

A Secure Group Key Management Protocol Communication Lower Bound *

Yang Richard Yang[†], Simon S. Lam
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
{yangyang, lam}@cs.utexas.edu

TR2000-24

July, 2000

Revised: September 2000

Abstract

We discuss in this paper a lower bound on communication cost for secure group key management protocols. To model a rekeying process, we introduce the concept of *rekey encryption graphs*. Using the rekey encryption graphs, we show that given the forward access control requirement, i.e. a user who has left the secure group cannot have access to future group keys, there exists a sequence of $2n$ user join and leave requests such that the amortized per request communication cost is $\Omega(\ln(n))$. Given the known protocols that have achieved this lower bound, in order to further improve the scalability of group rekeying communication protocol performance, future research needs to either allow more types of operations or to relax security requirements to achieve better performance.

*Research sponsored in part by National Science Foundation grant No. ANI-9977267 and grant no. ANI-9506048. Experiments were performed on equipment procured with NSF grant no. CDA-9624082.

[†]Contact author, Phone: (512)471-9599, Fax: (512)471-8885

1 Introduction

Many emerging network applications (such as teleconference and information dissemination services) are based upon a group communications model. As a result, securing group communications becomes a critical networking research issue. Recently, Internet Research Task Force (IRTF) has formed Secure Multicast Research Group (SMuG)[6] to investigate the problem of securing group communications. One major problem area in securing group communication is the group key management problem, which is concerned with the secure distribution and refreshment of user keying material.

The objective of a key management system is to add access control on top of efficient multicast communication such as over IP multicast [5]. A standard technique to this end is to maintain a common group key that is known to all multicast group members, but is unknown to non-group members. All group communication will be encrypted using this shared key. The main problem for this approach is that in a dynamic membership environment, users will join and leave the group, therefore, efficiently changing the group key becomes a performance issue.

It is clear that user join requests do not pose a issue because all users in the group share a common group key before the new user joins, and therefore can change to a new group key using the current group key. It is user leave requests that pose the scalability issue. Since the leaving user shares the group key with other users, in order to distribute a new group key to the remaining users, other keys may have to be used. In the simplest case, the key server may have to send the new group key encrypted by a remaining user's individual key, which is only shared between a user and the key server. If the number of users in the group is n , the complexity of this simple scheme has a complexity of $O(n)$.

In the past few years, several schemes have been proposed to improve rekeying performance, and these schemes can improve the rekeying complexity from $O(n)$ to $O(\ln(n))$ [9, 8, 1]. Besides the group key and individual keys, these schemes use auxiliary keys to improve rekeying performance. In particular, for two keys k and k' , they will use k to encrypt k' and then send the encryption to all users. Any user who have k will be able to decrypt k' .

With these proposed schemes, one remaining question is whether these schemes have achieved the best possible performance. In another words, can we do better? In order to answer this question, a study of the rekeying protocol lower bound will be helpful because it can not only show whether the proposed schemes have achieved the lower bound, it can also point out which constraints play an important role to derive the lower bound. In order to further improve performance,

either more types of operations have to be considered or some constraints have to be relaxed.

In the past few years, several lower bounds on broadcast encryption have been derived [2, 7, 3]. However, these lower bounds are mostly concerned with the tradeoffs between communication cost and storage cost. The lower bound closest to our result is derived in [3]. In [3], Canetti, Malkin and Nissim derived lower bounds on the tradeoffs between communication and user storage costs for group rekeying in a dynamic multicast group with one key server. In particular, they have derived the following two results: (1) Let M denote the set of users in a session. Let $b(n) + 1$ be the maximal number of keys held by any user in M , where n is the number of receivers in the session. Denote the rekeying communication cost $c(n)$ as the worst case number of keys to be encrypted when a user leaves. Then the rekeying communication cost satisfies $c(n) \geq n^{1/b(n)} - 1$. (2) Define a special class of rekeying protocols called *structure preserving* protocols. Intuitively, structure preserving protocols are those that maintain the property of “the set U has advantage over the user v ” across updates, for any subset U and user v . That is, if there is a set of users U all sharing a key k , and a user v which does not have this key, then after removing another user v' (whether $v' \in U$ or not), the users in U still hold some key k' that v does not hold. For this special class of rekeying protocol, the authors proved that the rekeying communication cost satisfies $c(n) \geq bn^{1/b} - b$, where $b + 1$ denotes the maximal number of keys held by any user in M .

These two results are the first to show the tradeoff between user storage cost $b(n)$ and communication cost $c(n)$ when a user leaves. From the communication lower bound of $c(n) \geq bn^{1/b} - b$, we may draw the conclusion that if we increase the number of keys each user holds to be very large, we may reduce the communication cost to be very low.

However, the bandwidth utilized by a rekeying protocol includes both user join communication cost and user leave communication cost, and the authors have only considered the leave communication cost. Intuitively, the storage cost $b(n)$ is related to join cost because the higher the number of keys a user holds, the higher the bandwidth to distribute these keys to the newly joined users. In the extreme case, when the key server distributes one key for each potential user subset when a user joins, the key server can reduce the leave communication cost to be constant. However, under this scheme, the communication cost to distribute the keys can be very high. From this discussion, we see that even if we are just interested in communication cost, we have to consider both user join communication cost and user leave communication cost.

The objective of this paper is to derive a lower bound on rekeying communication cost, considering both user join requests and leave requests. We show that given the forward access control requirement, i.e. a user who has left the secure group cannot have access to future group keys, there exists a sequence of $2n$ user joins and leaves such that the amortized per request communication cost is $\Omega(\ln(n))$.

The balance of this paper is organized as follows. In Section 2, we discuss our system model. We present our lower bound in Section 3. We conclude in Section 4.

2 System Model

2.1 Forward and backward access controls

There are two types of security requirements on a secure group key management system:

- Backward access control: A newly joined user cannot gain access to past group keys.
- Forward access control: After a user has left the secure group, she should not be able to gain access to future group keys.

When the only requirement is backward access control, a simple key management scheme with $O(1)$ complexity can be designed. Assume the current group key is g_i . When the $i + 1$ -th new user joins, the key server sets $g_{i+1} = H(k_i)$, where $H(\cdot)$ is a secure one way function. Then what the key server needs to do is to send g_{i+1} to the new user, and also multicasts a signal to other users so that these users will do the hash to get the new key. As another approach without using secure one way function, the key server can encrypt g_{i+1} by g_i . Again, this approach has $O(1)$ complexity.

The major difficulty arises when we have to provide forward access control. As we will see later, even without backward access control, in order to provide forward access control, the communication cost will have a lower bound of $\Omega(\ln(n))$, where n is the number of user join requests, and number of user leave requests. Hereafter, we only consider forward access control.

2.2 Rekey assumptions

To derive any lower bound on a system or to prove the security property of a system, we need to first define our system model. The assumptions we made about the system are:

- There is only one key server.
- After the key server has finished processing a request, all users in the session (joined, but has not left yet) share a common group key. The user who has left the secure group or has never joined the group does not have access to the key. We denote the group key after the i -th request as g_i . Notice that when the i -th request is a join request, and backward access control is not required, the key server may not change the group key, therefore, it is possible that g_i is the same as g_{i+1} in this case.
- When updating the keys, the key server uses one key k to encrypt another key k' .
- The communication cost is in terms of the number of times the key server encrypts one key with another key and distribute it. When we say a message, we mean one key encrypted by another key. In implementation, several encrypted keys can be put in one data packet.
- The adversary has infinite storage power, i.e., all messages distributed by the key server can be saved by the adversary.

2.3 Rekey encryption graphs

Inspired by the key graph approach [9] by Wong, Gouda and Lam, we use directed graphs to represent the rekeying process. However, we notice that a key graph in [9] represents just a snapshot of the rekeying process. In order to represent a whole rekeying process, and therefore make it possible to count the communication cost, we need to extend the key graph to include history informations. We call our extended graphs *rekey encryption graphs*.

Our rekey encryption graphs consist of a sequence of graphs $\{G_i\}_{i=1}^{\infty}$, where graph G_i models the rekeying process for the first i requests. Intuitively, we desire that the number of edges in G_i represents the number of messages that the key server has sent for rekeying the first i requests.

Next, we describe the nodes and edges in a rekey encryption graph G_i .

G_i will include two types of nodes: key nodes, and user nodes. The set of user nodes M includes all potential users. The second type of nodes is key node. We distinguish two types of key nodes. We first include a special class of key nodes called individual key nodes. These individual key nodes will be distributed by the key server using a secure channel established between a user and the key server at authentication time. The other key nodes represent the keys that the key server has ever generated and distributed by encrypting it using another key in G_i in order to process the first i requests.

G_i includes two types of edges. The first type of edges is from a user node u to its individual key node k . The second type of edges is inserted when a key k is distributed by the key server by encrypting it with another key k' . If the key server sends out such a message, we have an edge from k' to k .

For a graph G_i , we define a subgraph S_i . First, S_i includes a user node u if the user should be in the secure group after the first i requests (joined but not left). Second, S_i includes the current group key node g_i . Third, a key node or an edge is in S_i if it is on a path from a user node u in S_i to the current group key node g_i .

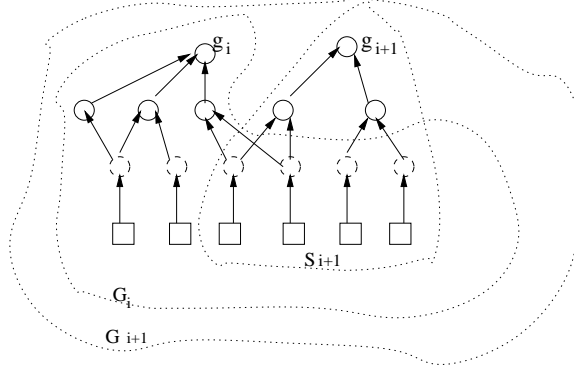


Figure 1: An example of rekey encryption graphs

Figure 1 shows examples of rekey encryption graphs. The whole graph represents G_{i+1} . Inside G_{i+1} in the dashed cycle is G_i . The six square nodes at the bottom of the graph are user nodes. The six dashed circle nodes just above the square nodes are the individual key nodes. There is an edge from a user node to its individual key node. Each edge from one key node k' to another key node k represents that the key server has sent out k encrypted by k' . g_i and g_{i+1} are the group keys of G_i and G_{i+1} , respectively. Also shown in Figure 1 is S_{i+1} , which includes the four user nodes on the right bottom, the group key node g_{i+1} , and all

the key nodes and edges that are on a path from a user node in S_{i+1} to the group key node g_{i+1} .

2.4 Rekey encryption graph properties

From our construction of rekey encryption graph G_i , we know that the number of edges in G_i represents the number of messages that the key server has sent to process the first i requests. We also notice that $G_i \subset G_j$, for any $i < j$.

However, not all graphs can represent a rekeying process. To represent a rekeying process that satisfy the forward access control requirement, a rekey graph G_i has to at least satisfy these following necessary properties:

- There is at least one path from a user node u to the current group key g_i if u is a member of the current group.
- There should be no path to the group key g_i if a user u is not in the current group.
- For the following property, we limit to $S_i \subset G_i$. Denote n as the total number of user nodes in S_i . Define $i(x)$ as the in-degree of a node x in S_i . Define $P(u)$ as the set of key nodes that are on a path from a user node u to the group key node g_i . We define the cost $c(u)$ of u as:

$$c(u) = \sum_{x \in P(u)} i(x) \quad (1)$$

Denote C as the maximal of $c(u)$ among the n users in S_i :

$$C = \max_{\forall \text{ user node } u} c(u) \quad (2)$$

Define $s(n)$ as the summation of the cost of all users in S_i :

$$s(n) = \sum_{\text{all user node } u} c(u) \quad (3)$$

Now, we can prove the following Lemma:

Lemma 1 $s(n) \geq n \ln(n)$

Proof: Define $n(x)$ as the number of user nodes in S_i that can lead to x . We observe that,

$$s(n) = \sum_{\text{all user node } u} c(u) = \sum_{\text{all key node } x} i(x)n(x) \quad (4)$$

Next, we prove $s(n) \geq n \ln(n)$ by induction on the height of the group key. It is obvious that the lemma holds for group key height 1. Assume the lemma is true for height less than or equal to h . First, we can see that we can reduce S_i into a tree by pruning some edges. Assume the group key has t non-overlapping connected children. Apply the induction assumption on each child branch, we have

$$s(n) \geq tn + \sum_{i=1}^t n_i \ln(n_i) \quad (5)$$

where n_i is the number of user nodes that are below the i -th child.

It is easy to verify that $f(x) = x \ln(x)$ is a convex function, therefore, we have

$$\sum_{i=1}^t n_i \ln(n_i) \geq t \frac{\sum_{i=1}^t n_i}{t} \ln\left(\frac{\sum_{i=1}^t n_i}{t}\right) \quad (6)$$

$$\geq n \ln\left(\frac{n}{t}\right) \quad (7)$$

Plug (7) into (5), we have

$$s(n) \geq nt + n \ln(n) - n \ln(t) \quad (8)$$

We know that $t - \ln(t) > 0$ for $t \geq 1$, therefore, $s(n) \geq n \ln(n)$. \square

Using Lemma 1, we have that

$$C \geq \ln(n) \quad (9)$$

2.5 Rekey encryption graph limitations

Before we present our lower bound, we discuss the limitation of our model in this subsection. The major limitation of this model is that we only allow the operation

of single encryption, i.e. encryption of one key by another key. It does not support other operations. For example, it is possible that one key k can be protected by two keys k' and k'' [4]. One way to implement this protection is to use the XOR of two keys, k' and k'' , to encrypt k . In this case, for an adversary to get k , it has to have both k' and k'' . To model this type of operation, we will have to increase the expressing power of rekey graph by adding AND edges. We will address this extension later.

3 Lower Bound of Rekeying Communication

With the preparation of the previous section, we prove in this section a lower bound on the secure rekeying communication cost, considering both join requests and leave requests. We notice that even though backward access control may not be required, the key server may want to distribute some keys at join time in order to reduce the cost of processing user leave request. Therefore, intuitively, it is important that we consider a sequence of operations instead of a single operation.

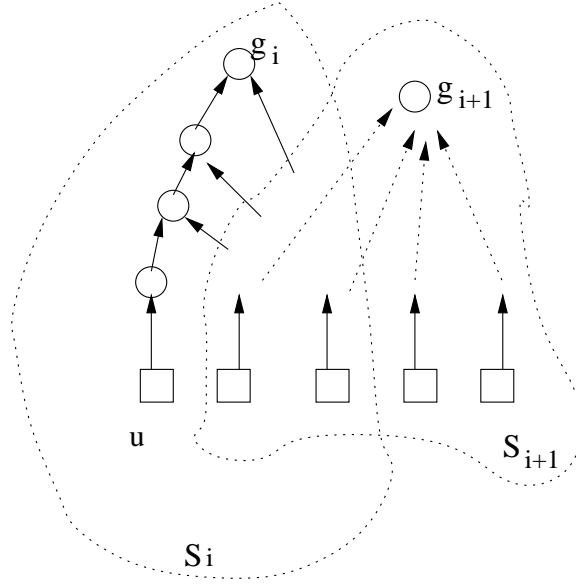


Figure 2: User u leaves

Suppose the i -th request is the leave request of user u . For the purpose of our construction, we assume that u does not re-join. Then because of forward access

control requirement, we know that there should not be a path from u to any group key g_j in G_j , where $j > i$. Let $P(u)$ denote the set of key nodes on the path from u to the group key node g_i in G_i . To satisfy the forward access control requirement, we know that no node in $P(u)$ should be in S_j , for any $j > i$, because otherwise user u can gain access to future group keys. Figure 2 shows the scenario.

From the definition of the cost of a user node, we know that there are at least $c(u)$ edges in S_i , and these edges will not be in S_j , where $j > i$. In particular, if we choose the user node with the largest cost, we know that there will be at least $\ln(n)$ edges in S_i , but these edges will not be in S_j , for any $j > i$.

Now, we can construct a sequence of $2n$ requests to have $\Omega(n \ln(n))$ edges. In another words, we will show that G_{2n} has at least $\Omega(n \ln(n))$ edges. Remember that the number of edges in G_i is the number of messages a key server has to send to process the i requests. Therefore, the amortized number of edges per request is $\Omega(\ln(n))$.

To construct the sequence, first assume n user join requests. These join requests are followed by a sequence of n user leave requests. Consider the first leave request. We know that this is the $n + 1$ -th request because there are n join requests ahead of it. We select the leaving user u as the user who has the maximum $c(u)$ of the remaining users. From previous discussion, we know that there are at least $\ln(n)$ edges in S_n but not in S_{n+1} . Since all previous requests are join requests, we know that all these edges are added to process the n join requests. Continue this process on the next leave request. In this case, we can get $\ln(n - 1)$ edges. We notice that the edges in the second leave may be either added when the key server is processing the n join requests or processing the first leave request. Continue this process for a total of n times, we have at least a total of $\ln(n) + \ln(n - 1) + \dots + 1 = \theta(n \ln(n))$ non-overlapping edges in the rekey encryption graph G_{2n} . Since there is a total of $2n$ requests, the amortized bandwidth requirements per request is $\Omega(\ln(n))$. Therefore, we have proved the following theorem:

Theorem 1 *Given forward access control security requirement, and the key server distributes non-individual keys by encrypting one key using another key, there exists a sequence of $2n$ requests such that the amortized per request communication cost is $\Omega(\ln(n))$.*

4 Conclusion

We discussed in this paper a lower bound on communication cost for secure group key management system. To model the rekeying process, we introduced the concept of *rekey encryption graphs*. Using the rekey encryption graphs, we were able to show that given the forward access control requirement, there exists a sequence of $2n$ join and leave requests such that the amortized per request communication cost is $\Omega(\ln(n))$. This lower bound indicates that when the only allowed operation for the key server to distribute a non-individual key is to encrypt it by another key, the communication cost will be $\theta(\ln(n))$, given the known protocols that have achieved this lower bound. Therefore, in order to further improve the scalability of group rekeying communication cost, other operations will have to be investigated. Another potential way to improve rekeying scalability is to relax the security requirements. For example, one potential is to allow some users to enjoy free ride when the lost of value can be offsetted by the saving of rekeying cost. We are currently investigating these possibilities.

References

- [1] David Balenson, David McGrew, and Alan Sherman. *Key Management for Large Dynamic Groups: One-way Function Trees and Amortized Initialization*, *INTERNET-DRAFT*, 1999.
- [2] C. Blundo, L. A. Frota Mattos, and D. R. Stinson. Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution. In *Advances in cryptology—CRYPTO '96*, Santa Barbara, CA, 1996.
- [3] Ran Canetti, Tal Malkin, and Kobbi Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *EuroCrypto '99*, 1999.
- [4] Isabella Chang, Robert Engel, Dilip Kandlur, Dimitrios Pendarakis, and Debanjan Saha. Key management for secure Internet multicast using boolean function minimization techniques. In *Proceedings of IEEE INFOCOM '99*, volume 2, March 1999.
- [5] S.E. Deering and D.R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

- [6] Internet Research Task Force (IRTF). The secure multicast research group (SMuG). <http://www.ipmulticast.com/community/smug/>.
- [7] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In *Advances in cryptology-EUROCRYPT '98*, Espoo, Finland, 1998.
- [8] D. Wallner, E. Harder, and Ryan Agee. *Key Management for Multicast: Issues and Architectures*, *INTERNET-DRAFT*, September 1998.
- [9] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. In *Proceedings of ACM SIGCOMM '98*, September 1998.