Group Rekeying with a Customer Perspective*

Melek Önen , Refik Molva
Insitut Eurécom
2229 Route des Crètes BP 193
06904 Sophia-Antipolis FRANCE
{*Melek.Onen,Refik.Molva*}*@eurecom.fr*

Abstract

In secure multi-party communications, several solutions have been proposed to deal with group rekeying. However, most of existing solutions including the most efficient ones still are severely lacking with respect to reliability and real customer expectations. 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. We thus 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 designed a hybrid reliability scheme based on a combination of ARQ and FEC that assures a quasi certain delivery of keying material to long-lived members. We then come up with an extensive method to determine the system parameters applicable to each member set based on the target 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 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 does not 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. Our aim is to guarantee that almost all long-lived members receive their keying material on time, ie. before the receipt of the corresponding encrypted multicast data.To achieve this aim, we propose a restructuring

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

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 different set will increase proportionally with the membership duration of the corresponding members . In order to assure a quasi certain reliability to long-lived members, we propose a hybrid reliability scheme combining both FEC and ARQ techniques.

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 hybrid reliability scheme implemented for the rekeying operation of long-lived members and present an extensive method for computing optimized system parameters offering the required reliable delivery 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 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 $E_{k_2}(k_0')$.

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.

1 In this paper, we only deal with the reliability of rekey packets; the reliability of data packets lies beyond our concern.

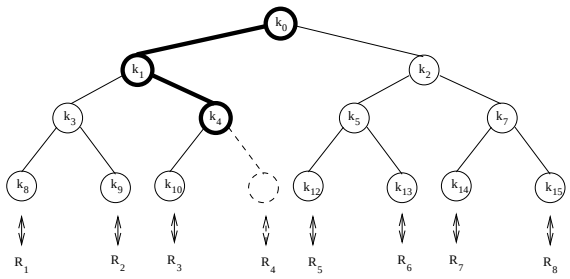

Figure 1. An Example of the LKH scheme

2.2. Shortcomings of LKH and related work

Although the LKH scheme has been proved to be communication 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 does not 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.

In order to deal with this problem, the authors in [6] 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. [5] have proposed a reliable rekeying protocol based on the use of pro-active-FEC in order to optimize bandwidth utilization.

2.2.3. "The one affects all" failure : In the LKH scheme, as well as in most of rekeying solutions, any arrival or departure of a recipient causes the update of the keying material of all members alike. Indeed, any rekeying operation at least requires the update of the data encryption key which is shared with all members of the group. In these conditions, frequent arrivals or 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 proposed partitioning scheme, the reliability assurance to each partition increases based on members' membership duration.

3. The solution

The basic idea behind our solution is that the key server partitions members in different categories based on their membership duration. The key server then uses error-correction mechanisms with a degree of reliability that depends on the "loyalty" of each category.

3.1. Partitioning

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

3.2. Rekeying the two monitored sets

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, **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.

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. The common data encryption key (K_{data}) 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 K_{data} is modified every T_v while rekeying **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. The key retrieval algorithm at each T_v during one T_p is described as follows :

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

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.

3.3. Membership management and rekeying process

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. 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 $\mathcal{G}_p$, during one T_p and at each T_v, R_i automatically retrieves the new data encryption key using the algorithm (1).

4. Optimizing system parameters

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 almost fully reliable delivery of keying material. 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 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 (including additional packets to offer reliability). We first introduce our hybrid reliability scheme which provides the evaluation of the rekeying cost for permanent members and then present the method to determine T_v, T_p and w.

4.1. Evaluation of rekeying cost for permanent members

In this section, after formally defining the reliability requirements of **permanent** members, we propose a hybrid method to ensure an almost fully reliable delivery of keying material and evaluate the cost of this reliability scheme.

4.1.1. Reliability requirements : Any key distribution scheme should provide a reliable delivery mechanism. Any member of the group should be able to receive its entire keying material, in order to be able to access the content of multicast data. In the context of the LKH scheme, this reliability issue becomes a more important problem because of the key dependency problem described in section 2.2.2. In our solution, it is required that every permanent member receives all of its keying material before the end of the corresponding rekeying interval, ie. before the receipt of the corresponding encrypted multicast data.

The group controller defines α, 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, the key server must ensure that the probability that more than $(1-\alpha)N_p$ of **permanent** members will not receive their corresponding keying material must not exceed $(1-\beta)$. Assuming X is a random variable representing the number of **permanent** members that don't receive all of their corresponding rekeying packets, the previous requirement can be expressed by the following inequality :

$$P\{X > (1-\alpha)N_p\} \leq (1-\beta) \quad (2)$$

To ensure reliability for rekeying protocols implementing the LKH scheme, some projects [4, 5, 6] proposed the use of FEC or retransmission techniques (ARQ) based on batch rekeying. We propose a hybrid reliability scheme which combines both techniques in order to achieve efficiency both at the sender and the recipients.

4.1.2. Hybrid reliability scheme : We have evaluated the performance of different existing reliability techniques for protocols implementing the LKH scheme and we found that the use of a hybrid reliability scheme combining FEC and retransmission techniques (ARQ) outperforms the existing ones. Due to space limitations, we only present here the proposed scheme.

We first need to define the structure of FEC blocks. In order to improve performance at the member's side, any member will only receive one block of rekey packets and if during the rekeying operation some losses occur, members affected from these losses will perform the key recovery operation only once. Consequently, we propose to split the key tree representing **permanent** members (with degree d and depth h) into disjoint subtrees of depth d^i where $i \in [1..h]$. Assuming that a member needs at most h keys, the block size $b(i)$ is defined as follows :

$$b(i) = h - i - 1 + \sum_{j=1}^{i} d^i \quad (3)$$

With this key regrouping method, some rekey packets will appear in several blocks. Thus, our scheme inherently combines FEC and retransmission techniques (ARQ), and lets members that are not able to recover their rekey packets, retrieve remaining keys from other blocks.

4.1.3. Cost of rekeying : The key server needs to assure the required degree of reliability to **permanent** members independently of the number of leaving members in both subgroups. The worst rekeying cost corresponds to the case where all keys of the key tree representing **permanent** members except their individual keys need to be modified. In this case, for every d members, one member leaves the group and thus each of the $(d-1)$ remaining members needs to receive all h keys located on the path from the root to its corresponding leaf except the individual key.

In our reliability scheme, given α, β, the chosen depth i for the subtree and the corresponding block size $b(i)$, the key server needs to compute the necessary number of parity packets per FEC block. The computed value must follow the source's reliability requirements given in the inequality (2). Let b be the block size, r be the number of parity packets for one FEC block and p be the packet loss probability. In order to compute the probability that a member R receives its specific packets in one block, we define the following 3 events :

- E = "R receives its h packets from its block";
- E_1 = "R receives at least b packets from its block and thus can recover all its h packets";
- E_2 = "R receives less than b packets but receives all its h packets "

We have :

$$p(E_1) = \sum_{l=b}^{b+r} C_{b+r}^{l} (1-p)^l p^{b+r-l} \quad (4)$$

$$p(E_2) = (1-p)^h \sum_{l=h+1}^{b-1} C_{l-h}^{b+r-h} (1-p)^{l-h} p^{b+r-l} \quad (5)$$

Since E_1 and E_2 are disjoint events we get $p(E) = p(E_1) + p(E_2)$. Assuming that the packet loss probability per member (p) is independent and identical for each member, the key server chooses the lowest value r satisfying the following inequality :

$$\sum_{l=0}^{(1-\alpha)N_p} C_{N_p}^{l} (1-p(E))^l p(E)^{N_p-l} \geq \beta \quad (6)$$

Since some keys inherently are replicated in other blocks and their degree of replication depends on their localization in the key tree, the probability that a member receives all required keying material is even higher than $p(E)$.

4.1.4. Example : In order to compare the rekeying cost using different techniques, we assume that $N_p = 65536$ and $p = 0.1$. The key server needs to ensure that 99.99% of **permanent** members receive their keys with a probability greater than 99.99%. We then have $\alpha = \beta = 0.9999$. Table 1 gives a comparison of the rekeying cost in different implementations. The WKA-BKR protocol [6] 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. We conclude that our hybrid scheme outperforms the other reliability schemes even when the block size is the smallest one (ie. with $i = 1$). We also realize that the rekeying cost decreases when the block size increases. However, in order to avoid excessive buffering at members, b needs to be chosen as small as possible.

Table 1. Rekeying cost comparison

Rekeying cost of different schemes when $N = 65536, p = 0.1, \alpha = 99.99\%, \beta = 99.99\%$						
Initial cost	Retransmission t = 5	WKA-BKR	Our scheme			
			b(1)=10	b(2)=25	b(3)=88	b(4)=343
70996	354980	311640	311296	155648	116738	105216

4.2. Determining T_v, T_p and w

In this section, we evaluate possible values for T_v, T_p and w.

To evaluate T_v, the key server computes the average number of members leaving 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.

Based on the results in [10], since members are assumed to leave the group either for a very short period after their arrival or at the end of the multicast session, for each real category, membership duration can be represented by an exponential distribution where the mean duration of membership for short-duration and long-duration members are denoted by M_s and M_l respectively. The ratio of short-duration members over N, the total group size, is denoted by γ. In the sequel of this section, we assume that the system is in a steady state.

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 :

$$L_v = \gamma N(1 - e^{-T_v/M_s}) + (1-\gamma)N(1 - e^{-T_v/M_l}) \quad (7)$$

The reader can refer to [5] for the computation of the average rekeying cost. Let $cost(L_v)$ be this cost based on L_v[2] and the overhead of packets ensuring reliability . T_v must then satisfy the following inequality where B is the necessary bandwidth only reserved for the rekeying operation :

$$cost(L_v) < B \times T_v \quad (8)$$

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 computation of the worst rekeying cost is explained in section 4.1.3. Let $cost(L_p)$ be this cost. Given the existing bandwidth of the network, $T_p = kT_v$ must follow the following inequality which again yields a lower bound on T_p:

$$k \times cost(L_v) + cost(L_p) < B \times T_p \quad (9)$$

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[3] :

$$N_p = k\gamma N e^{-w/M_s} + k(1-\gamma)N e^{-w/M_l} \quad (10)$$

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

4.3. 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. We set the block size b = 22. 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 :

$$\begin{aligned} T_v &\geq 46s \\ T_p &\geq 3726s \end{aligned}$$

Table 2 compares the computed minimum value of T_p with other schemes implemented in the same conditions. We realize that no matter the block size is, our scheme outperforms others in terms of security also. Hence, T_p should be as small as possible in order to let the minimum extra-time to leaving members.

Table 2. Comparison on T_p

Minimum values for T_p when $N = 65536$, $p = 0.1$, $\alpha = 99.99\%$, $\beta = 99.99\%$					
Retransmission t = 5	WKA-BKR	Our scheme			
		b(1)=10	b(2)=25	b(3)=88	b(4)=343
8418s	7452s	4600s	3726s	2806s	2530s

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 group rekeying solutions 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 alike from the loss rekey packets 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 as **permanent**. In order to offer higher reliability to **permanent** members, independently of the number of leaving members, the key server evaluates the overhead of rekeying packets ensuring reliability with the proposed hybrid reliability scheme. Based on this computation, it then adjusts the rekeying intervals T_v and T_p of the **permanent** and **volatile** member sets, respectively. Moreover, to keep the partitioning of members as perceived by the key server as close as

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 Here, N_p corresponds to the number of members who didn't 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 : $p(t \geq w) = e^{-w/M}$.

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.