Rekeying and Storage Cost for Multiple User Revocation[1]

Sandeep S. Kulkarni Bezawada Bruhadeshwar
Department of Computer Science and Engineering
Michigan State University
East Lansing MI 48824 USA

Abstract

In this paper, we focus on tradeoffs between storage cost and rekeying cost for secure multicast. Specifically, we present a family of algorithms that provide a tradeoff between the number of keys maintained by users and the time required for rekeying due to revocation of multiple users. We show that some well known algorithms in the literature are members of this family. We show that algorithms in this family can be used to reduce the cost of rekeying by $43\% - 79\%$ when compared with previous solutions while keeping the number of keys manageable. We also describe a scheme that allows one to reduce the number of keys further without increasing the rekeying cost.

Keywords : Secure Multicast, Hierarchical Key Management, Rekeying and Storage Tradeoffs, User Requirements and Capabilities, Heterogeneous Environments

1 Introduction

Applications such as conferencing, distributed interactive simulations, networked gaming and news dissemination are group oriented. In these applications, it is necessary to secure the group communication as the data is sensitive or it requires the users to pay for it. In the algorithms for secure group communication (e.g., [1–7]), a group controller distributes a cryptographic key, called the group key, to all users. The group key is used to encrypt data transmitted to the group. The group membership is dynamic. When group membership changes, to protect the privacy of the current users, the group controller changes and securely distributes the new group key.

When a user is admitted to the group, the group controller changes the group key and securely unicasts it to the joining user. To send the new group key to the current users, the group controller encrypts it with the old group key and multicasts it to them. Thus, the cost of rekeying for the

[1]Email: sandeep@cse.msu.edu, bezawada@cse.msu.edu
Web: `http://www.cse.msu.edu/~{sandeep,bezawada}`,
Tel: +1-517-355-2387, Fax: 1-517-432-1061
This work is partially sponsored by NSF CAREER 0092724, ONR grant N00014-01-1-0744, DARPA contract F33615-01-C-1901, and a grant from Michigan State University.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Rekeying and Storage Cost for Multiple User Revocation	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Michigan State University,Department of Computer Science and Engineering,East Lansing,MI,48824**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**12**	

group controller due to a joining user is small. However, when a user is revoked, i.e., the user leaves or is forcefully removed from the group, the group controller needs to securely unicast the new group key to each of the remaining users. Thus, revoking users from secure groups is more expensive.

Many solutions have been proposed (e.g., [2–5, 7]) for efficiently revoking a single user. In these solutions, for a group of N users, the group controller distributes the new group key in $O(logN)$ encrypted messages. To revoke multiple users, the group controller repeats the process of rekeying for each revoked user. Hence, in these solutions, the cost of rekeying is high. Moreover, if the group controller were to interrupt the group communication during the rekeying, the resulting delay is unreasonable for many applications. Thus, efficient distribution of the new group key to revoke multiple users is a critical problem in secure group communication.

One approach to revoke multiple users is to associate a key with every non-empty subset of users in the group. Thus, if one or more users are revoked, the group controller uses the key associated with the subset of the remaining users to encrypt the new group key and transmits it to the users. The advantage of this approach is that the communication overhead is only one message for revoking any number of users. However, the number of keys stored by the group controller and the users is exponential in the size of the group. In this paper, we describe a family of key management algorithms that reduce the rekeying cost due to multiple user revocation while keeping the storage cost manageable. Using our algorithms, the group controller can efficiently distribute the group key. The main contributions of our paper are as follows:

- We describe our family of key management algorithms for efficiently distributing the new group key when multiple users are revoked from the group. In our algorithms, the storage at the group controller is linear and the storage at the users is logarithmic in the size of the group. Also, we show that many existing algorithms (e.g., [3, 4]) are members of this family.
- We argue the applicability of our algorithms to scenarios where users have varying requirements or capabilities. As an illustration, we provide a scenario in which users are classified as long-lived or transient based on the duration of their group membership.
- We describe a scheme to further reduce the number of keys stored by the users and the group controller.

Organization of the paper: The paper is organized as follows. In Section 2, we describe the problem of group key distribution and discuss some related solutions. In Section 3, we describe our family of key management algorithms and present sample algorithms from this family. In Section 4, we describe a scheme which reduces the storage at the group controller and the users. In Section 5, we present the simulation results of our algorithms and compare their performance against previous solutions. In Section 6, we conclude the paper and discuss future work.

2 Key Distribution in Secure Multicast

To ensure group security, all users in the group share a group key, k_g. The group key is used to encrypt the data transmitted to the group. When users are revoked from the group, to protect the privacy of the remaining users, the group controller needs to change and distribute the new group key to the remaining users. To simplify the distribution of the new group key, each user maintains

additional keys (e.g., in [2–5, 7]), which are shared with other users. To send the new group key, k'_g, to the remaining users, the group controller encrypts k'_g using the shared keys not known to the revoked users. To reflect current group membership, the group controller also needs to change and distribute the shared keys that are known to the revoked users. There are two approaches available with the group controller for distributing the new shared keys. In the first approach, the group controller explicitly transmits the new shared keys (e.g., in [2, 3, 5]) to the current users. In our work, we adopt the second approach where the group controller and the users update the shared keys using the following technique: $k'_x = f(k'_g, k_x)$, where k_x is the old shared key, k'_x is the new shared key and, f is a one-way function. Using this technique, only those current users who knew the old shared key, k_x, will be able to get the new shared key, k'_x. This technique was also used in [4, 8]. However, this technique may be prone to long term collusive attacks, as described in [4], by the revoked users. To provide resistance against such attacks, the group controller adopts a policy in which the keys known to the current users are refreshed at regular intervals of time.

From the above discussion, we note that, the rekeying cost for the group controller to revoke multiple users is the cost of sending the new group key. We measure this cost in the number of messages sent and the encryptions performed by the group controller for distributing the new group key. In Section 3, we describe our key management algorithms and the techniques for distributing the new group key. Using simulation results, we show that our algorithms reduce the cost of rekeying by 43%-79% when compared with the existing solutions.

Related Work. Other approaches to address the problem of revoking multiple users are proposed in [9–13]. In [9], the group controller maintains a logical hierarchy of keys that are shared by different subsets of the users. To revoke multiple users, the group controller aggregates all the necessary key updates to be performed and processes them in a single step. However, the group controller interrupts the group communication until all the necessary key updates are performed and then, distributes the new group key to restore group communication. This interruption to group communication is undesirable for real-time and multi-media applications. In [10], to handle multiple group membership changes, the group controller performs periodic rekeying, i.e., instead of rekeying whenever group membership changes, the group controller performs rekeying only at the end of selected time intervals. However, the revoked users can access group communication until the group is rekeyed. This can either cause monetary loss to the service provider or compromise confidentiality of other users. In [11], the group controller maintains a logical hierarchy of keys similar to the solution in [9]. To revoke multiple users, the group controller distributes the new group key using keys that are not known to the revoked users. However, this solution achieves a good rekeying cost only if the size of the revoked users is small or very large.

In [12], Luby and Staddon focus on the tradeoff between the storage cost and the rekeying cost. They identify a lower bound on the rekeying cost based on the number of keys that the users maintain. Their work is based on previous work in [13] and assumes that an upper bound on the number of users, say x, that need to be revoked is known in advance. The key distribution algorithm in [13] uses the value of x to distribute the keys. Hence, if the number of users that need to be revoked is more than x then their algorithm fails to revoke them using the shared keys. By contrast, our algorithm does not assume that the number of revoked users is known in advance.

Notations. We use $k\{m\}$ to denote that message m is encrypted with key k. Only users who know k can decrypt this message. The adversary (anyone outside the group) can listen to all messages sent over the network. Hence, for simplicity, we assume that all communication is broadcast in nature and, hence, we do not explicitly identify the intended recipients of a message.

This assumption is similar to that in [2–5, 7].

3 Key Management Algorithms

In Section 3.1, we describe the basic structure and the associated key management algorithm. In Section 3.2, we describe our hierarchical key management algorithm for larger groups using the concepts in the basic scheme.

3.1 The Basic Structure

We arrange a group of K users as children of a rooted tree as shown in Figure 1. Let R be the root node. We use the tuple, $\langle R, u_1, u_2, \ldots, u_K \rangle$, to denote the basic structure.

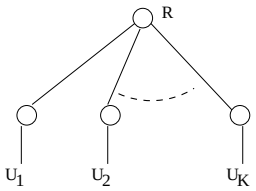


Figure 1: Partial View of Basic Structure

The key management algorithm we use for the basic structure is the *complete key graph* algorithm from [3]. In this algorithm, for every non-empty subset of users the group controller provides a unique shared key which is known only to the users in the subset. The group controller gives these keys to the users at the time of joining the group. Of the keys that a user, say u_i, receives, (1) one key is associated with the set $\{u_1, u_2 \ldots, u_K\}$ and, hence is known to all the users, and (2) one key is associated with the set $\{u_i\}$. The former key, say k_R, is the group key whereas the latter key is the personal key.

Thus, the number of keys stored by the group controller are $2^K - 1$ and the number of keys held by each user is 2^{K-1}. Now, we consider the process of rekeying in this scheme when one or more users are revoked from the group. The proof of the following theorem describes the rekeying process for user revocation.

Theorem 1. *In the basic structure, when one or more users are revoked, the group controller can distribute the new group key securely to the remaining users using at most one encrypted transmission.*

Proof. We consider 3 possible cases of user revocation from the basic structure.

Case 0. When no users are revoked, the group controller sends the new group key using the current group key that is known to all the users. Although, this trivial case is not important for the basic scheme, it is important for the hierarchical algorithm we describe in Section 3.2.

Case 1. When $m < K$ users are revoked from the group and the group controller needs to distribute the new group key to the remaining $K - m$ users. The group controller uses the shared key, k_{K-m} associated with the remaining subset of $K - m$ users to send the new group key. Thus, the group

controller transmits $k_{K-m}\{k'_g\}$. As the revoked users do not know k_{K-m}, only the current users will be able to decrypt this message.

Case 2. All users are revoked from the group. The group controller does not need to distribute the new group key and thus, does not send any messages. □

We note that, once the new group key is distributed, the current users update the necessary shared keys using the one-way function technique we described in Section 2. However, the basic structure requires the group controller and the users to store a large number of keys which is not practical if the group is large. In the Section 3.2, we present our hierarchical algorithm to reduce the number of keys stored at the group controller and the users. Our hierarchical algorithm preserves some attractive communication properties of the basic structure while reducing the storage requirement for the shared keys.

3.2 The Hierarchical Key Management Algorithm

In our hierarchical algorithm, we compose smaller basic structures in a hierarchical fashion. To illustrate the hierarchical structure, consider the sample structure $\langle R, R_1, R_2, \ldots, R_d\rangle$ shown in Figure 2, where each R_i, $0 \leq i \leq d$, further consists of the basic structure $\langle R_i, u_{i1}, u_{i2}, \ldots, u_{id}\rangle$. The parameter d is the number of elements in a basic structure and can be considered as the degree of the hierarchy. We note that, the degree can be different for different nodes in the hierarchy. However, for the sake of simplicity, in this paper, we assume that the nodes in the hierarchical structures have a uniform degree d.

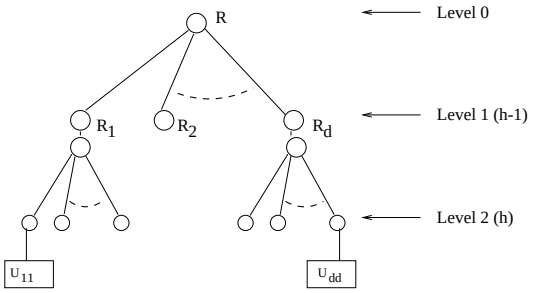


Figure 2: Partial View of Hierarchical Key Management Structure

Now, each of the basic structures of the form $\langle R_i, u_{i1}, u_{i2}, \ldots, u_{id}\rangle$ is associated with the shared keys as described in Section 3.1. The structure at next higher level, $\langle R, R_1, R_2, \ldots, R_d\rangle$, is also associated with shared keys. The personal key associated with R_i, $1 \leq i \leq d$ in structure $\langle R, R_1, R_2, \ldots, R_d\rangle$ is the same as the group key of the structure $\langle R_i, u_{i1}, u_{i2}, \ldots, u_{id}\rangle$. Further, the structure $\langle R, R_1, R_2, \ldots, R_d\rangle$ is associated with shared keys. Now, each user in the basic structure $\langle R_i, u_{i1}, u_{i2}, \ldots, u_{id}\rangle$ is provided with any shared key that is provided to R_i in the structure $\langle R, R_1, R_2, \ldots, R_d\rangle$. To illustrate our hierarchical algorithm, we consider four examples for d=N, 2, 3, 4. In the hierarchical structure, we denote the key associated with a subset $\{a, b, \ldots, z\}$ by $k_{ab\ldots z}$.

Example 0. When $d = N$, the key management algorithm corresponds to the basic structure (cf. Section 3.1) with $K = N$. Thus, the number of keys maintained by the group controller are $2^N - 1$ and the number of keys maintained by each user are 2^{N-1}.

Example 1. In Figure 3(a), we show a hierarchy with $d = 2$. Consider that the shared keys

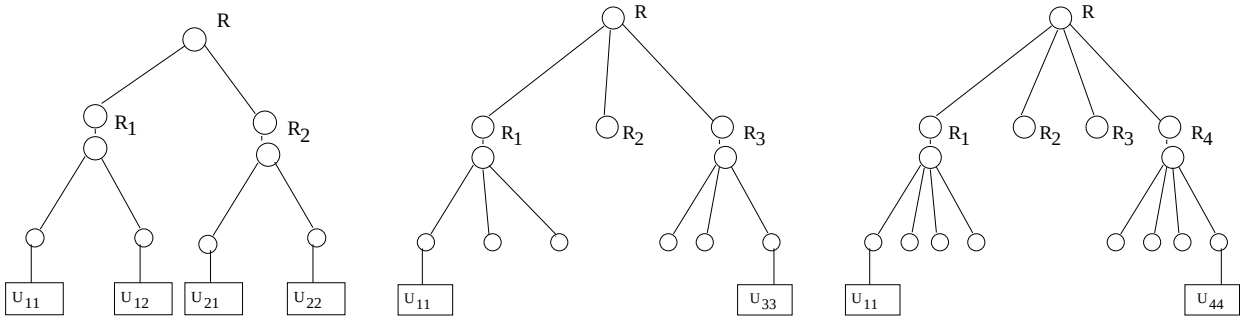


Figure 3: Hierarchies for (a) degree=2 (b) degree=3 and (c) degree=4

associated with $\langle R, R_1, R_2\rangle$ are $\{k_R, k_{R_1}, k_{R_2}\}$, and the shared keys associated with $\langle R_1, u_{11}, u_{12}\rangle$ are $\{k_{R_1}, k_{u_{11}}, k_{u_{12}}\}$. Then, in this scheme, user u_{11}, knows the shared keys $k_{u_{11}}$, k_{R_1} and k_R. We note that, the hierarchical algorithm for $d = 2$ corresponds to the logical key hierarchy proposed in [2,3].

Example 2. In Figure 3(b), we show a partial view of a sample hierarchy with $d = 3$. Consider that the shared keys associated with $\langle R, R_1, R_2, R_3\rangle$ are, $\{k_R, k_{R_1}, k_{R_2}, k_{R_3}, k_{R_1R_2}, k_{R_1R_3}, k_{R_2R_3}\}$, and the shared keys associated with $\langle R_1, u_{11}, u_{12}, u_{13}\rangle$ are, $\{k_{u_{11}}, k_{u_{12}}, k_{u_{13}}, k_{u_{11}u_{12}}, k_{u_{11}u_{13}}, k_{u_{12}u_{13}}, k_{R_1}\}$, then, the shared keys known to user u_{11} are, $\{k_{u_{11}}, k_{u_{11}u_{12}}, k_{u_{11}u_{13}}, k_{R_1}, k_{R_1R_2}, k_{R_1R_3}, k_R\}$. We note that, the hierarchy for $d = 3$ corresponds to the complementary key hierarchy proposed in [4].

Example 3. In Figure 3(c), we show a partial view of a sample hierarchy with $d = 4$. Consider that the shared keys associated with $\langle R_1, R_2, R_3, R_4\rangle$ are, $\{k_R, k_{R_1}, k_{R_2}, k_{R_3}, k_{R_4}, k_{R_1R_2}, k_{R_1R_3}, k_{R_1R_4}, k_{R_2R_3}, k_{R_2R_4}, k_{R_3R_4}, k_{R_1R_2R_3}, k_{R_1R_2R_4}, k_{R_1R_3R_4}, k_{R_2R_3R_4}\}$ and, the shared keys associated with $\langle R_1, u_{11}, u_{12}, u_{13}, u_{14}\rangle$ are, $\{k_{u_{11}}, k_{u_{12}}, k_{u_{13}}, k_{u_{14}}, k_{u_{11}u_{12}}, k_{u_{11}u_{13}}, k_{u_{11}u_{14}}, k_{u_{12}u_{13}}, k_{u_{12}u_{14}}, k_{u_{13}u_{14}}, k_{u_{11}u_{12}u_{13}}, k_{u_{11}u_{12}u_{14}}, k_{u_{11}u_{13}u_{14}}, k_{u_{12}u_{13}u_{14}}, k_{R_1}\}$. Then, the shared keys known to user u_{11} are, $\{k_{u_{11}}, k_{u_{11}u_{12}}, k_{u_{11}u_{13}}, k_{u_{11}u_{14}}, k_{u_{11}u_{12}u_{13}}, k_{u_{11}u_{12}u_{14}}, k_{u_{11}u_{13}u_{14}}, k_{R_1}, k_{R_1R_2}, k_{R_1R_3}, k_{R_1R_4}, k_{R_1R_2R_3}, k_{R_1R_2R_4}, k_{R_1R_3R_4}, k_R\}$.

Now, we describe the process of rekeying for user revocation for an arrangement with h levels.

Theorem 2. *In our hierarchical key management algorithm, when r users are revoked from a hierarchical structure with h levels, the group controller can distribute the new group key securely to the remaining users using at most $h.r$ encrypted transmissions.*

Proof. We mark all the nodes which are the ancestors of the revoked users. At each level, a marked node, R_i can be considered to be revoked from the structure, $\langle R, R_1 \ldots, R_i, \ldots, R_d\rangle$. As an illustration, in the hierarchy with $d = 4$ in Figure 3(c), if u_{11}, u_{44} are revoked users, then we mark R_1 and R_4. We consider u_{11} to be revoked from the basic structure, $\langle R_1, u_{11}, u_{12}, u_{13}, u_{14}\rangle$ and R_1 to be revoked from the structure $\langle R, R_1, R_2, R_3, R_4\rangle$. Similarly, u_{44} is revoked from the basic structure, $\langle R_4, u_{41}, u_{42}, u_{43}, u_{44}\rangle$ and R_4 is revoked from the structure, $\langle R, R_1, R_2, R_3, R_4\rangle$.

To send the new group key, at the lowest level (level h), the group controller needs to send at most one message (cf. Theorem 1 from Section 3.1) for the basic structures from which users are revoked. As the number of basic structures from which users are revoked is at most r, the rekeying cost due

to the h^{th} level is r. We note that, the number of encryptions and messages will be lower if more than one user is revoked from the same basic structure.

At the next higher level (level $h-1$), the number of revoked nodes, i.e., marked nodes, is at most r. At this level, to send the new group key, the group controller sends at most one encrypted message for each structure. Based on the key distribution in the hierarchical algorithm, this message is decrypted by the users which are children of the non-revoked nodes in each such structure. As the number of such structures at this level is at most r, the group controller sends at most r messages for this level. Further, in the worst case, the group controller sends r messages for all the levels in the hierarchy. Thus, for r revoked users, the cost of distributing the new group key is at most $h.r$ encrypted transmissions. □

We note that, at the highest level (level 1), there is only one structure. As this scenario is similar to user revocation from a basic structure, using Theorem 1 at this level, the group controller sends at most one encrypted message for the new group key. Therefore, we can reduce the total rekeying cost from Theorem 2 to $(h-1).r+1$. Thus, we have:

Theorem 3 *In our hierarchical key management algorithm, when r users are revoked, the group controller can distribute the new group key securely to the remaining users using at most $(h-1).r+1$ encrypted transmissions.* □

The upper bounds in Theorems 1 and 2 are tight for a small number of revoked users. If the number of revoked users is $O(N)$ where N is the group size, then the group controller can distribute the group key by only considering the structures at the lowest level. The group controller sends at most one message for each basic structure. As there are N/d basic structures at the lowest level, the group controller sends at most N/d messages.

Theorem 4 *In our hierarchical key management algorithm, for revoking any number of users, the group controller can distribute the new group key securely to the remaining users using at most N/d encrypted transmissions.* □

We can combine the results in Theorems 2 and 4 as follows. To revoke r users, for k lower levels in the hierarchy, the group controller uses the result from Theorem 2 and sends at most $(h-k).r$ encrypted messages. At level k, for each of the d^{k-1} structures, the group controller uses the result from Theorem 4 and sends at most d^{k-1} encrypted messages. Therefore, we can also say that the rekeying cost in our hierarchical key management algorithm is bounded by $(h-k).r+d^{k-1}$ where $1 \leq k \leq h$.

We note that, all the results we derived give upper bounds on the rekeying cost for revoking users. Thus, the minimum of these bounds is still an upper bound. The simulation results show that, on an average, the performance of our algorithms is slightly better than these bounds.

Some of the basic structures in the hierarchical structure may have less than d users. To revoke users, the group controller assumes that all the basic structures (cf. Section 3.1) are *full*. This assumption allows the group controller to distribute the group key according the rekeying techniques we described in Theorems 2, 3 and 4. We note that, in this model, the rekeying cost for the group controller does not increase and is determined by the actual number of revoked users. For a full hierarchical structure with h levels of hierarchy, the group controller stores, $(\frac{d^h-1}{d-1})(2^d-1)$ keys and, each user stores $h.(2^{d-1})$ keys. Thus, in the hierarchical structure, for small values of d, the user needs to store $O(h)$ keys as against $O(2^{N-1})$ keys in the basic structure.

4 Reducing Storage Requirements Further

In this section, we propose an approach to reduce the number of keys maintained by the users and the group controller by deriving some keys from other keys. We illustrate this approach in the context of the basic scheme with degree 3. Specifically, we illustrate how the singleton keys used by the users, say u_1, u_2 and u_3, can be used to derive the keys that are shared by a group of two users.

Let the singleton keys maintained by users u_1, u_2 and u_3 be $k_{u_{11}}$, $k_{u_{12}}$ and $k_{u_{13}}$ respectively. Now, the keys shared between a group of two users can be computed using these keys. For example, the key shared between u_1 and u_2, say $k_{u_{11}u_{12}}$, can be assigned $f(k_{u_{11}})$ where f is a one way function. (Likewise, $k_{u_{12}u_{13}}$ can be assigned to be equal to $f(k_{u_{12}})$ and $k_{u_{11}u_{13}}$ can be assigned to equal to $f(k_{u_{13}})$.) Now, if either $k_{u_{11}}$ or $k_{u_{11}u_{12}}$ ever needs to be changed (e.g., such a situation can occur in hierarchical algorithm), then users, u_1 and u_2 should be able to independently compute the newer version of the keys they maintain. Towards this end, we let f be the Rabin function, $f(x) = x^2 \mod N$, where N is a product of two large primes.

Now, whenever $k_{u_{11}}$ is changed, user u_1 updates $k_{u_{11}}$ as, $k'_{u_{11}} = k_{u_{11}}.(k^2_{u_{11}} + k'_g) \mod N$, where k'_g is the new group key. Now, $k_{u_{11}u_{12}}$ should be changed to $(k'_{u_{11}})^2 \mod N$. Thus, to update $k_{u_{11}u_{12}}$, user u_2 needs to compute $[(k_{u_{11}})^2(k^2_{u_{11}} + k'_g)^2 \mod N] \mod N$. Clearly, this value is equal to $((k^2_{u_{11}} \mod N) * (k^2_{u_{11}} \mod N + k'_g \mod N)^2) \mod N$. It is straightforward to observe that u_2 can compute $k'_{u_{11}u_{12}}$ as it is aware of $f(k_{u_{11}}) = (k_{u_{11}})^2 \mod N$ and the new group key k'_g. Moreover, based on the one-way function property of the Rabin function, u_2 cannot obtain $k'_{u_{11}}$.

With this approach, the keys maintained by u_1 are k_R (group key), $k_{u_{11}}$, and $f(k_{u_{13}})$. Thus, the number of keys maintained by u_1 is reduced from 4 to 3. And, the number of keys maintained by the group controller are reduced from 7 to 4. Similar approach can be easily used in hierarchical algorithm as well as for algorithms with higher degree. However, this issue is outside the scope of this paper.

5 Simulation Results and Analysis

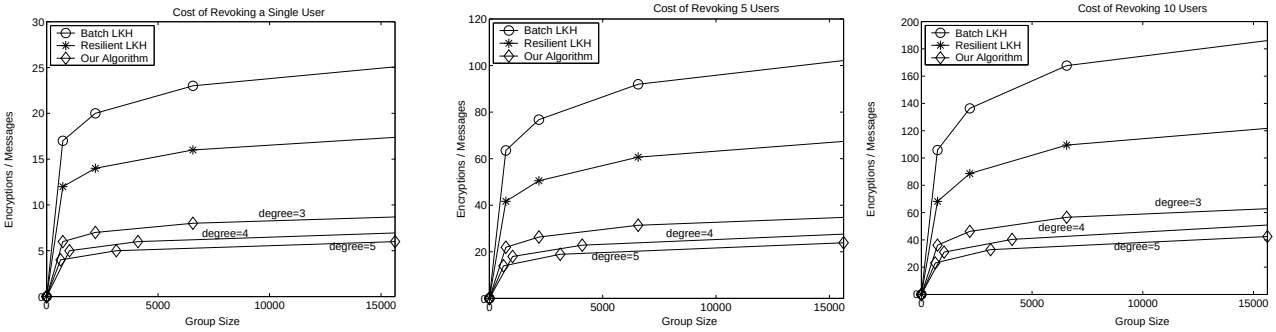
We compare the performance of our algorithms with the algorithms in [9, 11]. In [9], the group controller associates a set of keys with the nodes of a rooted tree and the users with the leaves of the tree. Each user knows the keys associated with the nodes on the path from itself to the root. To revoke a user, the group controller recursively distributes the changed keys at the higher levels in the key tree using the changed keys at the lower levels. To revoke multiple users, instead of sequentially distributing the changed keys for each revoked user, the group controller processes all the key updates in a single step. This reduces the cost of changing a key multiple times if it is known to multiple revoked users. In [11], the group controller maintains a key tree similar to [9]. Each node in the key tree is associated with a public key and a private key pair. To revoke multiple users, the group controller traverses the tree and determines the common ancestors of the remaining users. The group controller uses the public keys of these ancestors to send the new group key to the remaining users. The remaining users use the private keys known to them and determine the new group key from the information sent by the group controller.

Methodology of Experiments. We denote the algorithm from [9] by *Batch LKH*, the algorithm from [11] by *Resilient LKH* and our algorithm by *Our Algorithm*. In the simulations, we assume

that the algorithms maintain full and balanced hierarchies (respectively, trees) of keys. For each experiment, we selected a random set of users to be revoked from the group, and recorded the number of encrypted messages sent by the group controller for the new group key.

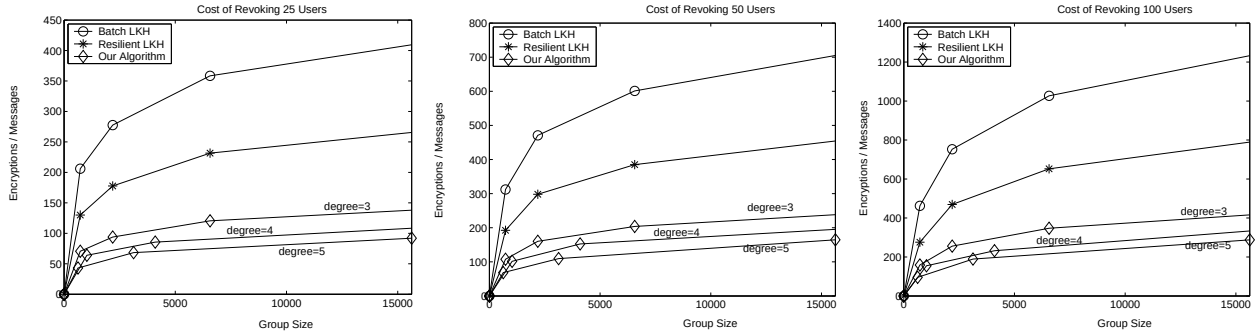
We simulated the algorithms with degrees 3, 4 and 5. For each experiment, we computed the average cost of user revocation over 100 trials. The results are shown in Figures 4-5. Regarding our algorithms, we consider degrees 3, 4 and 5. We also experimented with these degree values for the algorithms in [9, 11] and selected the version with the minimal cost.

Based on these figures, as the degree of the hierarchy increases, the rekeying cost reduces due to the reduction of the height h of the hierarchy. From these results, we observe that our algorithms perform much better than the existing solutions. Specifically, the cost of rekeying in our algorithm is $66\% - 79\%$ less than that of [9] and $43\% - 74\%$ less than that of [11]. Finally, the algorithm in [9] is an optimization to the logical key hierarchy in [3] for handling multiple user revocations. Hence, our algorithm reduces the rekeying cost to a value that is less than that in [3].



(a) (b) (c)

Figure 4: Comparison of Rekeying Costs for Revoking. (a) One User (b) 5 users and (c) 10 users

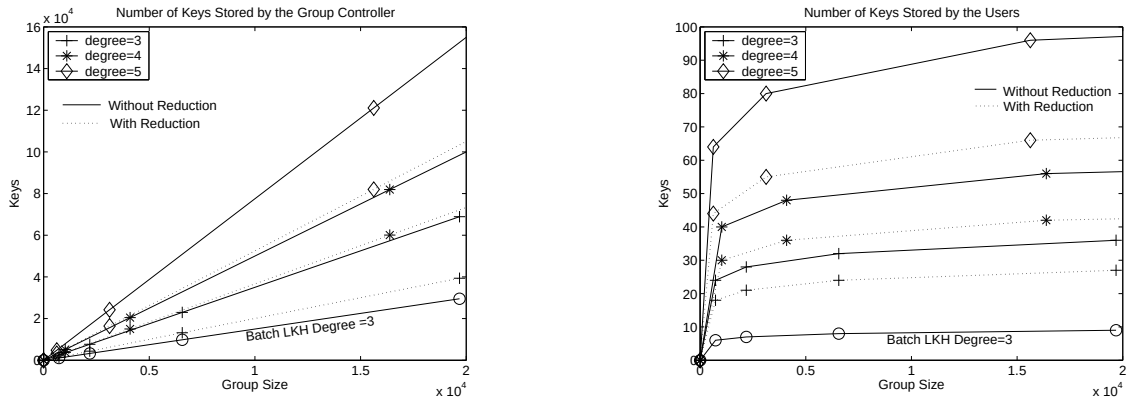


(a) (b) (c)

Figure 5: Comparison of Rekeying Costs for Revoking (a) 25 users (b) 50 users and (c) 100 users

In figure 6, we compare the keys maintained by our algorithm for various degrees with the keys maintained in [9,10]. As we can see, the number of keys is manageable and there is a tradeoff by which maintaining a larger number of keys per user, it is possible to reduce the cost of rekeying.

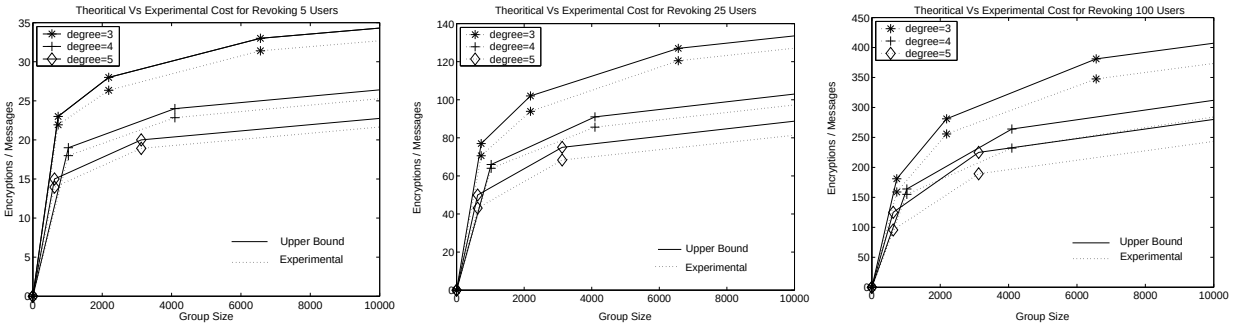
Another important observation in this context is illustrated in Figure 7. Specifically, in this figure,



(a) (b)

Figure 6: Number of Keys Stored by the (a) Group Controller and (b) Each User

we compare the upper bound identified in Section 3.2 with the experimental value. As shown in Section 3.2, the upper bound for rekeying cost is the minimum of $(h-k).r+d^{k-1}$, where $1 \leq k \leq h$. From this figure, it follows that the experimental value is a close estimate to the upper bound that is computed analytically. For this reason, the group controller can use this analytical estimate in deciding the choice of degree that should be chosen so that the rekeying cost remains within acceptable limits.



(a) (b) (c)

Figure 7: Comparison of Theoretical Upper Bounds Vs Experimentally Results for Rekeying Cost to revoke (a) 5 users (b) 25 users and (c) 100 users

6 Conclusion

In this paper, we presented a family of algorithms that provide a tradeoff between the number of keys maintained by the users and the time required for rekeying due to the revocation of multiple users. We showed that our algorithms reduce the cost of rekeying by $43\%-79\%$ when compared with the previous solutions in [9, 11] while keeping the number of keys manageable. We also described a scheme that allows one to reduce the number of keys further without increasing the rekeying cost.

Our algorithms also enable the group controller to deal with heterogeneous set of users that have

different capabilities. We illustrate this by a simple example. Consider the case where the basic structure at the root level has a degree 2, the trees rooted at the left child of the root can only maintain a small number of keys, and the users rooted at the right child of the root can maintain a large number of keys. Now, we can use a smaller degree for the tree rooted at the left child and a larger degree for the tree rooted at the right child. With such a design, the users in the left tree will receive only a small number of keys whereas the users in the right tree will receive a large number of keys. It follows that for the right tree, the group controller can take advantage of reduced rekeying cost provided by the use of a tree with larger degree, while still allowing users with lower capabilities to participate in the group communication.

Alternatively, our algorithms could be used in cases where the nature of the users varies. To illustrate this issue, consider the case where the group consists of two kinds of users, *long-lived* and *short-lived*. If the group controller can obtain such information (which may contain a small number of errors) based on the past behavior of users then the group controller can provide a smaller number of keys to long-lived users and a larger number of keys for short-lived users. In other words, the group controller can provide a preferential treatment to long-lived users.

Our algorithms are also suited for overlay multicast applications. In overlay multicast [14–16], the end nodes perform the processing and forwarding of multicast data without using IP multicast support. As these tasks result in increased overhead at the end nodes, reducing control traffic is an important problem for overlay multicast. Our algorithms reduce the overhead at the end nodes by reducing the number of group key update messages sent by the group controller. These benefits are also desirable in wireless systems which are constrained in battery power. In [17], the measurements on wireless network interface cards show that transmission consumes more battery power than reception if the idle listening time of the interface is small. As streaming multicast sessions result in minimal idle time, the energy consumption is dominated by the amount of transmitted data. Thus, in heterogeneous systems which comprise of wired and wireless systems, our algorithms can be used to improve battery longevity of wireless systems by reducing the amount of traffic they need to transmit forward.

Our hierarchical algorithm can be combined with the logical key hierarchy in [3]. A major motivation for such a combination is to reduce the storage and computational overhead of the users. To achieve this, the group controller determines the utility of different keys at each level during user revocation over a period of time and discards those keys which are the least useful. For example, the group controller can maintain only logical keys at higher levels in the hierarchy if additional shared keys, as required in our hierarchical algorithms, are not useful. Such hybrid schemes will allow the group controller to adapt to heterogeneous systems where users have varying requirements and capabilities. We are working on suitable techniques to combine these algorithms based on the application requirements.

References

[1] H.Harney and C.Muckenhirn. Group key management protocol (GKMP) specification. RFC 2093, July 1997.

[2] Debby M. Wallner, Eric J. Harder, and Ryan C. Agee. Key management for multicast: Issues and architectures. RFC 2627.

[3] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 2000.

[4] Sandeep S. Kulkarni and Bezawada Bruhadeshwar. Adaptive rekeying for secure multicast. *IEEE/IEICE Special issue on Communications: Transactions on Communications*, E86-B(10):2948–2956, October 2003.

[5] D.McGrew and A.Sherman. Key establishment in large dynamic groups using one-way function trees. Manuscript.

[6] S.Mittra. Iolus: A framework for scalable secure multicasting. In *Proc. ACM SIGCOMM'97*, pages 277–288, 1997.

[7] Marcel Waldvogel, Germano Caronni, Dan Sun, Nathalie Weiler, and Bernhard Plattner. The versakey framework: Versatile group key management. *IEEE Journal on Selected Areas in Communications*, 1999.

[8] Isabella Chang, Robert Engel, Dilip Kandlur, Dimitrios Pendarakis, and Debanjan Saha. Key management for secure internet multicast using boolean function minimization techniques. In *Proceedings IEEE Infocomm'99*, volume 2, pages 689–698, March 1999.

[9] X. Steve Li, Yang Richard Yang, Mohamed Gouda, and Simon S. Lam. Batch updates of key trees. In *Proceedings of the Tenth International World Wide Web Conference (WWW10)*, Hong Kong, China, May 2001.

[10] S. Setia, Samir Koushish, and Sushil Jajodia. Kronos: A scalable group re-keying approach for secure multicast. In *IEEE Symposium on Security and Privacy*, pages 215–228, 2000.

[11] Hartono Kurnio, Safavi-Naini Rei, and Huaxiong Wang. Efficient revocation schemes for secure multicast. In *International Conference on Information Security and Cryptology 2001, Lecture Notes in Computer Science*, volume 2288, pages 160–177, December 2002.

[12] M. Luby and J.Staddon. Combinatorial bounds for broadcast encryption. In *Advances in Cryptology - EUROCRYPT'98*, number 1403 in LNCS, pages 512–526, Espoo, Finland, 1998. Springer-Verlag.

[13] A.Fiat and M.Naor. Broadcast encryption. In *Advances in Cryptology - CRYPTO'93*, number 773 in LNCS, pages 480–491. Springer-Verlag, 1994.

[14] Y.-H.Chu, S.G.Rao, S.Seshan, and H.Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1456–1471, October 2002.

[15] B.Zhang, S.Jamin, and L.Zhang. Host multicast: A framework for delivering multicast to end users. In *IEEE INFOCOM*, March 2000.

[16] J.Liebeherr, M.Nahas, and W.Si. Application-layer multicasting with delaunay triangulation overlays. *IEEE Journal on Selected Areas in Communications*, 20(8):1472–1488, October 2002.

[17] Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE Infocom*, Anchorage AK, April 2001.