of rekeying packets. Finally, we analyze and validate the efficiency of this protocol with simulations.

II. Problem Statement

A. Group Rekeying and LKH

The LKH scheme was independently proposed by Wong et al. [2] and Wallner et al. [3] and proved to be communication optimal in [4]. 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 and 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. In order to ensure backward and forward secrecy [5] which respectively prevents a member for accessing the data sent before its arrival or after its departure, the key server needs to rekey the whole group at each member join or leave.

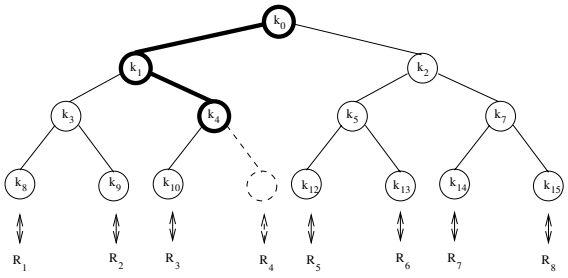

Fig. 1. An Example of the LKH scheme

For example, 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 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 $E_{k_2}(k'_0)$.

Although LKH has been proved to be communication optimal in [4], it still suffers from some drawbacks in terms of

scalability. Hence, when there are frequent arrivals or departures, individual rekeying becomes inefficient: at each arrival or departure of a member, the key server needs to immediately rekey the whole group. Batched rekeying algorithms have therefore been proposed in [6] and in [7] whereby join and leave requests collected during an interval are processed by rekeying operations performed during the subsequent interval.

B. Reliability Requirements

Existing reliable multicast protocols [8] cannot be considered as suitable to the delivery of the keying material where the information is assumed to be sensitive. Hence, LKH raises special properties that are not addressed by existing solutions. First, LKH exhibits the sparseness property: while a key server sends a set of encrypted keys, each member only needs a small fraction of these keys. In addition to this sparseness property, there is a strong relationship between rekeying packets: since updated keys are transmitted while being encrypted with some other keys, members should have received these encryption keys. We consider two different classes of relationships between keys that explain the reason of the need for a strongly reliable delivery of the keying material:

- the inter-dependency (time dependency) among keys raising the relationship between keys of different intervals;
- the intra-dependency (spatial dependency) among keys raising the relationships between keys transmitted within a single interval.

The key server should thus ensure that each member receives all of its updated keying material before the beginning of the subsequent rekeying interval. Recently some studies [9], [10] have focused on this issue and different reliability schemes using either Forward Error Correction (FEC) [11] or replication techniques have been proposed.

C. Customer satisfaction

While security, scalability and reliability are throughly addressed by existing reliable rekeying approaches, real life requirements such as customer satisfaction are overlooked or not even addressed by most solutions. Hence, in all proposed solutions, the LKH scheme still suffers from the "one affects all" scalability failure [5] which occurs when the arrival or departure of a member affects the whole group. Frequent rekeying operations may have a strong impact on all members alike, regardless of their behavior. The key server should at least minimize the impact of rekeying due to frequent arrivals or departures on a set of privileged members. In [1], we suggest a new approach that takes into account group members' membership duration and provide a strongly reliable key delivery for long-duration members in order to minimize the impact of rekeying operations caused by mass arrivals or departures of short-duration members.

III. The reliable partitioning protocol

A. Partitioning group members

In [12], 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. Therefore, we propose to partition 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 t_{th} in the group, then it becomes **permanent**.

Volatile and **permanent** members are respectively regrouped in two key trees denoted by $\mathcal{G}_v$ and $\mathcal{G}_p$ that are managed and updated separately and periodically. Requests are thus collected in batch. Assuming that **volatile** members' arrivals and departures will happen very frequently, the key server sets the corresponding 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 corresponding rekeying interval T_p will be set to be much longer than T_v.

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 delivery of the keying material with a very high probability before the receipt of multicast data encrypted with these keys.

B. The hybrid reliability scheme

We provide a hybrid reliability scheme that combines proactive FEC with proactive replication techniques. The structure of FEC blocks is defined thanks to the user oriented key assignment algorithm described in the following section. In this algorithm, any permanent member receives its complete required updated keys from one FEC block of rekeying packets.

1) The user oriented key assignment algorithm: Since LKH provides a hierarchical structure between keys and since members that are in a same subtree share a certain number of keys, we propose to split G_p into disjoint subtrees and define a specific block for members of each subtree. We denote d_p and h_p as the respective outdegree and depth of G_p. A member needs to receive at most $h_p - 1$ keys from G_p and the data encryption key. With respect to the chosen subtree, members share the keys associated with the nodes that are in the path from the root of G_p to the parent of the root of the resulting subtrees. We define the block size of a rekeying packet based on the parameter a such that $(d_p)^a$ defines the number of members of the specific subtree. In this case, the block regroups:

- the data encryption key encrypted with the actual root key of $\mathcal{G}_p$ denoted by $k_{p,0}$;

- all the encrypted keys in the path from the parent of the root node of the subtree to the root of G_p ($(h_p - 1 - a)$ keys). They are shared by all the corresponding members and thus are encrypted only once for this specific block;
- all the keys from the subtree except those at leaf nodes, encrypted with keys at children nodes: $\sum_{i=1}^{a}(d_p)^i$.

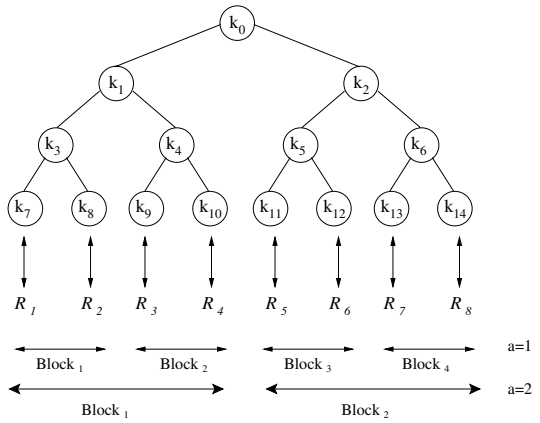

Fig. 2. An illustration for the User Oriented key assignment algorithm where $a = 1$ and $a = 2$

For example, in figure 2 where G_p is a binary key tree of depth 4, when the key server sets $a = 1$, it defines one block per two members. In this case, the key server partitions members in four separate subtrees.With this key regrouping method, some rekeying packets will appear in several blocks. Hence, if for example k_0 needs to be updated, members R_1, R_2, R_3 and R_4 must receive $E_{k_1}(k'_0)$. If the key server sets $a = 1$ as illustrated in figure 2, this rekeying packet will appear in $Block_1$ and $Block_2$. Therefore, if R_1 loses this specific rekeying packet from its corresponding block, it can get it from $Block_2$ without performing any additional operation. Thus, our scheme inherently combines proactive FEC and proactive replication techniques, and lets members who have not received their rekeying packets from their block, retrieve remaining keys from other blocks.

2) Providing customer satisfaction: In the proposed solution, since there is always a probability of loss, the key server first defines some bounds on the expected losses. It defines α and β such that α denotes the portion of **permanent** members that receive their keying material with probability at least as high as β. Given the number of **permanent** members N_p and assuming X is a random variable representing the number of **permanent** members that do not receive all of their corresponding rekeying packets, the previous requirement can be expressed by the following inequality:

$$P\{X > (1 - \alpha)N_p\} \leq (1 - \beta) \quad (1)$$

Based on this equation, the key server can define the reliability parameters in order to ensure the delivery of the keying material to almost all permanent members.

C. Related Work

In [13], similarly to our protocol, authors base their research on the observations made in [12] and propose to divide the key tree into two partitions regrouping members with respect to their membership duration. In [14], authors investigate the problem of member revocation and find the optimal number of keys distributed to each member. However, the aim of these works only were to optimize the overall communication overhead or the memory usage respectively. We, on the contrary, proposed the partitioning scheme in order to deal with the problems of reliability and customer satisfaction. In addition to the basic partitioning scheme, for each set of members, we set different rekeying intervals and different reliability parameters for each partition. We provide a longer rekeying interval for **permanent** members during which the data encryption key can be automatically retrieved by these members.

IV. Simulation based validation of the proposed protocol

We carry out a simulation based study of the rekeying protocol described in the previous section. The features of the protocol are analyzed in terms of simulation metrics that we deem relevant to assess its basic advantages over the classical LKH scheme:

- the communication overhead;
- the number of losses.

Simulation results are used to understand that the proposed rekeying protocol offers an optimization in terms of scalability, reliability and answers to the requirement of customer satisfaction over LKH. We therefore analyze the performance of the proposed rekeying protocol over the classical LKH scheme.

- **Case 1**: we only implement the partitioning scheme defining the two different key trees respectively for **volatile** and **permanent** members and the key server transmits rekeying packets without any additional packets;
- **Case 2**: we apply our user oriented key assignment algorithm where the key server regroups rekeying packets such that any member only receives one block to retrieve all of its updated keying material;
- **Case3**: we analyze the efficiency of the proposed hybrid reliability scheme which aims to offer a reliable delivery of the keying material to almost all permanent members given parameters α and β.

A. Implementation

The proposed rekeying mechanism is implemented as a discrete-event simulator in C. The events represent the actions of a key server and operations performed by recipients such as join, leave or change status to become **permanent**. In terms of reliability, we simulate a packet loss probability which is independent for each client as an input simulation parameter. We therefore can analyze the number of members losing at least one rekeying packet during each rekeying operation. Finally, we can also simulate different customer behaviors. The main parameters for this aspect are the inter-arrival and service

time distributions for each class of client (**short-duration** and **long-duration**). The simulator either uses a random variable generator, given the required parameters for each type of distribution or the client behavior can explicitly be specified. Thanks to this option, one can simulate the efficiency of the proposed rekeying protocol with respect to real customer behavior.

B. Simulation set-up

In our simulation, we consider two real categories of members that are equally present in the group: their inter-arrival time are distributed exponentially with a mean of one second. Their membership duration are distributed exponentially with a mean $M_s = 1000s$ for short-duration members and $M_l = 100000s$ for long-duration members. We set the simulation time to $T = 10000s$. The rekeying interval is set to $T_s = 60s$ for the single key tree scheme and to $T_v = 60s$ and $T_p = 60 * 10 = 600s$ for the proposed partitioning scheme. The packet loss probability is set to $p = 0.1$ and the threshold time to $t_{th} = 3000s$.

Based on these initial parameters, in the following sections we analyze and compare the performance of the proposed scheme with respect to LKH.

C. Case 1: Performance of the partitioning scheme with no reliability mechanism

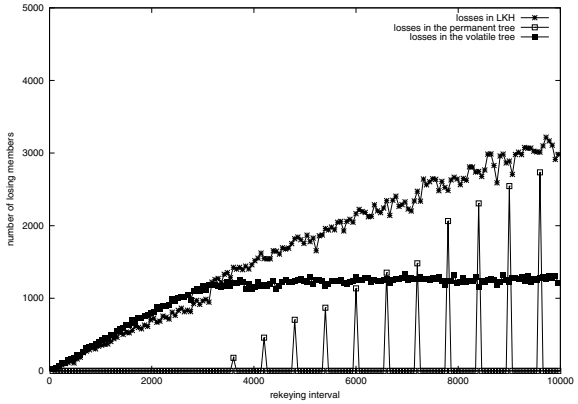

Fig. 3. Number of losses experienced by long-duration members in Case 1

Given the simulation set-up defined in section IV-B, we analyze the number of losses experienced by long-duration members in both schemes where there is no reliability method taken into account. From figure 3, we realize that, before the threshold value $t_{th} = 3000$, the number of losses experienced by long-duration members in the volatile key tree is close to the one observed in the single key tree case. Beginning from $t = t_{th}$, this number does not vary in the volatile key tree. This is due to the fact that long-duration members are transfered to the permanent key tree when their membership duration reaches t_{th} and thus the number of long-duration members considered as volatile becomes constant. Symmetrically, since long-duration members can join the permanent key tree at each T_v, the number of losses experienced in the permanent key tree increases with the number of long-duration members transfered to this key tree. Moreover, since in the proposed partitioning scheme rekeying operations are only performed every T_p, we observe that members only lose their keying material at each T_p.

TABLE I
TOTAL NUMBER OF LOSSES EXPERIENCED BY LONG-DURATION MEMBERS IN CASE 1

Single Key tree	Volatile key tree	Permanent key tree	Both key trees
282025	173729	15830	189559

From table I that presents the total number of losses experienced by long-duration members, we can derive that the proposed partitioning scheme reduces the number of losses by 33% with respect to the classical LKH scheme.

TABLE II
TOTAL REKEYING COST IN CASE 1

Single key tree	Volatile Key tree	Permanent key tree	Both key trees
123030	142412	5309	147721

Table II illustrates the total rekeying cost (number of rekeying packets) resulting from the classical LKH scheme and the proposed partitioning scheme. The proposed partitioning method reduces the rekeying cost for permanent members while increasing the total rekeying cost by 20%.

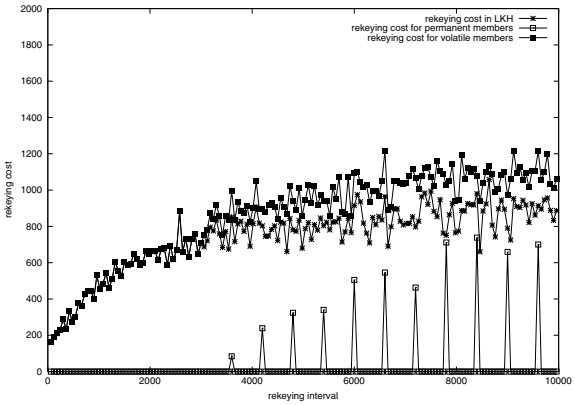

Fig. 4. Rekeying cost in Case 1

From figure 4 illustrating the rekeying cost for each key tree at each interval, we observe that the rekeying cost for volatile members is slightly higher than the rekeying cost of the LKH scheme. This difference is in fact due to the rekeying messages transmitted in order to move members, whose membership duration reaches the threshold value t_{th}, from the volatile key tree to the permanent one. In the same figure, the number of rekeying packets destined to permanent members is very low with respect to volatile members. From this observation, we can conclude that since the rekeying cost resulting from the update of the permanent key tree is low, given a bandwidth limitation, the key server can manage a strongly reliable delivery for members of this category.

D. Case 2: Impact of the User oriented Key assignment algorithm

We now analyze the efficiency of the proposed User-Oriented Key Assignment algorithm where the key server regroups rekeying packets such that any member only receives one block to retrieve all of its updated keying material. We first perform several simulations with different values of a and analyze and compare the performance of this algorithm with the previous case. We remind that this algorithm only is implemented for **permanent** members.

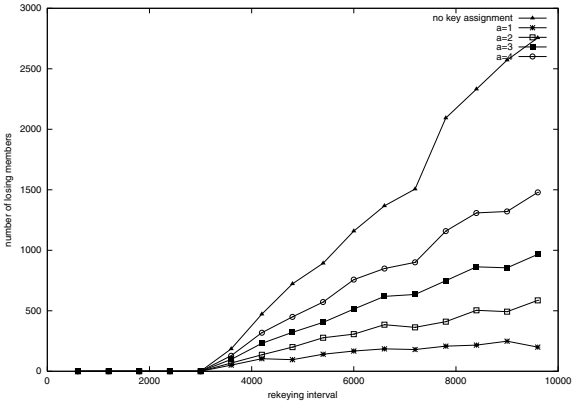

Fig. 5. Losses experienced by permanent members in Case 2

Figure 5 illustrates the number of **permanent** members losing rekeying packets at each T_p. We observe that this number increases when a increases. This fact was expected since when a increases, the number of members needing packets from this block increases, and packets are less replicated. However, the number of losses resulting from Case 1 (no key assignment) is always greater than those when the proposed user-oriented key assignment algorithm is implemented. We thus can conclude that the proposed user-oriented key assignment algorithm reduces the number of losses even when there is no FEC implementation.

Figure 6 illustrates the number of rekeying packets sent by the key server to **permanent** members. We only show the rekeying cost at each T_p, since the key server does not send any rekeying packet apart from these intervals. We observe that the rekeying cost resulting from the user-oriented key assignment algorithm is greater than the rekeying cost in the permanent key tree in Case 1. This observation is not surprising since some keys are inherently replicated in different blocks.

Moreover, in figure 5, we observe that when a increases, the number of losses increases almost linearly. From this observation, we can conclude that the key server must choose the minimum possible value for the block size. However, figure 6 shows that the rekeying cost decreases exponentially. We notice that between $a = 1$ and $a = 2$ the difference is very high and beginning from $a = 3$ the rekeying cost is almost stable. Since there is a tradeoff between these two simulation metrics, we can conclude that in order to achieve scalability with customer satisfaction, the key server can set $a = 2$ or $a = 3$.

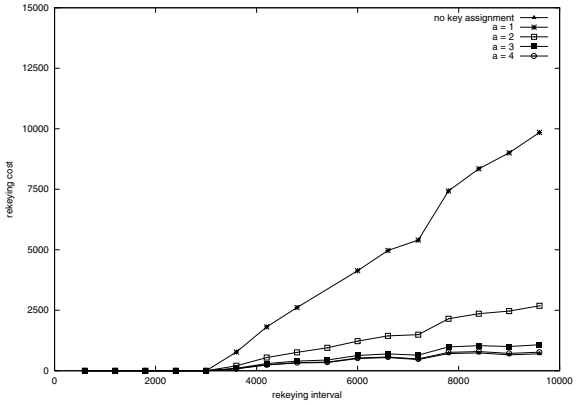

Fig. 6. Rekeying cost in Case 2

E. Case 3: Impact of the hybrid reliability scheme

Now that we analyzed the performance of the proposed user-oriented key assignment algorithm and the estimation of the optimal FEC block size based on a, we turn to the efficiency of the hybrid reliability scheme which defines the number of parity packets for each FEC block depending on α and β. In the previous simulations, we showed that the key server needs to set either $a = 2$ or $a = 3$ in order to optimize the performance of the key assignment algorithm. We thus only take these two cases into account and set $\alpha = 0.99$ and $\beta = 0.99$.

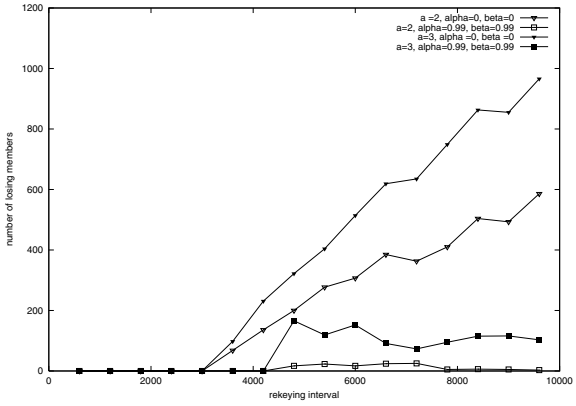

Fig. 7. Losses experienced by permanent members in Case 2 and Case 3

Figure 7 illustrates the number of losses experienced by **permanent** members for different values of a and for (α, β) set to $(0, 0)$ (Case 2) and $(0.99, 0.99)$. We observe that the implementation of the hybrid reliability scheme ($\alpha = 0.99$, $\beta = 0.99$) reduces the number of losses which almost reaches zero. Thanks to this simulation, we can conclude that the hybrid reliability scheme reduces the number of **permanent** members losing their keying material and achieves the requirement of customer satisfaction.

Figure 8 displays the rekeying cost with and without the implementation of the hybrid reliability scheme. We notice that the rekeying cost slightly increases when the key server

generates parity packets with $a = 3$ (based on $\alpha = 0.99$, $\beta = 0.99$) whereas for $a = 2$ the difference becomes much higher. Since the number of losses depicted in figure 7 remains acceptable when $a = 3$, we can conclude that this value achieves the best tradeoff between losses and rekeying cost.

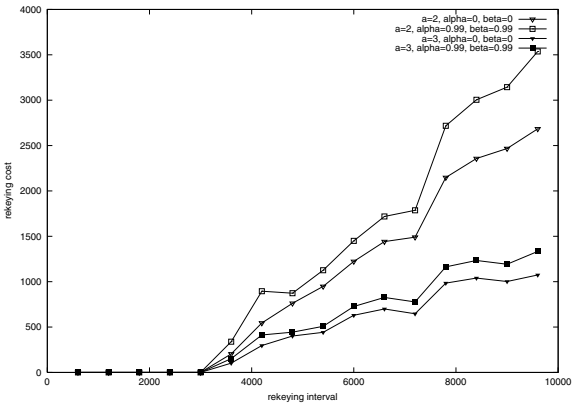

Fig. 8. Rekeying cost in Case 2 and Case 3

F. Summary

TABLE III
EFFICIENCY OF THE PROPOSED REKEYING PROTOCOL: SUMMARY

	Case 1	Case 2		Case 3	
		a=2	a=3	a=2	a=3
Total Losses	16058	3728	6256	286	1030
Total Rekeying Cost	5309	16270	7316	20592	10372

Table III summarizes the simulation results for each scenario. In this table, we illustrate the total number of losses and the total rekeying cost for the **permanent** key tree. We first realize that the use of the proposed User-Oriented Key Assignment algorithm significantly reduces the number of losses (77% for $a = 2$ and 61% for $a = 3$). Furthermore, the implementation of the hybrid reliability scheme shows a much better performance in terms of losses. Unfortunately, this advantage has an impact on the rekeying cost (about 100% of increase for $a = 3$ in Case 3). However, as shown in table II, this cost still remains low with respect to the **volatile** key tree's one (142412). Consequently, this additional cost will have a small impact on the total rekeying cost regrouping both **volatile** and **permanent** members (less then 4%).

V. CONCLUSION

In this paper, we analyzed the performance of the proposed rekeying protocol over LKH scheme in an incremental way. We first evaluated the efficiency of the partitioning scheme and showed that this scheme reduces the number of losses while slightly increasing the total rekeying cost. However, the rekeying cost resulting from the update of the **permanent** key tree being very low with respect to the one resulting from the update of the volatile key tree, the key server can transmit additional packets for members on this category in order to offer a reliable delivery. We then proposed to analyze the efficiency of the hybrid reliability method for **permanent** members. We first showed that the use of the user-oriented key assignment algorithm reduces the number of losses and that the key server needs to define the block size as high as possible in order not to observe a high rekeying cost. We then came up with the definition of additional parity packets based on α and β and showed that although this method increases the rekeying cost, it remains efficient with respect to the total rekeying cost resulting from the whole group (**volatile** and **permanent** members).

This study reveals that security concerns cannot be addressed without taking into consideration some other crucial network parameters and real-life expectations. In order to reduce the impact of the tradeoff between security, scalability and reliability requirements, we suggested a classification of recipients with respect to some real-life criteria such as the membership duration in order to optimize system parameters and offer a better service for privileged members.

REFERENCES

[1] M. Önen and R. Molva, "Group rekeying with a customer perspective," in *Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS'04)*, Newport Beach, California, July 2004.

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

[3] Debby M. Wallner, Eric J. Harder, and Ryan C. Agee, "Key management for multicast: Issues and architectures," RFC 26 27, june, 1999.

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

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

[6] 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*, 2001, pp. 525–534.

[7] F. Zhu, A. Chan, and G. Noubir, "Optimal Tree Structure for Key Management of Simultaneous Join/Leave in secure multicast," in *MILCOM*, 2003.

[8] J. W. Atwood, "A classification of reliable multicast protocols," *IEEE Network*, vol. 18, no. 3, May 2004.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] R. Poovendran and John S. Baras, "An information theoretic analysis of rooted-tree based secure multicast key distribution schemes," in *CRYPTO*, 1999, pp. 624–638.