Performance Comparison of Stateful and Stateless Group Rekeying Algorithms[1]

Weifeng Chen **Lakshminath R. Dondeti** **Ye Sun**

California Univ. of Pennsylvania, California, PA, USA
Qualcomm, San Diego, CA, USA
West Virginia University, Morgantown, WV, USA

Summary

Scalable group rekeying schemes proposed in the literature can be classified into two categories: stateful schemes, e.g., logical key hierarchy (LKH) based approaches, and stateless schemes, e.g., subset difference based member revocation (SDR) mechanism. They differ mainly on the interdependency of rekey messages and messaging overhead in rekeying. SDR messaging overhead in rekeying is dependent on the membership during an entire multicast session whereas LKH messaging overhead is dependent on membership of the group during a rekeying instance.

In this paper, we study the advantages and applicability of stateful and stateless rekeying algorithms to different groups and multicast security applications. We analytically compare the storage cost and the rekeying cost (number of encrypted keys) of LKH and SDR in immediate and batch rekeying scenarios. Our simulation studies show that LKH performs better in immediate rekeying and small batch rekeying, whereas stateless rekeying performs better as we process membership changes in larger batches. In some cases, stateless rekeying is observed to be as inefficient as encrypting the group key separately for each member of the group. We also report on the effect of member adjacency on SDR rekeying cost that it seems to have more impact on rekeying cost than the number of membership changes. We further show that the analysis of SDR rekeying cost in [12] is incomplete and present a better result.

Key words:

Network security, Multicast security, Group rekeying, Stateless rekeying

1. Introduction

The dramatic growth of profitable services on the Internet, such as Pay-Per-View, stock quote distribution, has been benefited from the successful deployment of secure communication. The IETF multicast security working group identified three problem areas in secure group communication, viz., key distribution, data origin authentication, and policy management. Group key distribution facilitates confidential group communication as well as enforcement of access control. A *group key* k_g, thus, is introduced to encrypt group data. Strict group privacy requires that a host be able to decrypt group communications only when it is a member of the group. Thus, when a new member joins, k_g needs to be updated in order to prevent the new member from accessing past communications in the group. This property is known as backward access control. The group key also needs to be changed to prevent departing members from accessing future data and enforce forward access control. The process of generating and distributing a new group key is referred to as a rekeying instance. Although it is easy to enforce backward access control – by multicasting the new group key, k'_g, encrypted using the current k_g – it is difficult to ensure forward access control. Typically, we need to encrypt k'_g with a common key (or set of keys) known only to the remaining membership. Many algorithms [4, 9, 10, 12, 17] have been proposed for scalable rekeying of large groups (in the order of thousands or more of members). Several surveys of group rekeying algorithms are available in the literature [11, 13]. In the rest of this paper, we will refer to an entity called *Group Controller and Key Server* (GCKS), by the IETF MSEC framework for dealing with key distribution and rekeying.

Based on the interdependency of rekeying messages, group key management algorithms can be classified into stateful schemes ([4, 9, 17]) and stateless schemes ([12, 15]). Under a stateful scheme, the GCKS uses key encryption keys (KEKs) sent in a given rekeying instance to encrypt the keys to be sent in the next (or future) rekeying instance(s). Rekeying algorithms based on a logical hierarchy of KEKs (LKH) fall into this category. In stateless rekeying, rekey messages are encrypted only by the keys distributed during member registration. This allows rekey messages to be independent of each other. Naor, *et al.* described in [12] two stateless rekeying schemes: Complete Subtree Revocation (CSR) and Subset Difference Revocation (SDR). SDR is the more efficient approach among these two. Stateless rekeying comes at a

[1] Part of this work appeared in the proceedings of the Fourth International Workshop on Networked Group Communication (NGC 2002).

Manuscript received March 5, 2008
Manuscript revised March 20, 2008

cost however. More specifically, rekeying cost in SDR could be more than that in LKH. In this paper, rekeying cost is measured by the number of encrypted keys during a rekeying instance.

Analytical cost comparison of these two schemes has been done elsewhere [12], but an important difference between the two schemes was overlooked in that study. LKH rekeying is based on instantaneous membership in the group, whereas SDR rekeying is based on total membership of the group during the entire secure multicast session, and the revoked member set at the time of rekeying. These differences are highlighted in this paper.

Considering that rekeying cost in LKH and SDR is dependent on members' positions in the key trees, and on the key tree sizes, we use simulation for the comparison study. We use both real-life group membership data and DaSSF [6] generated data. Our simulation studies show the following important and interesting results. The SDR appears at first glance to be more efficient than LKH since its (asymptotic) communication costs are lower than those of LKH. But we demonstrate that the reality can be quite different, and LKH can outperform SDR in many scenarios. In particularly, LKH has a smaller rekeying cost than SDR in immediate rekeying and in small batch rekeying. When the batch size increases, SDR outperforms LKH. We also show that the result in [12] about SDR rekeying cost is incomplete. Instead, we present a better result that takes member adjacency into account. The present work helps us understand the respective advantages of these two schemes, and provides guidelines for applications in choosing the appropriate rekeying scheme according to the group membership behavior and the rekeying policy.

The rest of this paper is organized as follows. In Section 2 we briefly overview LKH and SDR. We present an analytical comparison between LKH and SDR in Section 3. Section 4 describes our implementation and simulation. Section 5 concludes the paper.

2. LKH and SDR

This section briefly presents necessary background of the Logical Key Hierarchy (LKH) and Subset Difference Revocation (SDR). In LKH [16, 17], the GCKS organizes the keys in a logical hierarchy. The root node of the key tree represents the group key and each of other nodes corresponds to a KEK. Each active member in the group is assigned a leaf node and receives all the KEKs of nodes along the path from the assigned leaf to the root. Consequently, the group key is known to all active members and a leaf node represents the unique key a member shares with the GCKS. When a member joins or leaves the group, in order to keep the backward/forward access control, all the keys in its path to the root must be changed. Each of the updated key is further encrypted by the KEKs corresponding to the node's children and sent out.

Similar to LKH, SDR constructs a binary key tree and each node in the binary tree corresponds to a key. During member registration, each member is also assigned a leaf node, whose position determines the set of secret information distributed to the member. During a rekeying instance, members to remain in the group are divided into a set of subsets. Each subset has a key and the new group key is sent encrypted with the keys of the resultant subsets. The keys of the resultant subsets can be computed from the secret information received during registration. SDR is a stateless scheme because the secret information is determined by the position of the leaf node assigned to a member, and keeps unchanged during rekeying.

2.1 Key tree size

We begin with an important key tree size difference between LKH and SDR that is overlooked in a quick comparison in [12].

Due to the fact that all the keys known to (or to be known to) departing (or joining) members are changed during rekeying, LKH allows reassignment of an empty tree node position left by a departing member, to a joining member. Additionally, recalling that LKH rekeying cost is determined by the height of the key tree, it is efficient to dynamically contract or expand the key tree to just hold current members in the group. Some algorithms have been proposed to dynamically keep the tree full and balanced [5, 18].

However, each leaf node in the SDR key tree cannot be assigned to more than one member, since the secret information associated with leaf nodes is unchanged. The SDR key tree is thus larger than the LKH key tree. In particular, the SDR key tree should be large enough to hold all potential members. Typically, the size of all potential members is estimated and then the SDR key tree is constructed in advance. The number of potential members is strongly related to the length of the multicast session, e.g., a military battle or a baseball game on Pay-Per-View. Generally speaking, the longer the session is, the larger the size of potential members. Thus, when we compare the performance of these two schemes, we consider a particular multicast session assuming that the size of potential members of the session is estimated correctly. When the multicast session is given, the difference between the size of a LKH key tree and a SDR key tree obviously depends on the relationship between the number of potential members and concurrent members. This relationship is application-specific, as we will see in this paper.

2.2 Symbols used in the paper

This section presents the notation necessary for our further discussion.

- N: the set of potential members in the given multicast session. N includes all the members who have ever joined the group during the session. This is only used in SDR. $N = |\ N\ |$.
- R: the set of members to stay in the group after the rekeying, referred to as remaining members. $R = |R|$.
- J: the set of joining members since last rekeying. It may be empty in the case when there are no members joining the group since last rekeying. $J = |J|$.
- D: the set of departing members since last rekeying. Similarly, D may be empty. $D = |D|$.
- M: the set of members need to be considered at the time when the GCKS is going to perform rekeying. M consists of remaining members who will get the new group key and departing members who should not know the new group key, i.e., $M = R \cup D$. $M = |M|$.
- Q: all potential members except the remaining members, i.e. $Q = N \setminus R$. Q is only used in SDR. $Q = |Q|$.
- $\{K\}_k$: K is encrypted with k.

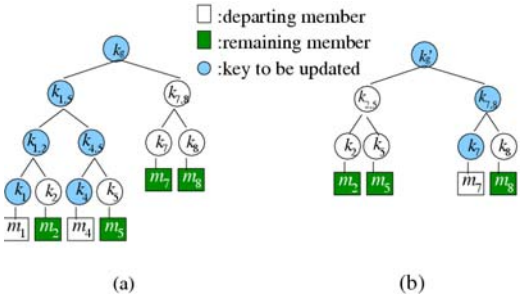


Fig. 1: Stateful rekeying of LKH.

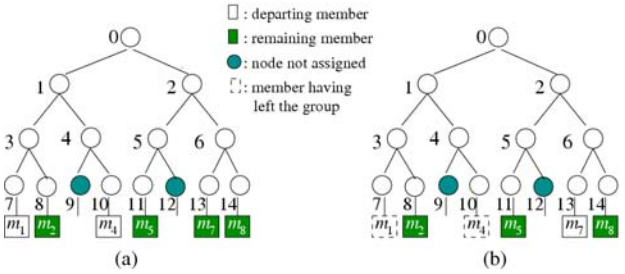


Fig. 2 Stateless rekeying of SDR.

Figures 1 and 2 show a LKH and an SDR key tree respectively in two consecutive rekeying instances[2], where $N = \{m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8\}$. Specifically, in Figure 1(b) and 2(b), $M = \{m_2, m_5, m_7, m_8\}$, $R = \{m_2, m_5, m_8\}$, $J = \phi$, $D = \{m_7\}$ and $Q = \{m_1, m_3, m_5, m_6, m_7\}$. As we will see, the performance of SDR is dependent on N and R whereas that of LKH is dependent on M, J and D. One can see that the LKH key tree can be dynamically adjust to hold M members, whereas the SDR key tree has a fixed size of N. It should be noted that the relationship between N and M is application dependent. To get an idea for the comparison of N and M, consider that N would represent the total number of TVs tuned to any part of a Super Bowl broadcast, whereas M would represent the number of TVs tuned to a given part of the broadcast.

[2] The degree of the LKH key tree may be varied. We only demonstrate the binary one although we also consider other degrees in Section 4.

3. Analytical comparison

Two performance metrics are considered in our comparison between LKH and SDR: (i) key storage at both member side and the GCKS side and (ii) rekeying cost. The rekeying cost includes unicast cost (to new joining members) and multicast cost (to other members).

3.1 Member and GCKS key storage

The GCKS using LKH needs to store all the keys in the key tree (internal nodes and leaf nodes), incurring a storage cost of $2M - 1$. Each member stores all the keys along its path to the root, i.e., $\log M$ keys. Key storage of SDR can be analyzed based on [12]. Table 1 summaries the key storage comparison, where we can see that SDR is expensive in both the member storage and the GCKS storage, due to the large key tree size ($N > M$).

Table 1: Key storage of LKH and SDR.

	GCKS storage	Member storage
LKH	$2M - 1$	$\log M$
SDR	$2N\log N+2$	$(\log^2 N+\log N)/2+2$

It will be helpful to see the actual key storage requirements for GCKS and group members in practice. We list them in the next section when we have values for N and M in specific scenarios.

3.2 Rekeying cost

In this subsection, we compare the rekeying cost of LKH and SDR, including unicast cost and multicast cost. There are three approaches to organize the rekeying messages: *user-oriented, key-oriented* and *group-oriented* [17], which result in different number of rekeying messages. In order to address the differences of these three approaches and SDR rekeying, we will use the number of unit-size encrypted messages to measure the rekeying cost.

Unicast generally incurs when the GCKS needs to send information to a new member that is only shared between the GCKS and the member. Multicast generally is applied to distribute updated KEKs to multiple receivers. Consequently, the unicast cost equals to the member's key storage shown in Table 1.

It should be pointed out that the unicast cost to a joining member in SDR varies, depending on whether the joining member is a new member or a rejoining member. This variance comes from the fact that the keys associated with the SDR key tree nodes are unchanged during rekeying. Under an SDR scheme, a leaf node can only be assigned to at most one member; when a member goes offline and joins the group again, the member is typically assigned the same leaf node as before such that the SDR key tree has a size equal to the total number of unique members, rather than to the total number of joining instances. Thus if a joining member m had been in the group, the GCKS does not need to unicast any keys to m. If it is the first time that m joins the group, the unicast cost is the same as the key storage. But under a LKH scheme, every joining member, no matter new or rejoining, incurs the same amount of unicast cost equal to the key storage at member side.

Given a multicast session, the total number of rekeying instances in the whole session varies depending on the rekeying strategies. A GCKS may process membership changes immediately or in a batch. Batch rekeying is proposed to reduce the excessive overhead of immediate rekeying but with loss of strict access control [7, 14]. Two types of thresholds can be used for batch rekeying. In the first, the GCKS conducts rekeying after a fixed time period and in the second, the GCKS rekeys after a fixed number of members have left the group since last rekeying. We refer to these variations as *periodic batch rekeying* and *membership batch rekeying*, respectively.

Work in [7, 17] provides an analysis of the multicast cost of LKH rekeying. In LKH immediate rekeying, only the keys along the path from the departing/joining member to the root are updated. Each of the updated key is multicast encrypted by the KEKs corresponding to the node's children, resulting in $2\log M$ multicast cost. Batch rekeying cost using LKH depends on the relationship between the number of joining members (J) and the number of departing members (D) during a batch. More details can be found in [7].

The multicast cost of SDR equals to the number of resultant subsets divided by remaining members R since the updated group key is multicast encrypted using the keys of the resultant subsets. Naor *et al.* pointed out in [12] that the SDR multicast cost is $2Q - 1$ in the worst case and $1.38Q$ in average. This statement is not complete since it overlooks the relationship between N and Q. Since subset $S_{u,v}=S_u \backslash S_v$ ={descendants of node u} − {descendants of node v}, at least one member in R and one member in Q are required to generate $S_{u,v}$. If either R or Q is small, R is divided into fewer subsets. Consequently, the multicast cost should be determined by min(R, Q), achieving the highest value when R equals to N=2.

The preceding comparison is summarized in Table 2, where we do not include the cost of batch rekeying due to the complicated expression for LKH batch multicast cost detailed in [7]. As for the batch SDR multicast cost, we conjecture that it is similar with the immediate one since SDR is stateless. Such conjecture is verified by our simulation described in the next section. We also use simulation to do more detailed comparison on the batch multicast cost of LKH and SDR.

Table 2: Rekeying cost of LKH and SDR.

	Unicast cost	Immediate multicast cost
LKH	$\log M$	$2\log M$
SDR	$0^{[1]}$ or $(\log^2 N+\log N)/2+2^{[2]}$	$O(\min(R, Q))$

*[1]: to rejoining members. [2]: to new members

As a final comment, we regard the multicast cost as the critical cost of the rekeying cost for the following two reasons. First, multicast accounts for most of the rekeying traffic. Second, multicast cost is more expensive than unicast cost with respect to the number of links that a message travels.

4. Comparison using simulation

In this section, we first verify the relationship of the SDR rekeying cost on N and R. We then use both real-life data and simulated data to compare the rekeying cost of LKH and SDR in different scenarios, which particularly includes immediate rekeying, periodic batch rekeying and membership batch rekeying.

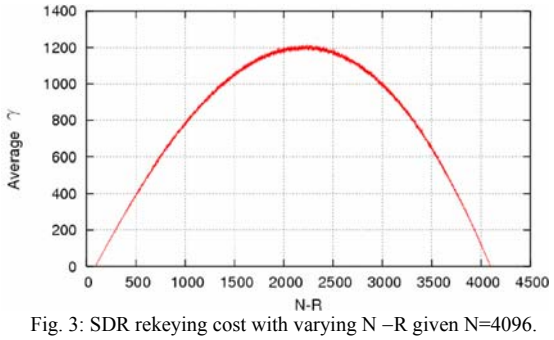
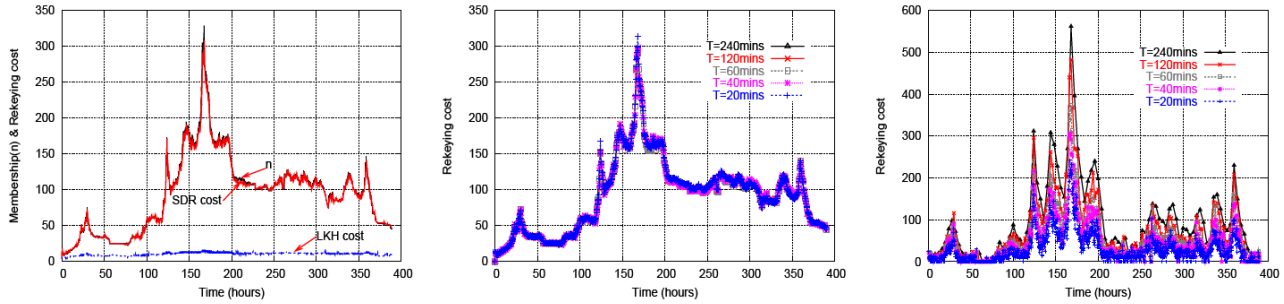


Fig. 3: SDR rekeying cost with varying N –R given N=4096.

4.1 SDR rekeying cost with N and R

We assume a fixed N = 4096 and construct a balanced SDR key tree with 4096 leaf nodes. We then vary the number of the remaining members (R) and calculate the rekeying cost, which is equal to the number (γ) of resultant subsets of these R members. For a given R, we simulate 100 instances, in each of which these R remaining



(a) Immediate rekeying cost. (b) Periodic batch rekeying cost using SDR. (c) Periodic batch rekeying cost using LKH.

Fig. 4: Rekeying costs of LKH and SDR for NASA STS-71 Session with various rekeying strategies.

members are randomly assigned the leaf nodes independently. An average number of γ is then calculated on the 100 instances for the corresponding R. Figure 3 shows the result. The figure confirms our conjecture that if either R or $N–R$ is small, γ is also small; γ achieves the highest value when $N – R = R = N/2$.

4.2 Real-life MBone simulation

The real-life data comes from Multicast Backbone (MBone) information collected by Almeroth *et. al.*, including NASA STS-71, NASA STS-65, IMS, IPNG and UCB, total five sessions [2]. For a particular session, the collected data includes the inter-arrival time between two consecutive members joining the session and the duration time that a member staying in the session. Based on these data, we can obtain the membership. We then simulate rekeying on the membership change using LKH and SDR separately. Simulation results of the five sessions were similar and we present the largest session, named STS-71 [1].

According to [2], during the whole session, there are around 4000 unique hosts joined the STS-71 session. We then choose N = 4096 as the number of all potential members for this particular secure multicast session. When SDR is used, we construct a fixed binary key tree with 4096 leaf nodes and randomly assign a leaf node to a new joining member. If a joining member had been in the group[3], the member is assigned the same leaf node as the one assigned before. If LKH is the rekeying mechanism, we dynamically adjust the binary key tree based on the algorithm in [5] to hold the currently active members `M` in the multicast session.

We first apply the immediate rekeying on STS-71 session, using LKH and SDR separately. Figure 4(a) shows the result, from which one can see that immediate rekeying cost using LKH is much smaller than the one using SDR. More specifically, the average LKH rekeying cost was 12.15 and the average height of the LKH key tree was 7.78 because of the dynamic adjustment. This result is consistent with Table 2. The average SDR rekeying cost was 138.98, which is comparable to the average number of the active members. The curves of the membership (M) and the SDR rekeying cost coincide in the figure. This is because $Q = N – R$ is very close to N in this scenario. The R remaining members are dispersed in the key tree and most of resultant subsets only cover one remaining member. In this case, SDR rekeying cost is very close to R.

Figures 4(b) and 4(c) show the rekeying cost of SDR and LKH in periodic batch rekeying. Each figure includes the costs for five different batch periods (T). From the figures, we observe that SDR rekeying cost remains almost the same in the five different batch periods, while LKH rekeying cost increases as T increases.

Batch rekeying was proposed to reduce the rekeying cost [7, 14]. To clear the confusion that in LKH, batch rekeying cost (Figure 4(c)) looks much higher than immediate rekeying cost (Figure 4(a)), it should be noted that the batch rekeying cost is the cost for the corresponding batch period, whereas, the immediate rekeying cost is instantaneous. If we add up all the immediate rekeying cost associated in a batch period, the sum is higher than the corresponding batch cost.

4.2.1 Explanation

To better understand the above results, we introduce the following metrics of the two schemes.

We use the height of node u in the key tree to represent the height of a subset $S_{u,v}$, denoted as $H_{u,v}$. Generally speaking, the higher H is, the more members in `R` the subset covers. A subset $S_{u,v}$ with height one only includes one remaining member while excluding one member in `Q`, as $S_{3,7} = \{m_2\}$ shown in Figure 2(a). Specifically a subset with height H covers at least 2^{H-1} members. We use $\overline{H}_S$ to denote the average value of the subset heights.

[3] For privacy, the data we got does not have the identities of the hosts. We randomly assign each host an ID such that we can recognize the rejoining according to the ID.

Table 3: Average metrics of SDR and LKH with periodic batch rekeying in NASA STS-71 Session (n_{max} = 329)

T (mins)	$\overline{Q}$	$\overline{\gamma}$	$\overline{H}_S$	$\overline{\alpha}_R$	$\overline{\alpha}_Q$	$\overline{M}$	$\overline{C}$	$\overline{H}_L$	$\overline{J}$	$\overline{D}$
20	93.90	92.65	1.02	1.03	2.07	99.74	31.17	7.56	5.85	5.81
40	93.88	92.64	1.02	1.03	2.07	105.59	45.15	7.68	11.69	11.61
60	93.77	92.52	1.02	1.03	2.07	111.34	54.7	7.68	17.52	17.40
120	93.87	92.63	1.02	1.03	2.07	129.09	76.96	7.41	34.95	34.72
240	93.58	92.28	1.02	1.03	2.07	164.35	98.65	7.68	69.88	69.37

We also define adjacent degree to denote the adjacency of the leaf nodes. Consider the leaf nodes in the SDR key tree in any rekeying instance. There are two kinds of leaf nodes, corresponding to remaining members R and others Q respectively. Adjacent nodes of the same kind, corresponding to R or Q, form a *block*. The leaf nodes are thus composed of alternate blocks. A block is associated with an adjacent degree, defined as the length of the block, i.e., the number of nodes in the block. Note that for a block with a single node, the corresponding adjacent degree is one. We then define α_R as the average adjacent degree of the blocks corresponding to remaining members R. Similarly, we have the definition of α_Q. For example, the leaf nodes in the SDR key tree in Figure 5 form four blocks of R. The adjacent degrees of these four blocks are four ($\{m_1, m_2, m_3, m_4\}$), one ($\{m_6\}$), three ($\{m_8, m_9, m_{10}\}$) and one ($\{m_{13}\}$), resulting in α_R = 9/4 = 2.25. Similarly, α_Q equals to 1.75 in this example. In general, a small α_R implies that the remaining members are dispersed in the key tree and more subsets are generated to cover R. On the other hand, a larger α_R indicates that the remaining members may be grouped, incurring fewer subsets.

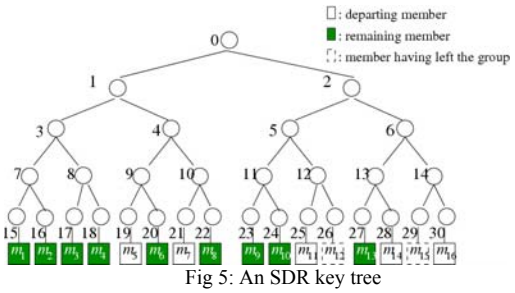


Fig 5: An SDR key tree

Table 3 shows the average metrics of the periodic batch rekeying in NASA STS-71 Session. Both the average subset height $\overline{H}_S$ and adjacent degree $\overline{\alpha}_R$ are close to one, indicating that most subsets contain only one remaining member. This is reasonable since around 300 remaining members are likely to be dispersed in the key tree with 4096 leaf nodes. In Table 3, $\overline{C}$ is the average rekeying cost using LKH. $\overline{H}_L$ is the average value of the heights of the LKH key tree conducting dynamic contraction. J and D are the number of joining members and departing members during a batch period T, respectively. From the result in [7], LKH rekeying cost increases as J and D increase.

From Figures 4(a), 4(b), 4(c) and Table 3, we can see that LKH performs better in immediate rekeying (J = 1 or D = 1) and small batch rekeying (T is small). But as T increases, SDR rekeying begins to perform better than LKH. When T = 240 minutes, the average LKH rekeying cost is higher than that in SDR. Finally, we observe that LKH rekeying cost is sensitive to T while SDR cost is insensitive. Such results are more obvious if we conduct membership batch rekeying shown next.

4.2.2 Cross-over point in membership batch rekeying

When membership batch rekeying is conducted, the GCKS sets a threshold Γ and performs rekeying when it receives Γ departing requests (i.e., Γ members depart the group). We vary the threshold Γ from 5 to 100. For a given Γ, the GCKS conducts membership batch rekeying with the particular Γ throughout the whole STS-71 session, and then the average rekeying cost is calculated for that Γ. Figure 6 presents the result. It is interesting to observe from the figure that the LKH rekeying cost monotonously increases with Γ, whereas the SDR rekeying cost keeps stable, demonstrating a *cross-over point* of LKH to SDR graphs. Thus it is clear from the result that LKH performs better for immediate rekeying as well as small batch rekeying whereas SDR rekeying is better for large batch rekeying. Note that the cross-over point of LKH with a degree of two to SDR incurs when batch size Γ equals to 55. We also observe from Figure 6 that LKH achieves the smallest rekeying cost with a degree of four. This has been reported earlier in the literature [17].

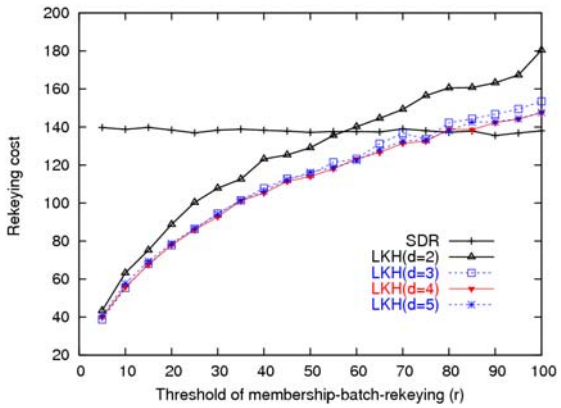


Fig. 6: LKH and SDR rekeying cost for membership batch rekeying in NASA STS-71 Session.

4.2.3 Actual key storage

Finally, we calculate the key storage requirements of LKH and SDR in STS-71 session. Considering 3DES for encryption, we need 128-bit or 16B keys. Due to the fact that in LKH, key storage at the GCKS and group members changes with M, we compute the average values. Using the results in Table 1, the average key storage is 90.83 bytes at member side and 3002.90 bytes at the GCKS side. In SDR, both GCKS and group members have fixed size key storage, which are 1572864 bytes and 1232 bytes respectively given that N = 4096. Thus we can see that stateless schemes require much more key storage than LKH, at both the GCKS side and the member side.

4.3 DaSSF simulation

As we have seen in the NASA STS-71 session, the number of current members, M, is much smaller than the number of potential members, N. It will be desirable to compare the performance of LKH and SDR in the scenario when M is more significant. In this subsection, we use DaSSF [6] to generate such groups.

Using DaSSF, we can simulate members' activities. Particularly, in the model we consider, the group is empty initially. Each potential member waits for a randomly distributed waiting time, ω, before he/she joins the group, and consequently stays in the group for another randomly distributed duration time, δ, before he/she leaves the group. All potential members repeat this process till the end of the multicast session. By specifying different distributions of ω and δ, we can obtain different group behaviors.

We first use DaSSF to further explore the effect of the member adjacency to the SDR rekeying cost, and then compare the rekeying cost of LKH and SDR in the simulated group with a larger number (M) of current members.

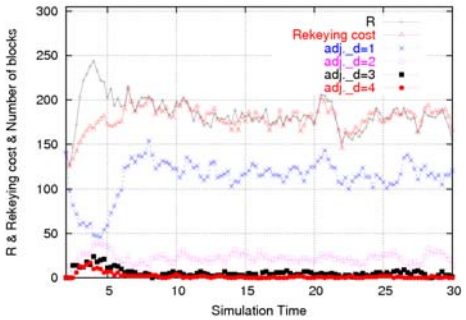


Fig. 7: SDR rekeying cost and average adjacent degree

4.3.1 Effect of member adjacency

We simulate a group with N = 1024, where ω is exponentially distributed with mean 1, i.e., each member waits for an average time of 1 unit of simulation time before he/she joins the group, and δ is a distribution of Pareto [3, 8]. We then conduct SDR periodic batch rekeying on this group with T = 0.25 unit of simulation time. Figure 7 shows that the SDR rekeying cost mostly coincides with Q. This is because $Q \approx 200$, much smaller than N. At any rekeying instance, we also calculate the number of the blocks of nodes corresponding to members in Q with 4 different adjacent degrees: one, two, three and four, which are also shown in Figure 7. Blocks with adjacent degree greater than four is rare and we do not include in the figure. From the figure, we observe that rekeying cost reduces if the number of blocks with large adjacent degree increases, i.e., members in Q are more adjacent in the key tree. In particular, from simulation time zero to eight, the number of blocks with adjacent degree one is relatively small, and the number of blocks with adjacent degree three and four is much larger than the rest of simulation period. The larger number of blocks with adjacent degree three and four in this period indicates that members in Q are more adjacent in the key tree, incurring a rekeying cost smaller than R. In the rest of the simulation period from time eight to 30, blocks with adjacent degree one dominates all the blocks, indicating that members in Q are mostly separate in the key tree. As a consequence, each resultant subset only excludes one member in Q and the rekeying cost is very close to Q.

Generally speaking, when members in R or Q are more adjacent, the number of resultant subsets and then the SDR rekeying cost is reduced.

4.3.2 Large simulated group behavior

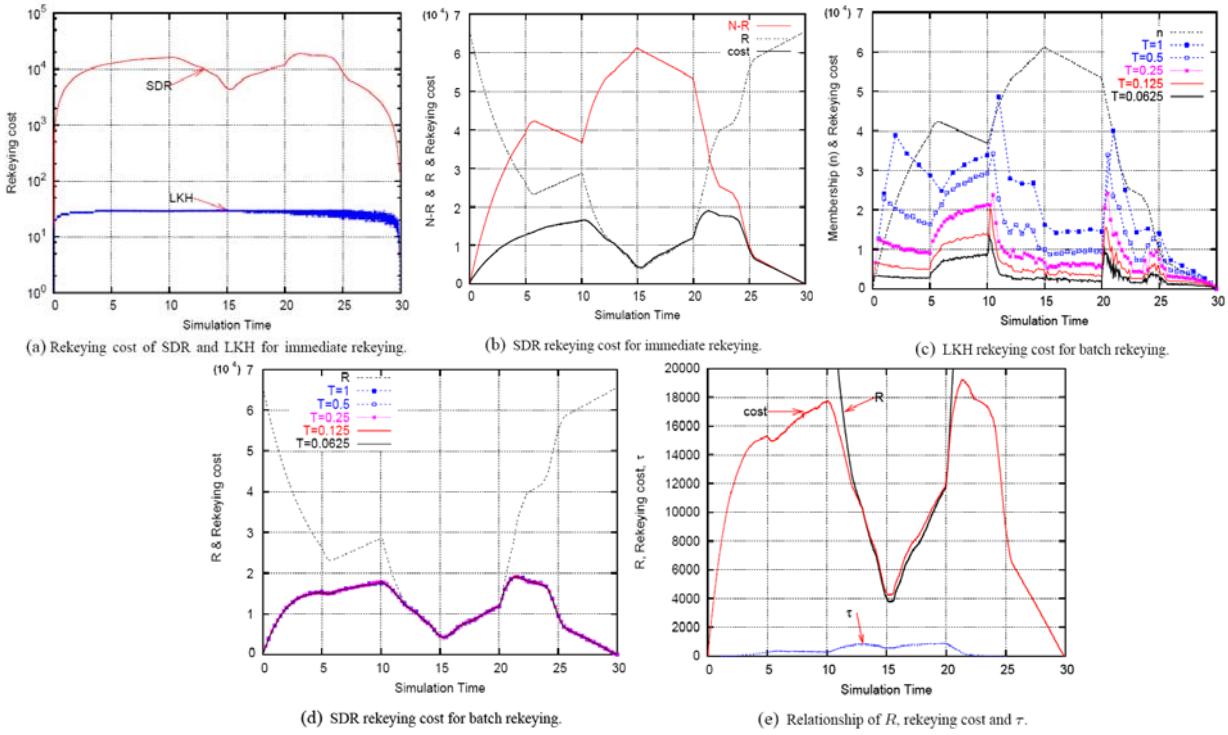
If we consider the ratio of the SDR rekeying cost (γ) to the number of remaining members (R), SDR is not efficient at all in NASA STS-71 session i.e., $\gamma/R \approx 1$. This means that SDR rekeying is as expensive as encrypting the updated group key separately for each remaining member. That is mainly due to R being much smaller than N. We notice that in some real applications, the number of remaining members R can be much larger than Q. For instance, consider the audience of pay-per-view events, or a football game. Most people join at some point during the game and many (around half the audience) tune in from start to finish. During the session, many people join and leave, while most start leaving towards the last quarter of the lifetime of the group. Such groups are larger in size than the NASA STS-71 session. This motivates us to simulate a larger group using DaSSF.

We assume a larger size of potential members, $N = 2^{16}$. By adjusting members' waiting time (ω) and duration time (δ) in DaSSF, we simulated a group with membership (M) shown as the top curve in Figure 8(c). Notice that M is much larger compared to the STS-71 session, and more importantly it is closer to N. This membership has a peak

Table 4: Average metrics of SDR and LKH with periodic batch rekeying in DaSSF simulation with $N = 64K$ ($n_{max} = 61806$).

T [1]	$\overline{Q}$	$\overline{\gamma}$	$\overline{\alpha}_R$	$\overline{\alpha}_Q$	$\overline{M}$	$\overline{C}$	$\overline{H}_L$	$\overline{J} = \overline{D}$ [2]
0.0625	34514.52	11226.77	4.13	1.28	34929.22	3506.51	15.63	369.48
0.125	34441.98	11227.00	4.11	1.27	35249.88	5932.27	15.65	738.97
0.25	34585.75	11256.74	4.13	1.27	36117.34	9548.78	15.66	1477.93
0.5	34003.56	11092.56	4.04	1.24	36800.26	14868.53	15.70	2955.87
1	34550.70	11243.27	4.10	1.24	39467.87	22008.26	15.73	5911.73

*[1]: Units of the batch period compared to the 30-unit of total simulation time.
[2]: The simulation starts with $n = 0$ and ends with $n = 0$, thus the total number of joining members equals to the total number of departing members during the whole simulation.



(a) Rekeying cost of SDR and LKH for immediate rekeying.
(b) SDR rekeying cost for immediate rekeying.
(c) LKH rekeying cost for batch rekeying.
(d) SDR rekeying cost for batch rekeying.
(e) Relationship of R, rekeying cost and τ.

Fig. 8: Experiment results for the simulated group with $N = 64K$.

(almost all the potential members are in the group) at simulation time 15 and the average number of active members through the whole session exceeds half of N. This simulated membership provides us a group that behaves differently from the STS-71 session. We investigate the relative advantages of LKH and SDR rekeying for this group.

Figure 8(a) shows the immediate rekeying cost of SDR and LKH in the simulated group. Compared to the cost of SDR, LKH immediate rekeying is much less so we plot the rekeying cost in log-scale. Similar to Figure 4(a), LKH rekeying cost for immediate rekeying is related with the height of the key tree. Whereas, SDR rekeying cost is much different from the one in Figure 4(a). To be clear, we plot the SDR cost separately in Figure 8(b), where we include the curve of R to show that SDR rekeying cost is related to N and R. It is interesting to note that although R increases in period from simulation time 10 to 15, SDR rekeying cost decreases. The reason is that $Q = N - R$ decreases in this period and $Q < R$. Recall that a subset $S_{u,v}$ not only needs to cover members in R (descendants of node u but not of v), but also excludes some members in Q (descendants of node v). Thus fewer subsets are needed when Q decreases. This result also confirms our conjecture that SDR rekeying cost is determined by $\min(N-R, R)$ (see Table 2).

We plot the batch rekeying cost of LKH and SDR in the simulated session in Figure 8(c) and 8(d) respectively, each of which includes the costs of five different batch periods (T). Again, while LKH rekeying cost increases as T increases, SDR rekeying cost stays nearly the same for different T, much less than R. Numerically, $\gamma/R \approx 1/3$. Thus compared to STS-71 session, SDR rekeying is more efficient in the group with a larger R. We also observe that in this simulated group, SDR rekeying cost for immediate rekeying is almost the same as the one for batch rekeying,

which is also observed in STS-71 session (Figure 4(a) and 4(b)). The reason is the stateless property of SDR. More specifically, the SDR rekeying cost at a given point of time only depends on the positions of remaining members in the SDR key tree at that time, no matter what rekeying mechanism (e.g., immediate rekeying or batch rekeying) is used. This statelessness results in that SDR performs more efficient as the batch size increases. However, the larger batch size is, the more information exposed to unauthorized members. This issue is addressed in the exposure-oriented rekeying [19].

Table 4 shows the average metrics for the two schemes, which allows us to take a closer look at how these algorithms perform. There is also a cross over point where SDR begins to perform better than LKH, in the simulated group. Note that for batch period $T = 0.5$, LKH rekeying cost is higher than that in SDR.

4.3.3 Actual key storage

Perhaps unsurprisingly, the large storage cost becomes more pronounced in the larger group. LKH key storage required in the simulation scenario for GCKS and members are $1030K$ bytes and 228 bytes respectively. Whereas, the key storage of GCKS and group members in SDR are fixed to $32768K$ bytes and 2176 bytes respectively, given $N = 64K$.

4.4 Discussion

The simulation results above based on both MBone STS-71 session and DaSSF simulated group show that SDR rekeying cost is much higher than LKH cost in immediate rekeying and small batch rekeying, whereas SDR outperforms LKH when the batch rekeying period increases.

Now let us focus on the relationship between γ and $\min(R, N-R)$. Since each member in R is covered by one and exactly one subset, it is obvious that the number of resultant subset $\gamma \le R$. However, if $Q \le R$, γ may be larger than Q. For example, consider the subtree rooted at node 1 in Figure 5, where $Q = 2$, $R = 6$, and the number of subsets incurred by this subtree is three, e.g., $S_{1,4}$, $S_{9,19}$ and $S_{10,21}$. This is also the worst case of γ where $\gamma = 2Q - 1$ [12]. We show that if $\gamma > Q$, the following subtree T in the SDR key tree must exist when rekeying. Let u be the root of T. T satisfies the following three conditions:

(1) u has a child v_1. All the members attached to the descendants of v_1 are in R;

(2) Among all the members attached to the descendants of the other child v_2 of u, at least two but not all are in Q;

(3) Any two members in Q attached to the descendants of v_2 are NOT siblings.

Note that in such a subtree T, the number of subsets incurred by T is one more than the number of leaf nodes of T that correspond to members in Q.

In Figure 8(a), we can see that from simulation time 13 to 20, γ is bigger than Q. We calculate the number of the subtrees T satisfying the above conditions, denote as τ, and plot it in Figure 8(e). We can see that τ is much higher in the period from simulation time 13 to 20 than in other periods.

Then we observe that SDR rekeying cost (number of subsets) approximately equals to

$$\begin{cases} \dfrac{R}{\alpha_R}, & \text{if } R \le Q \\ \dfrac{Q}{\alpha_Q} + \tau, & \text{if } Q < R \end{cases} \tag{2}$$

From Equation (2), it follows that the adjacency of members in R or Q directly affects rekeying cost in SDR. In our simulation, we assigned members to positions in the key tree randomly. In real world deployments it may be more efficient to assign members based on their join and departure behavior. For example, note that members from a time zone may join and leave a conference call around the same time. Therefore, SDR rekeying would be efficient if all members within a time zone (or from an office location) are assigned to neighboring positions within the key tree.

Finally, the SDR rekeying cost at a given point of time is independent on the rekeying mechanisms, e.g., immediate rekeying or batch rekeying. It only depends on the positions of members in R in the SDR key tree when the GCKS conducts rekeying. The positions of members affect the SDR rekeying cost through the member adjacency, as Equation (2) states.

5. Conclusion

In this paper, we use simulation to verify analytical comparison between logic key hierarchy based group rekeying scheme and a stateless scheme — Subset Difference Revocation scheme (SDR). Our simulation studies show that LKH outperforms SDR in immediate rekeying and small batch rekeying. We also observed that LKH is preferred in the scenarios when the number of current members M is much smaller than the number N of potential members, and when membership variance is low. In groups where $M \approx N$, SDR performs better for batch rekeying. If $M \ll N$, SDR rekeying cost coincides with M, i.e., each of the remaining members belongs to a different subset. Although it is stated in [12] that the average SDR rekeying cost is $1.3Q$, it may be much less than Q in most cases, because of the member adjacency factor. In particular, we conclude that the SDR rekeying cost is

closely related with min(R, $N - R$) and member adjacency, as stated in Equation (2).

Acknowledgment

The authors would like to thank Professors Don Towsley and Jim Kurose at the University of Massachusetts at Amherst for their insightful comments. We also thank Professor Kevin Almeroth for sharing the MBone data he collected. This research has been supported in part by the NSF under grant awards EIA-0131886 and EIA-0080119, and by the Research Assistant Grant from John Jay College, City University of New York.

References

[1] 1995 sts-71 space shuttle mission. http://science.ksc.nasa.gov/shuttle/missions/sts-71/mission-sts-71.html.

[2] K. C. Almeroth and M. H. Ammar. Collecting and modeling the join/leave behavior of multicast group members in the MBone. In *Proceedings of the Symposium on High Performance Distributed Computing*, pages 209–216, Syracuse, NY, August 1996.

[3] V. Bolotin. Modeling call holding time distributions for ccs network design and performance analysis. *IEEE Journal on Selected Areas in Communications*, 12(3):433–438, 1994.

[4] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key management for secure internet multicast using boolean function minimization techniques. In *Proceedings IEEE Infocomm'99*, volume 2, pages 689–698, 1999. Replaced as chen's work

[5] L. Dondeti, S. Mukherjee, and A. Samal. DISEC: A distributed framework for scalable secure many-to-many communication. In *Fifth IEEE Symposium on Computers and Communications*, Antibes-Juan les Pins, France, July 3-6, 2000.

[6] Dartmouth Scalable Simulation Framework. http://www.cs.dartmouth.edu/~jasonliu/projects/ssf/.

[7] X. Li, Y. Yang, M. Gouda, and S. Lam. Batch rekeying for secure group communications. In *Proceedings of World Wide Web Conference 10 (WWW10)*, Hong Kong, May 2001.

[8] A. Marbini and L. Sacks. Considering user behaviour in active servers queue management. In *London Communications Symposium*, 2001.

[9] D. A. McGrew and A. T. Sherman. Key establishment in large dynamic groups using one-way function trees. Technical Report No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, 1998.

[10] S. Mittra. Iolus: A framework for scalable secure multicasting. In *SIGCOMM* 97, pages 277–288, Cannes, France, 1997.

[11] M. J. Moyer, J. R. Rao, and P. Rohatgi. A survey of security issues in multicast communications. *IEEE Network Magazine*, November/December 1999.

[12] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. *Lecture Notes in Computer Science*, 2139:41–62, 2001.

[13] S. Rafaeli. A decentralised architecture for group key management, September 2000.

[14] S. Jajodia S. Setia, S. Koussiah and E. Harder. Kronos: A scalable rekeying approach for secure multicast. *In IEEE Symposium on Security and Privacy*, Berkeley, CA, May 2000.

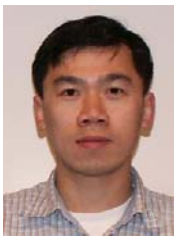
[15] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean. Self-healing key distribution with revocation. In *Proceedings of IEEE Symposium on Security and Privacy,* The Claremont Resort Oakland, CA, May 2002.

[16] D.Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. IETF, RFC 2627, http://www.faqs.org/rfcs/rfc2627.html.

[17] C. K. Wong, M. G. Gouda, and S. S. Lam. Secure group communications using key graphs. In *ACM SIGCOMM'98,* pages 68–79, Vancouver, Canada, September, 1998.

[18] Y. Yang, X. Steve Li, X. Zhang, and S. S. Lam. Reliable group rekeying: A performance analysis. In *ACM SIGCOMM'01*, pages 27–38, San Diego, CA, 2001.

[19] Q. Zhang and K. L. Calvert. On rekey policies for secure group applications. In *Proceedings of 12th International Conference on Computer Communications and Network, (ICCCN 2003)*, Dallas, October, 2003.



Weifeng Chen received the B.S. and M.S. from Peking University and Chinese Academy of Sciences in 1998 and 2001, respectively. He received his Ph.D. in Computer Science from the University of Massachusetts at Amherst in 2006. He is currently an Assistant Professor in the Department of Mathematics and Computer Science at California University of Pennsylvania. His research interest includes network security, privacy and forensics.

Lakshminath R. Dondeti is a senior security architect in the advanced technology group at Nortel Networks, Billerica, MA. He is also the coauthor of Multicast and Group Security (Artech House, 2003). Dr. Dondeti received his B.E. from Motilal Nehru Regional Engineering College, Allahabad, India. He received his M.S. and Ph.D. from The University of Nebraska-Lincoln, all in computer science.

Ye Sun received the B.A. and M.S. from Beijing Normal University in 1998 and 2001, respectively. She received her Ph.D. from Texas A & M University in 2006. She is currently an Assistant Professor at West Virginia University.