Sender Key Storage Reduction of Secure Multicast Key Management Schemes Using One-Way Function Tree

Mingyan Li, Radha Poovendran*, and David A. McGrew
{myli, radha}@ee.washington.edu, mcgrew@cisco.com

Dept of EE, University of Washington Cisco Systems, Inc.
Seattle WA, 98195-2500 San Jose, CA 95134
Sender Key Storage Reduction of Secure Multicast Key Management Schemes Using One-Way Function Tree

Mingyan Li, Radha Poovendran,∗ and David A. McGrew
{myli, radha}@ee.washington.edu, mcgrew@cisco.com
Dept of EE, University of Washington
Seattle WA, 98195-2500
Cisco Systems, Inc.
San Jose, CA 95134

University of Washington, Dept. of EE, UWEEETR-2001-0002
February 2001

Abstract
Developing scalable infrastructure services for secure multicast communications has been an active research area [1]-[10]. One-way function tree (OFT) [1, 6] is a secure multicast key distribution scheme with logarithmic key update communication overhead in group size \(N\). The OFT scheme has been proposed as a candidate for secure multicast over Internet to the IETF under Multicast Security (MSEC) working group. Though it exhibits good scalability in key update communication, it requires \(O(N)\) cryptographic keys to be stored by the group owner/center (GC). This paper addresses the problem of designing an OFT with minimal GC storage while maintaining the same key update communication growth rate. In order to improve the storage performance, we formulate the problem of GC storage minimization as a constraint optimization problem. By solving this problem, we show that the center storage can be made to grow slower than linear in group size. We also present an explicit design algorithm for storage performance optimized OFT with a given amount of key update communication overhead.

Keywords: Security in Digital Systems; Secure Multicast Communications; Optimization

1 Introduction

Many recent multimedia communication applications are based on group communications. Pay per-view, Satellite and web broadcast of news and movies are some of the well known examples of the group communication models where there is a single sender with multiple receivers. Figure 1 is an abstraction of multicast communication model with a multicast enabled router in the network. When a sender has to deliver identical data to multiple receivers,

![Multicast Communication Model](image_url)

Figure 1: A multicast communication model with a multicast-enabled router.

multicasting is an efficient communication model. It reduces computational overhead of the sender and consumes less network bandwidth [2]. However, the mainstream adoption of multicast communication depends on the ability of the sender in securing the communication so that only the intended end receivers have access to data. The standard

∗This research is funded by NSF grant ANI-0093187
approach of controlling access to communication is to use cryptography. Since identical data is delivered to all the
group members, the owner of the group denoted as the Group Center (GC) can minimize the computation by using a
single cryptographic key called the Session Encryption Key (SEK) (also known as the Traffic Encryption Key (TEK))
and symmetric key encryption. Since the SEK is also used for decryption, every valid receiver has to have the access
to the SEK during the session so it can access data.

Since the SEK is shared by the entire group, any change in group membership requires the SEK to be updated in
order to protect past, present, and future communications. When a member is deleted, the current SEK needs to be
updated to prevent the deleted member from accessing the future communications. Since only valid members need to
have access to the new SEK, there needs to be an additional set of keys (not accessible to the past members) called key
encryption keys (KEK) to encrypt and distribute the new SEK to the valid members of the group.

Thus the problem of controlling the access to multicast communication reduces to the problem of securely distri-
buting KEK’s to ensure only valid members have access to cryptographic keys at any given instant. This is the
multicast key management problem [2]. The solution to this problem should be scalable as a function of group size $N$
in key update communication and key storage, which are two important overheads in multicast key management.

In [3], Harney and Muckenhirn presented the group key management protocol (GKMP) in which each member
shares a unique KEK with the GC. In addition, all members also share a group KEK. The GC thus needs to store
$(N+2)$ KEKs. Although this scheme requires only three KEKs to be stored by a user, under member deletion, the GC
has to individually contact the rest of group members one at a time to update the SEK and the group KEK, resulting
in $(N-1)$ key update messages. In [2], a minimal storage scheme that allows GC to use pseudorandom functions
to generate the KEKs assigned to individual members was presented. This scheme requires the GC to store only a
single seed to generate $N$ KEKs. Thus the minimal storage scheme would reduce the GC key storage requirements of
GKMP from $(N+2)$ to 3. However, the update communication grows as $(N-1)$ for the minimal storage scheme.
The minimal storage scheme can be viewed as a variation of GKMP.

Recently, tree based hierarchical key distribution schemes [7, 9, 10] have emerged as one of the preferred solutions
to the multicast key management problem by IETF. Seminal work based on logical key hierarchy (LKH) was indepen-
dently conducted by Wallner, Harder, and Agee [9] and Wong, Gouda, and Lam [10]. Under this approach, key update
communication and user key storage grow as $O(\log N)$. The user key storage of the LKH was related to the entropy
of member dynamics in [8], which further showed how to design optimal LKH by considering the the average number
of KEKs to be updated under a member deletion. Although the LKH is efficient with respect to user key storage, the
key storage of the sender grows linearly in group size $N$.

One-way function tree (OFT) was proposed by McGrew and Sherman in [6] and by Balenson, McGrew, and
Sherman in [1] to further reduce the update communication of the LKH by a factor of two for binary trees, and a factor
of $\frac{a}{n-1}$ for $a$-ary trees. Unlike LKH where the keys are assumed to be independent, the keys on OFT are related by
one-way function [1, 6] with a computational scheme that allows the derivation of a high level node key from keys of
all its children nodes, hence reducing the rekey messages per update. However, the key storage requirement of the GC
in OFT grows linearly with group size $N$.

In order to further reduce the GC storage of tree based key distribution schemes, a hybrid model which combines
the LKH and minimal storage scheme was presented in [2] by Canetti, Malkin, and Nissim. They made use of the
tradeoff between the key update communication and the storage of the GC and noted that if the cluster size is $O(\log N)$,
the storage of the GC reduces from $O(N)$ to $O\left(\frac{N}{\log N}\right)$. Their approach does not show how to minimize key storage
of GC for a prespecified amount of key update communication overhead.

A design approach that makes use of an upper bound on the key update communication is useful for applications
where energy and/or bandwidth are limited. Such an approach will enable the designer to prespecify the amount of
storage or update communication that can be tolerated by the application. In such specifications, the aim should be
to perform at least of the same order as LKH since it requires $O(\log N)$ update communication for a group of size $N$. In our recent work we have addressed the problem of center storage reduction of the hybrid tree using LKH, for a
prespecified update communication upper bound [4, 5].

In this paper, we address the problem of minimizing the GC key storage of the OFT while preserving the loga-

rithmic user key storage and key update communication. We show that the problem of key storage minimization with
a specified communication budget can be posed as a constraint optimization problem, and then present an explicit
design for a hybrid OFT. We also compare the LKH and OFT schemes.

This paper is organized as follows. The OFT is reviewed in section 2 and the hybrid structure of the OFT is

---

1The GC has to store $\frac{a^N-1}{a-1}$ for an $a$-ary tree.
proposed in section 3. We then derive an analytical model of the hybrid OFT. In section 4, we formulate the key storage minimization problem as a constraint optimization problem, and show that the problem can be converted into a fixed point equation. Based on the solution, a design algorithm is summarized. Section 5 presents numerical design examples.

2 Review of One-Way Function Tree

McGrew and Sherman [6] and Balenson et al. [1] proposed a key management scheme by constructing an OFT. Figure 2 illustrates a binary one-way function distribution tree for a group of seven members with the member M₈ requesting to join the group.

![Figure 2: A Binary OFT for A Group of Seven Members with the Eighth Member M₈ Requesting Join.](image)

2.1 Distribution of Keys in OFT

Every member is uniquely assigned to a leaf node on the tree, thus fixing the number of leaves to be the group size N. For every node x in the tree, there are two associated keys, an unblinded node key Kₓ, and a blinded node key K′ₓ. The blinded node key is the output of one way function g(·) with the unblinded node key as the input, i.e., K′ₓ = g(Kₓ). Unblinded keys are used to encrypt and decrypt rekey messages, and blinded keys allow computing unblinded keys of high levels from lower levels, thus reducing the number of new keys the GC has to encrypt and multicast during key updates.

The OFT works as follows. The group controller (GC) generates an unblinded key for each leaf node, and the unblinded keys of interior nodes are computed by applying a mixing function f(·) to the blinded keys of all its children. Explicitly, Kₓ is computed from a children of node x as:

\[ Kₓ = f(K'_{child₁}, K'_{child₂}, ..., K'_{childₐ}) \]

where a is the degree of the tree. For example, given the keys K₃,₁ and K₃,₂, member M₁ in Figure 2 can compute K₂,₁ as K₂,₁ = f(g(K₃,₁), K₃,₂). If a member is given the unblinded leaf key and the blinded keys of the siblings of every node along the path from its leaf to the root, the member will be able to compute all the unblinded keys along its path to the root to decrypt necessary rekey messages. Since there are (a − 1) siblings to a node on each level and the height of a tree is \( \log_a N \), a member needs to store \([1 + (a − 1) \log_a N]\) keys. For example, member M₁ has to store the set of KEKs \{K₃,₁, K₂,₂, K₃,₂, K₁,₂\}.

2.2 Member Addition in OFT

A new member has to be assigned to a unique leaf and a new unblinded key needs to be associated with that node. The GC computes the keys along the new member’s key path and multicasts the new blinded keys to the current

---

2Because of space limit, only unblinded keys are shown in Figure 2.
siblings along the path. The new member is given necessary keys using unicast. For example, in Figure 2, if a new member \( M_8 \) is added into the group, the GC generates \( K_{3,8} \) and recomputes interior keys. It broadcasts \( \{K_{3,8}'\}K_{3,7}, \{K_{2,4}'\}K_{2,3}, \{K_{1,2}'\}K_{1,1} \), where the notation \( \{m\}_K \) denotes the encryption of message \( m \) with key \( K \). Member \( M_7 \) decrypts \( K_{3,8}' \) to recompute \( K_{2,4}, K_{1,2} \) and \( K_0 \); member \( M_5 \) and \( M_6 \) decrypt \( K_{2,4}' \) to compute new \( K_{1,2} \) and new \( K_0 \). Therefore, the key update communication for a member addition is the number of siblings on each level times the height of the tree, given as \((a-1) \log_a N\).

### 2.3 Member Deletion in OFT

When a member is deleted from the group, all the interior keys he once shared with other members have to be updated. The GC computes new blinded keys and send the keys to proper subgroups. The key update communication for a member deletion is also given as \((a-1) \log_a N\).

### 2.4 GC Storage in OFT

Since the GC needs to store only the \( N \) unblinded keys of the leaf nodes, the storage is \( N \), which is independent of the degree of the tree. Note that LKH has the GC storage as \( a\frac{a^{N-1}}{a-1} \), which is a function of the degree of the tree. We now use the hybrid model to minimize the center key storage.

### 2.5 Comparison between LKH and OFT

Both the LKH and the OFT have a tree structure, and the height of the tree determines the user storage and the key update communication as \( \mathcal{O}(\log N) \) for both the schemes. The GC storage for both schemes is related to the group size \( N \) as \( \mathcal{O}(N) \). However, the LKH and the OFT are different in the way the keys are computed and stored. The keys on a LKH tree are generated independently, and the GC has to store all the keys of a tree. Hence, the GC storage of the LKH depends on the height, which is a function of the tree degree \( a \) and the group size \( N \). In contrast, in OFT, given all the leaf keys at the bottom level of the tree, the GC can derive all other keys on the tree. Therefore, the GC in OFT only stores all the leaf keys and the GC storage is independent of the degree of the tree. Hence the GC storage in OFT cannot be optimized with respect to the degree of the tree.

### 3 Towards Performance Optimization of One-Way Function Tree

In order to improve the performance of OFT, we note that the GC storage in the OFT scheme is equal to the number of leaves in the tree. If the number of leaves can be set as a variable, then we can control the GC storage. One approach [2, 4] is to assign multiple members to a leaf node. Then by controlling the number of members assigned to a leaf node, we can vary the number of leaves and thus the GC storage.

The main idea is to divide the group of size \( N \) into clusters of size \( M \) with every cluster assigned to a unique leaf node. Then there are \( \frac{N}{M} \) clusters (also leaves), and we need to build a tree of depth \( \log_a \frac{N}{M} \). Figure 3 illustrates a binary hybrid OFT with cluster size \( M = 3 \) and a group of 12 members.

We notice that the structure in Figure 3 consists of two parts, the OFT, and the clusters. The OFT scheme is used as inter-cluster key management scheme to limit key update communication, and the minimal storage is used as the intra-cluster scheme to reduce GC storage requirement. The combined scheme is called hybrid OFT. We note that we do not change the way keys are computed in OFT.

In the hybrid tree presented in Figure 3, a user needs to store \( (1 + (a-1) \log_a \frac{N}{M}) \) KEK’s required by the OFT scheme, plus one KEK required by the minimal storage scheme within the cluster. When a member is deleted, the total number of key update messages, denoted by \( C \), is \((a-1) \log_a \frac{N}{M} \) within the tree plus \((M-1)\) within the cluster, leading to:

\[
C = M - 1 + (a-1) \log_a \frac{N}{M} \tag{1}
\]

The number of keys stored by the GC, computed as the number of leaves of the hybrid OFT plus seeds for \( \binom{N}{M} \) clusters, is

\[
S = \frac{2N}{M} \tag{2}
\]
Since the OFT scheme has logarithmic update communication [1, 6], in the hybrid OFT, we want to keep the update communication as $O(\log N)$ except some scale factor $\beta$. This can be expressed as:

$$M - 1 + (\alpha - 1) \log_{\alpha} \frac{N}{M} \leq \beta \log_{\alpha} N,$$

where the communication scale factor $\beta$ indicates how much communication can be allotted for key updates. The choice of parameter $\beta$ should satisfy the inequality given later in (6). We note that as in the case of original OFT [1, 6] minimization of (1) with respect to $\alpha$ shows that the key update communication in hybrid OFT is minimized for binary trees.

### 4 Minimization of Key Storage with Communication Constraint

In the hybrid OFT scheme, the storage and the update communication are functions of the cluster size $M$. The selection of $M$ should be such that the update communication scales at least of the order of $O(\log N)$. Hence the optimization problem is posed as

$$\min \left( \frac{2N}{M} \right) \text{ w.r.t. } M$$

subject to the communication constraint given in (3).

The following theorem presents the solution to this constraint optimization problem.

**Theorem 1:** Optimal cluster size $M$ that minimizes the storage function $S = \frac{2N}{M}$ while satisfying the update communication budget $C = M - 1 + (\alpha - 1) \log_{\alpha} \frac{N}{M} \leq \beta \log_{\alpha} N$ is obtained by the largest root of the equation

$$M - \lambda \ln M + \lambda \ln N - 1 = \beta \log_{\alpha} N,$$

where $\lambda = \frac{(\alpha - 1)}{\ln \alpha}$ and $\mu = 1 + (\beta - \alpha + 1) \log_{\alpha} N$.

**Proof:** Since the storage is a monotonically decreasing function of $M$, the largest value of $M$ satisfying the update communication constraint will be the solution of this constraint optimization. Hence, the optimal value of the cluster size is computed by the equation:

$$M - \lambda \ln M + \lambda \ln N - 1 = \beta \log_{\alpha} N$$

The update communication, given in (1) and also in the left-hand side of (5), is a convex function of $M$ and attains its minimum value $[\lambda \left( 1 + \frac{N}{\lambda} \right) - 1]$ at $M = \lambda$. Hence the factor $\beta$ should satisfy the following inequality in order to solve equation (5),

$$\beta \geq \left( \frac{\lambda(1+ \ln \frac{N}{\lambda}) - 1}{\log_{\alpha} N} \right)$$

After some algebra, it can be shown that for large values of $N$, the asymptotic lower bound of $\beta$ approaches $(\alpha - 1)$.
Equation (5) can be rewritten as

$$M - \lambda \ln M = \mu \tag{7}$$

where $\mu = 1 + (\beta - a + 1) \log_a N$.

**Design Solution for OFT**

The fixed point equation (7) is a contraction mapping with the largest root as the solution. We set the initial value of $M$ to be $M_0 = \mu$. After some algebra, a series approximation to $M$ is given by

$$M = \mu \prod_{i=1}^{\infty} \left(1 + \left(\frac{\lambda}{\mu}\right)^i \ln \mu\right), \tag{8}$$

The asymptotic value of $M$, denoted by $M_\infty$, is given by

$$M_\infty = \lim_{N \to \infty} \mu \prod_{i=1}^{\infty} \left(1 + \left(\frac{\lambda}{\mu}\right)^i \ln \mu\right) = 1 + (\beta - a + 1) \log_a N + \lambda \ln[(\beta - a + 1) \log_a N] \approx 1 + (\beta - a + 1) \log_a N \tag{9}$$

For large values of $N$, the largest root of the equation (5) converges to $M_\infty$, and grows as $O(\log N)$.

We can derive the same solution using Newton’s method. By setting $M_0 = \mu$, the first-order approximation is $M_1 = \mu + \lambda \ln \mu$. Letting $N \to \infty$ yields the same solution as (9).

The corresponding GC storage for $N \to \infty$, denoted by $S_\infty$, is

$$S_\infty = \frac{2}{(\beta - (a - 1)) \left(\frac{N}{\log_a N}\right)} \tag{10}$$

Hence, the constraint optimization leads to the optimal growth of the GC storage as $O\left(\frac{N}{\log N}\right)$ which is far better than $O(N)$ growth, when the update communication is constrained to grow as $O(\log N)$. We note from (8) that accuracy of computation can be increased to any desired amount. An explicit design procedure based on our design solution is given below.

**Hybrid OFT Design Steps:**

1. Initial design data: group size $N$, communication scale factor $\beta$, and choose $a = 2$ if not specified
2. Check the condition given in (6). If satisfied, go to step 3. Otherwise the design is not feasible
3. Compute the first order approximation of the optimal cluster size $M$ using (9)
4. Construct a hybrid tree of degree $a$ and cluster size $M$

## 5 Design Examples

As illustration, we set $\beta = (1 + \lambda) \ast \ln a$ which leads to $M_\infty = \ln N$ and $S_\infty = \frac{2N}{\ln N}$. As a specific design example, we choose the group size $N = 1000$, the communication budget factor $\beta = 2$, and the degree of the tree $a = 2$. Using (9), the cluster size $M$ is computed to be 10. A binary tree with cluster size 10 requires 200 keys to be stored in the GC, while the update communication is about $1.6 \log_2 N$, with the factor less than 2. Table 1 presents a numerical comparison between the OFT and the hybrid OFT in terms of key storage for several pairs of (degree $a$, group size $N$).

From the column 4 of the table, we note that the optimal cluster size $M_\infty$ can lead to significant improvements in GC storage over values obtained in [1, 6] for a given communication budget.
Table 1: Comparison of GC Storage of OFT and Hybrid OFT Schemes

<table>
<thead>
<tr>
<th>(Degree a, Group size N)</th>
<th># of keys in GC by OFT</th>
<th># of keys in GC by Eq (10)</th>
<th>Storage reduction in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2, 2^10)</td>
<td>1024</td>
<td>205</td>
<td>80.0</td>
</tr>
<tr>
<td>(2, 2^20)</td>
<td>1048576</td>
<td>104858</td>
<td>90.0</td>
</tr>
<tr>
<td>(3, 2^10)</td>
<td>1024</td>
<td>325</td>
<td>68.3</td>
</tr>
<tr>
<td>(3, 2^20)</td>
<td>1.07 x 10^9</td>
<td>1.13 x 10^8</td>
<td>89.4</td>
</tr>
<tr>
<td>(4, 2^10)</td>
<td>1024</td>
<td>410</td>
<td>60.0</td>
</tr>
<tr>
<td>(4, 2^20)</td>
<td>1048576</td>
<td>209715</td>
<td>80.0</td>
</tr>
</tbody>
</table>

6 Conclusions

In this paper, we used a hybrid tree approach to minimize the key storage of the sender in OFT based multicast key distribution. We derived the optimal storage using a fixed point equation and showed that the center storage of the hybrid OFT can be reduced from $O(N)$ to $O(\frac{N \log N}{\log a})$. Based on our results, we presented an explicit design algorithm for a hybrid OFT with a predefined amount of key update communication. Our approach does not change the security/insecurity of the keys computed by original OFT. Moreover, our approach is applicable to a more secure version of OFT called OFC [1].

References


