Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing Letters 93 (2005) 191–198

Information Processing Letters

www.elsevier.com/locate/ipl

# Minimizing center key storage in hybrid one-way function based group key management with communication constraints

Mingyan Li [a], Radha Poovendran [a,*,1], David A. McGrew [b]

[a] *Department of Electrical Engineering, University of Washington, Seattle, WA 98195, USA*
[b] *Cisco Systems, Inc. San Jose, CA 95134, USA*

Received 1 October 2001; received in revised form 30 August 2004

Available online 13 December 2004

Communicated by Y. Desmedt

**Abstract**

We study the problem of designing a storage efficient secure multicast key management scheme based on one-way function trees (OFT) for a prespecified key update communication overhead. Canetti, Malkin and Nissim presented a hybrid model that divides a group of $N$ members into clusters of $M$ members and assigns each cluster to one leaf node of a key tree. Using the model, we formulate a constrained optimization problem to minimize the center storage in terms of the cluster size $M$. Due to the monotonicity of the center storage with respect to $M$, we convert the constrained optimization into a fixed point equation and derive the optimal $M^*$ explicitly. We show that the asymptotic value of the optimal $M^*$, given as $\mu + \frac{a-1}{\log_e a} \log_e \mu$ with $\mu = \mathrm{O}(\log N)$ and $a$ being the degree of a key tree, leads to the minimal storage as $\mathrm{O}(\frac{N}{\log N})$, when the update communication constraint is given as $\mathrm{O}(\log N)$. We present an explicit design algorithm that achieves minimal center storage for a given update communication constraint.
© 2004 Elsevier B.V. All rights reserved.

*Keywords:* Cryptography; Key management/distribution; Group rekey; Secure multicast/broadcast; Optimization; One-way function

## 1. Introduction and background

Developing scalable infrastructure services for secure multicast and secure broadcast communications is an active research area [1–14]. The group owner/center (GC) can share a single cryptographic key called the Session Encryption Key (SEK) with the entire group and use symmetric key encryption (for minimal computation) to distribute data securely to all intended group members. Whenever there is a change in membership, the current SEK becomes invalid and needs to be updated. To encrypt and distribute the new SEK to the valid members of the group, an additional set of keys called key encryption keys (KEKs) is used.

[*] Corresponding author.
*E-mail addresses:* myli@ee.washington.edu (M. Li), radha@ee.washington.edu (R. Poovendran).
[1] This work is supported by CAREER grant from NSF ANI-0093187 and YIP from ARO under Cooperative Agreement DAAD19-02-1-0242.

0020-0190/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2004.10.012

Thus, the problem of controlling the access to multicast and broadcast communications reduces to the problem of distributing KEKs securely to ensure that only valid members have access to cryptographic keys at any given instant. This is the group key management/distribution problem [2].

Two of the most important efficiency parameters in multicast key management are key update communication and key storage [2,13,14]. Hence, any candidate solution to a key management problem should be scalable in both these parameters, as a function of group size $N$. Key update communication is measured as the number of rekey messages sent by the GC, in order to update SEK and exposed KEKs, and key storage as the number of keys stored in the GC (center storage) and in a member (user storage). While there are other efficiency parameters for a secure multicast model, such as latency, computational cost of the GC and members [11,14], we focus on key storage and update communication in this paper. We also abstract away some implementation details, such as key tree maintenance, message format, and choice of encryption algorithms.

Tree-based key distribution schemes, including logical key hierarchy (LKH) [13,14], one-way function tree (OFT) [1,11], and one-way function chain (OFC) [3], have emerged as the preferred solutions to the multicast key management problem, due to their scalability properties. Both update communication and user key storage grow as $O(\log N)$ in these models, while the center storage increases $O(N)$. Canetti, Malkin, and Nissim [2] proposed a hybrid model which combines LKH and a minimal storage scheme to reduce the GC storage from $O(N)$ to $O(\frac{N}{\log N})$ and observed that the product of update communication and the GC storage is $\Theta(N)$. Their approach however does not address how to minimize center storage when the key update communication is bounded a priori.

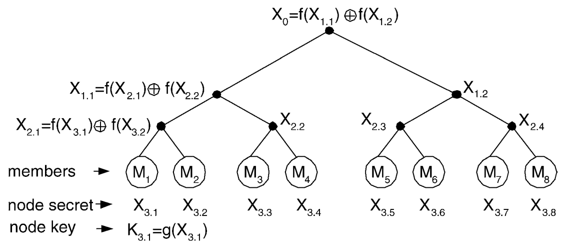
Our main contribution in this paper is that for hybrid trees, we formulate the center storage computation for a given communication bound as a constrained optimization problem. Based on our formulation, we derive a fixed point equation for the optimal cluster size. An upper bound on update communication is a common restriction for applications where energy and/or bandwidth are limited. Our approach will enable the designer to specify the amount of key update communication overhead that can be tolerated by the application. The need for the center storage reduction was discussed in [2] and [11]. The reduction of the center storage enables keys stored in the GC to be loaded into RAM [2], and hence be accessed faster. The manager's node (center) storage is identified as one of the two bottlenecks that limit the maximum possible group size [11]. Thus, minimization of the GC storage will help increase the maximum supported group size.

We note that although we make use of the hybrid tree model proposed by Canetti, Malkin and Nissim in [2], our approach differs in that we formulate the design of a hybrid key tree as a constrained optimization problem, and derive the optimal solution by solving a fixed-point equation. Unlike [2], we provide an analytical technique to select an optimal design parameter that trades off between storage and rekey communication overhead.

## 2. Hybrid one-way function trees

### 2.1. One-way function trees

Sherman and McGrew [11] and Balenson et al. [1] proposed a key management scheme by constructing an OFT. Fig. 1 illustrates a binary 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 $n$ in the tree, there is a node secret $X_n$ and a node key $K_n$ with $K_n = g(X_n)$, where $g(\cdot)$ is the right half of a length-doubling pseudorandom function $H(\cdot)$ [11]. The root secret, $X_0$, is the group key. Node secrets are used to derive secrets of higher levels from



$X_0=f(X_{1.1})\oplus f(X_{1.2})$
$X_{1.1}=f(X_{2.1})\oplus f(X_{2.2})$
$X_{1.2}$
$X_{2.1}=f(X_{3.1})\oplus f(X_{3.2})$
$X_{2.2}$
$X_{2.3}$
$X_{2.4}$
members
$M_1$ $M_2$ $M_3$ $M_4$ $M_5$ $M_6$ $M_7$ $M_8$
node secret
$X_{3.1}$ $X_{3.2}$ $X_{3.3}$ $X_{3.4}$ $X_{3.5}$ $X_{3.6}$ $X_{3.7}$ $X_{3.8}$
node key
$K_{3.1}=g(X_{3.1})$

Fig. 1. A binary OFT tree with 8 members. Each node $n$ in an OFT is associated with an unblinded node secret $X_n$ and a node key $K_n$, where $K_n = g(X_n)$ with $g(\cdot)$ being the right half of a length-doubling pseudorandom function $H(\cdot)$ [11]. Node keys are not shown in the figure except for one leaf node. $f(X_n)$ is blinded node secret, where $f(\cdot)$ is the left half of $H(\cdot)$. The node secret of an interior node and the root are computed as XOR of blinded node secrets of all its children. For example, $X_{2.1} = f(X_{3.1}) \oplus f(X_{3.2})$.

lower levels, and node keys are used to encrypt and decrypt rekey messages. There are two versions of a node secret: unblinded node secret $X_n$ and blinded node secret $f(X_n)$, where $f(\cdot)$ is the left half of $H(\cdot)$ [11].

A member is assigned the unblinded node secret of its associated leaf and all blinded secrets of the siblings of every node along the path from its leaf to the root. Therefore, it can compute all the unblinded node secrets and thus node keys along its path to the root and decrypt necessary rekey messages. In an $a$-ary ($a > 1$) key tree, there are $(a-1)$ siblings to a node on each level and the height of the tree is $\log_a N$, hence, a member needs to store $[1 + (a-1)\log_a N]$ keys. For example, member $M_1$ in Fig. 1 has to store $\{X_{3.1}, f(X_{3.2}), f(X_{2.2}), f(X_{1.2})\}$.

The rekey operation after a member revocation is more expensive than that of a member addition [14], hence we consider only member revocation when evaluating the key update communication overhead. In an OFT scheme, if a member is revoked from the group, its leaf secret becomes invalid and needs to be updated, and all node secrets along the path from its leaf to the root have to be recomputed. For example, when $M_1$ in Fig. 1 leaves the group, the GC chooses a new node secret $X'_{3.1}$, and computes new blinded node secrets $f(X'_{3.1}), f(X'_{2.1}), f(X'_{1.1})$. To enable the remaining valid members to obtain the necessary new node secrets, the GC broadcasts $\{E_{K_{3.2}}(f(X'_{3.1})), E_{K_{2.2}}(f(X'_{2.1})), E_{K_{1.2}}(f(X'_{1.1}))\}$, where $E_K(m)$ denotes the encryption of the message $m$ using key $K$. The number of rekey messages with one key per message is $(a-1)\log_a N$.

Note that the GC needs to store at least the node secrets of every member in an OFT scheme, and hence by adjusting the number of leaves in the tree, we can control the GC storage. One approach [2,8] is to assign multiple members to a leaf node of an $a$-ary tree, to reduce the number of tree leaves and thus GC storage.

## 2.2. Hybrid trees

In order to reduce the GC storage, authors in [2] proposed to divide the group of $N$ members into clusters of size $M$ with each cluster assigned to a unique leaf node of an OFT. Then there are $\lceil N/M \rceil$ clusters, and we need to build an OFT of height $\log_a \lceil N/M \rceil$. For simplicity, we assume $N$ is a multiple of $M$. Fig. 2 illustrates a binary hybrid OFT. We notice that the hybrid OFT structure consists of two parts, a base OFT,[2] and clusters. An OFT scheme is used as an inter-cluster key management scheme to limit key update communication, and a minimal storage scheme [2] is used as an intra-cluster scheme to reduce GC storage requirement.

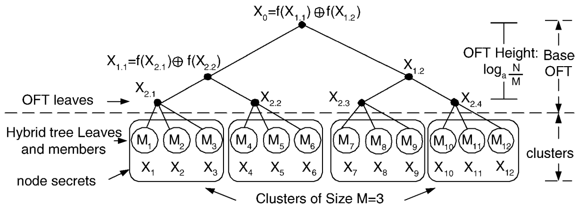

$X_0=f(X_{1.1}) \oplus f(X_{1.2})$
$X_{1.1}=f(X_{2.1}) \oplus f(X_{2.2})$
$X_{1.2}$
$X_{2.1}$
$X_{2.2}$
$X_{2.3}$
$X_{2.4}$
OFT leaves
Hybrid tree Leaves and members
node secrets
$M_1$ $M_2$ $M_3$ $M_4$ $M_5$ $M_6$ $M_7$ $M_8$ $M_9$ $M_{10}$ $M_{11}$ $M_{12}$
$X_1$ $X_2$ $X_3$ $X_4$ $X_5$ $X_6$ $X_7$ $X_8$ $X_9$ $X_{10}$ $X_{11}$ $X_{12}$
Clusters of Size M=3
OFT Height: $\log_a \frac{N}{M}$
Base OFT
clusters

Fig. 2. A binary hybrid OFT with group size $N = 12$ and cluster size $M = 3$. A hybrid OFT consists of a base OFT tree (everything above the dotted line) and clusters (below the dotted line). Each cluster assigned to one leaf node of the OFT. A minimal storage key management scheme is used within each cluster. Note that there are 4 leaf nodes in the base OFT, but 12 leaf nodes in the hybrid tree.

In the hybrid tree presented in Fig. 2, a user needs to store $(1 + (a-1)\log_a \frac{N}{M})$ node secrets required by the base OFT scheme, plus one KEK required by the minimal storage scheme within the cluster. For example, member $M_1$ in Fig. 2 stores $\{X_{2.1}, f(X_{2.2}), f(X_{1.2})\}$ as required by the base OFT and $X_1$ by the minimal storage scheme. When $M_1$ is deleted, the keys along the path from its base OFT leaf to the root need to be updated. The GC chooses a new $X'_{2.1}$ and broadcasts $\{E_{K_2}(X'_{2.1}), E_{K_3}(X'_{2.1})\}$ to update keys in the cluster, and $\{E_{K_{2.2}}(f(X'_{2.1})), E_{K_{1.2}}(f(X'_{1.1}))\}$ for the base OFT. Therefore, the total number of key update messages per member leaving, denoted by $C$, is $(a-1)\log_a \frac{N}{M}$ for the base OFT plus $(M-1)$ for the cluster, leading to:

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

In a minimal storage scheme, the GC uses a secret key, which we call a seed, as an index of a pseudo-random function to generate keys for each user. Therefore, in a hybrid OFT, the number of keys stored by the

[2] In the rest of the paper, we refer to the OFT used in a hybrid model as a base OFT, and to the original OFT scheme as the OFT scheme.

GC, denoted by $S$, computed as the number of leaves of the base OFT plus seeds for $(N/M)$ clusters, is

$$S = \frac{2N}{M}. \tag{2}$$

## 3. Minimization of center storage under communication constraint

We want to design a hybrid OFT with the minimal center storage by choosing the optimal cluster size $M$, while keeping the update communication under a given constraint. Based on the computation of the key update communication (1) and center key storage in (2), we formulate the constrained optimization as:

$$\text{minimize}\left(\frac{2N}{M}\right) \quad \text{w.r.t. } M \tag{3}$$

subject to the communication constraint:

$$M - 1 + (a-1)\log_a \frac{N}{M} \leqslant \beta(N), \tag{4}$$

where $\beta(N)$ is the constraint on the number of rekey messages per update and is an application dependent design parameter. We note that $\beta(N)$ is general to represent an arbitrary constraint on the rekey messages. However, to have feasible solutions, $\beta(N)$ must satisfy the inequality given later in (6).

Note that the constrained optimization given by (3) and (4) allows explicit derivation of the optimal cluster size for a given communication constraint, however, such derivation is not readily attainable from [2] where the product of the GC storage and update communication is given as $\Theta(N)$.

**Theorem 1.** *The optimal cluster size $M$ that minimizes the storage function $S = \frac{2N}{M}$, while satisfying the update communication budget $C = M - 1 + (a-1)\log_a \frac{N}{M} \leqslant \beta(N)$, is obtained by the largest root of the equation $M - \lambda \ln M = \mu$, where $\lambda = \frac{(a-1)}{\ln a}$ and $\mu = 1 + \beta(N) - (a-1)\log_a N$.*

**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. Hence, the optimal value of the cluster size is computed by

$$M^* - \lambda \ln M^* + \lambda \ln N - 1 = \beta(N). \tag{5}$$

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

$$\beta(N) \geqslant \left[\lambda\left(1 + \ln \frac{N}{\lambda}\right) - 1\right] \tag{6}$$

$$\geqslant \log_2 N + 1.914. \tag{7}$$

The equality of (7) is achieved at $a = 2$. After some algebra, it can be shown that for large values of $N$, the asymptotic lower bound of $\beta(N)$ approaches $(a-1) \cdot \log_a N$. Eq. (5) can then be rewritten as:

$$M^* - \lambda \ln M^* = \mu. \qquad \square \tag{8}$$

**Solution to OFT design problem.** The fixed-point equation (8) is a contraction mapping in the range of interest $[\lambda, \infty]$. We set the initial value of $M$ to be $M_0 = \mu$. After some algebra, a series approximation to $M$ is given by:

$$M^* = \mu \prod_{i=1}^{\infty} \left(1 + \left(\frac{\lambda}{\mu}\right)^i \ln \mu\right). \tag{9}$$

The asymptotic value of $M$ when $N \to \infty$, denoted by $M_\infty$, is given by:

$$\begin{aligned} M^*_\infty &= \lim_{N\to\infty} \mu \prod_{i=1}^{\infty} \left(1 + \left(\frac{\lambda}{\mu}\right)^i \ln \mu\right) \\ &= \lim_{N\to\infty} \left[\mu + \lambda \ln \mu + \mathrm{O}\left(\frac{\ln \mu}{\mu}\right)\right] \\ &= \mu + \lambda \ln \mu \\ &= 1 + \beta(N) - (a-1)\log_a N \\ &\quad + (a-1)\log_a\left[1 + \beta(N) - (a-1)\log_a N\right]. \end{aligned} \tag{10}$$

When $N \to \infty$, the largest root of Eq. (5) converges to $M_\infty$, and grows as $\mathrm{O}(\log N)$. 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 (10).

The asymptotic optimal cluster size $M^*_\infty$ (10) is monotonically decreasing with respect to $a$ when $\mu \leqslant N$. First, we note that $\lambda = \frac{(a-1)}{\ln a} > 0$ is increasing with $a$ and $\mu = 1 + \beta(N) - \lambda \ln N > 0$ is decreasing with $a$, i.e., $\frac{\partial \lambda}{\partial a} > 0$ and $\frac{\partial \mu}{\partial a} < 0$. When $\mu \leqslant N$, which in most cases holds due to the fact that the communication budget $\beta(N)$ is normally stringent, then

$$\begin{aligned}\frac{\partial M^*_\infty}{\partial a} &= \frac{\partial \mu}{\partial a} + \frac{\partial \lambda}{\partial a}\ln\mu + \frac{\lambda}{\mu}\frac{\partial \mu}{\partial a}\\ &< \frac{\partial \mu}{\partial a} + \frac{\partial \lambda}{\partial a}\ln\mu\\ &= -\frac{\partial \lambda}{\partial a}\ln N + \frac{\partial \lambda}{\partial a}\ln\mu \leqslant 0.\end{aligned}$$

Therefore, $a = 2$ maximizes $M^*_\infty$ and thus minimizes the center storage when $\mu \leqslant N$.

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

$$S_\infty = \frac{2N}{\mu + \lambda\ln\mu}\begin{cases} = 2N & \text{if } \beta(N) = \lambda\ln N,\\ \approx \frac{2N}{(\beta(N)-(a-1)\log_a N)} & \text{otherwise.}\end{cases} \tag{11}$$

We note that for $\beta(N) = \lambda\ln N = (a-1)\log_a N$, which is the communication cost for an OFT, the $S_\infty$ of a hybrid OFT is $2N$, which agrees with the GC storage in an OFT. When the update communication of the hybrid OFT is constrained to grow the same order as an OFT, i.e., $\beta(N) = \mathrm{O}(\log N)$, the constraint optimization leads to the optimal growth of the GC storage as $\mathrm{O}(\frac{N}{\log N})$ which is far better than $\mathrm{O}(N)$ growth.

The hybrid model presented in [2] achieves the same level of scalability without optimization. However, [2] did not present any explicit formula to compute the optimal cluster size as we have in (9). We now present an explicit design procedure based on our design solution.

**Hybrid OFT design steps.**

(1) Initial design data: group size $N$, maximum allowable rekey messages per update $\beta(N)$, and degree of the tree $a$ (chosen to be $a = 2$ if not specified since a binary OFT and $M^*_\infty$ jointly minimize the GC storage).
(2) Check the condition given in (6). If satisfied, go to step 3. Otherwise, the design is not feasible.
(3) Compute the optimal cluster size $M$ using (9), where $\lambda$ and $\mu$ are defined in Theorem 1.
(4) Construct a hybrid tree of degree $a$ and cluster size $M$.

## 4. Design examples

As a design example, we have a group size of $N = 1,000,000$, and a constraint on the number of rekey messages $\beta(N) = 40$ and choose the degree of the tree $a = 2$. Using $M = \mathrm{O}(\log N)$ given in [2] and choosing base to be 2, the computed cluster size $M$ is approximately 20. Based on (1) and (2), a binary tree with cluster size 20 requires 100,000 keys to be stored in the GC, while the number of rekey messages is about 35, which is less than 40. If we use (9) to calculate the optimal cluster size, we have $M^* = 25$ and the GC storage is 80,000 keys with 40 messages per update. We note the optimal cluster size $M^* = 25$ achieves further reduction in GC storage by 20% compared to [2], while maintaining the communication constraint.

Table 1 presents a numerical comparison of GC storage and key update communication between an OFT, the hybrid OFT [2], and the hybrid OFT using

Table 1
The GC storage reduction and key update communication increase of hybrid OFT schemes with asymptotic optimal cluster size compared to OFT schemes, and hybrid OFT schemes in [2] for different initial design data (degree $a$, group size $N$, communication (comm.), and maximum (max.) allowable rekey messages per update $\beta(N)$)

| Degree $a$, group size $N$, max. allowable messages $\beta(N)$ | Hybrid OFT with optimal cluster size $M^*$ | | | Comparison with OFT | | Comparison with hybrid OFT [2] $M = \mathrm{O}(\log N)$, base chosen to be $a$ | |
|---|---|---|---|---|---|---|---|
| | $M^*_\infty$ by (10) | GC storage | update comm. | GC storage reduction | update comm. increase | GC storage reduction | update comm. increase |
| $(2, 2^{10}, 17)$ | 10 | 17 | 188 | 90.0 | 36.2 | 8.4 | 4.8 |
| $(2, 2^{20}, 34)$ | 18 | 34 | $1.12 \times 10^5$ | 94.4 | 38.5 | 6.6 | 3.4 |
| $(3, 2^{10}, 19)$ | 11 | 19 | 175 | 90.9 | 30.8 | 46.1 | 22.7 |
| $(3, 2^{30}, 59)$ | 27 | 59 | $7.8 \times 10^7$ | 96.3 | 34.6 | 30.7 | 15.4 |
| $(4, 2^{10}, 22)$ | 12 | 22 | 167 | 91.7 | 27.2 | 59 | 26.0 |
| $(4, 2^{20}, 44)$ | 20 | 44 | $1.01 \times 10^5$ | 95.0 | 29.4 | 51 | 21.1 |

the asymptotic optimal cluster value (10) for different sets of parameters: degree $a$ and group size $N$. To illustrate, we set $\beta(N) = (1+\lambda)\ln N$ which leads to $M^*_\infty = 1 + \ln N + \lambda \ln(1+\ln N)$ and $S_\infty \approx \frac{2N}{\ln N}$. From columns 5, 6, 7, and 8 of the table, we note that the asymptotic optimal cluster size $M_\infty$ can lead to over 90% reduction in GC storage compared to the values obtained in [1], and on average 30% reduction compared to [2], with update communication increase of about 30 and 15%, respectively.

## 5. Related work and new developments

### 5.1. Hybrid models

Heyman et al. considered another hybrid tree model with clusters of $M$ members assigned to leaf nodes of a base key tree [5]. The intra-cluster scheme in [5] is a power set key management that assigns one key for each possible subset of $M$ members, resulting in $2^M - 1$ keys for $M$ members and $2^{M-1}$ for each member. The amount of update communication is reduced to $1 + (a-1)\log_a \frac{N}{M}$, because only one message within the cluster is needed to inform the remaining members of the ID of the key to be used for revocation of one or multiple members. However, key storage requirements for both the center and the users increases exponentially with respect to cluster size. In our study, we minimize the center storage while increasing the communication cost by only a constant factor.

A hybrid tree model that trades off collusion resilience for reduction in key storage and key update communication was proposed in [4], while we assume perfect collusion resistance. Members are categorized into static and dynamic based on their duration of membership and level of trust [10]. Each static member serves as subgroup manager for a cluster of dynamic members, to achieve better "1 *affects n scalability*" [10] and to alleviate the management of mass joins and leaves of dynamic members. We do not assume such heterogeneity in member dynamics and trust.

### 5.2. Security of OFT

Recently, after the submission of our paper, Horng [6] identified a security weakness on the OFT [1,11]. Horng demonstrated that when an evicted member colludes with the next joined member, they can recover the group key used between the deletion and the addition. Ku and Chen [7] found out that such a vulnerability is due to the fact that the blinded secrets known to a removed member will be used in the future group key establishment and proposed a remedy that invalidates all the blinded secrets known to a deleted member by changing the leaf secrets. The improved OFT achieves collusion resilience at the expense of increasing complexity of rekeying.

Since the security of a hybrid OFT is dependent on that of the underlying OFT, the hybrid model using the OFT [1,11] is subject to the collusion attack identified by Horng. However, adopting the solution proposed in [7] for a hybrid OFT, eliminates the security vulnerability. Fig. 3 illustrates the collusion attack and the remedy. The constrained optimization formulation (3) and (4) can be easily extended to the improved hybrid OFT, by modifying the left-hand side of (4), i.e., the update communication $C$. In the improved hybrid

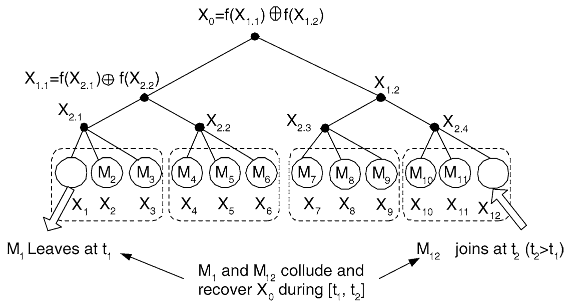

$X_0 = f(X_{1.1}) \oplus f(X_{1.2})$
$X_{1.1} = f(X_{2.1}) \oplus f(X_{2.2})$
$X_{1.2}$
$X_{2.1}$
$X_{2.2}$
$X_{2.3}$
$X_{2.4}$
$M_2$ $M_3$ $M_4$ $M_5$ $M_6$ $M_7$ $M_8$ $M_9$ $M_{10}$ $M_{11}$
$X_1$ $X_2$ $X_3$ $X_4$ $X_5$ $X_6$ $X_7$ $X_8$ $X_9$ $X_{10}$ $X_{11}$ $X_{12}$
$M_1$ Leaves at $t_1$
$M_{12}$ joins at $t_2$ ($t_2 > t_1$)
$M_1$ and $M_{12}$ collude and recover $X_0$ during $[t_1, t_2]$

Fig. 3. The collusion attack [6] on the hybrid OFT and the improved OFT [7] as a remedy. Consider the scenario that member $M_1$ leaves at time $t_1$ and $M_{12}$ joins at $t_2$ with $t_2 > t_1$. Let $X'_n$ and $X''_n$ denote the new secret of node $n$ at $t_1$ and $t_2$. Before $t_1$, $M_1$ holds node secrets $X_1$, $X_{2.1}$, and blinded secrets $f(X_{2.2})$ and $f(X_{1.2})$. When $M_1$ leaves at $t_1$, $X_{2.1}$ is updated, and so are $X_{1.1}$ and $X_0$ due to bottom up key derivation in an OFT. However, $f(X_{1.2})$ known to $M_1$ remains valid, i.e., $f(X'_{1.2}) = f(X_{1.2})$. Note that $X'_0 = f(X'_{1.2}) \oplus f(X'_{1.1}) = f(X_{1.2}) \oplus f(X'_{1.1})$. When $M_{12}$ joins at $t_2$, node secret $X_{2.4}$ is updated and so are $X_{1.2}$ and $X'_0$. The rest node secret remain the same from $t_1$ and hence $f(X''_{1.1}) = f(X'_{1.1})$. $M_{12}$ is assigned $X''_{12}$, $X''_{2.4}$, $f(X'_{2.3})$ and $f(X'_{1.1})$. If $M_1$ contributes $f(X_{1.2})$ and $M_{12}$ offers $f(X'_{1.1})$, they can collectively compute $X'_0$ used during $[t_1, t_2]$. A remedy [7] is to rekey all the blinded node secrets known to a deleted member after each eviction by changing leaf secrets. For example, upon deleting $M_1$, not only $X_{2.1}$ but also $X_{2.2}$ and $X_{2.3}$ are changed. The update of $X_{2.3}$ invalidates the $f(X_{1.2})$ held by $M_1$ and thus prevents the collusion attack.

model, the update communication of deleting a member is the cost to change its associated leaf secret, plus the cost to update all the $(a-1)\log_a \frac{N}{M}$ blinded secrets it holds. Therefore, the update communication overhead is

$$C = \left((a-1)\log_a \frac{N}{M} + 1\right) \times \left(M + (a-1)\log_a \frac{N}{M}\right) - 1.$$

Derivation of the analytical optimal cluster $M^*$ in the improved hybrid OFT remains an open problem.

### *5.3. Lower bound on update communication*

In [9], Micciancio and Panjwani established a tight lower bound on the update communication in multicast key distributions as $\log_2 N$. The fact that the bound matches one of the communication efficient schemes, OFC [3], up to a small additive constant, proves the optimality of this bound. Using the hybrid OFT, the lower bound of update communication (7) also matches the bound up to an additive constant term. Superseding previous results including $3\log_3 N$ in [12], this bound [9] provides guidance in search of optimal key distribution schemes.

## 6. Conclusions

In this paper, we formulated the design problem of a hybrid OFT key management scheme as a constrained optimization problem in order to minimize the center key storage under a key update communication budget. We derived the optimal cluster size $M^*$ by solving the equation of the form $M^* - \lambda \ln M^* = \mu$, with $\lambda$ and $\mu$ are model parameters. We presented an explicit design algorithm using the optimal cluster size, when update communication is prespecified. Our design methodology can be generalized to other tree based key management schemes, such as LKH, OFC, and the improved OFT that are perfect collusion resilient, with proper adjustment of the expressions for center storage and update communication.

Compared to OFT, the hybrid OFT with the optimal cluster size achieves reduction in GC storage at the expense of more rekey messages per update and more computation at the center. The tradeoff between center storage, key update communication, and center/user computation calls for further research. It is also equally important to evaluate other practical performance metrics such as cost of batch operations, resynchronization, rekeying for undetected compromise in the future study. It remains an open question whether it is possible to develop a collusion resistant scheme with GC storage lower than $O(\frac{N}{\log N})$ while maintaining the update communication as $O(\log N)$. Such a scheme may need exploration of additional relationships between keys, while preserving collusion resistance.

## Acknowledgements

The authors sincerely thank referees for their comments that significantly improved the presentation of this paper.

## References

[1] D. Balenson, D.A. McGrew, A.T. Sherman, Key management for large dynamic groups: one-way function trees and amortized initialization, Internet Draft (work in preparation), Internet Engineering Task Force, draft-irtf-smug-groupkeymgmt-oft-00.txt, July 2000.

[2] R. Canetti, T. Malkin, K. Nissim, Efficient communication-storage tradeoffs for multicast encryption, in: Eurocrypt'99, 1999, pp. 456–470.

[3] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, Multicast security: a taxonomy and some efficient constructions, in: Proc. IEEE Infocom'99, 1999.

[4] C. Duma, N. Shahmehri, P. Lambrix, A hybrid key tree scheme for multicast to balance security and efficiency requirement, in: Proc. 12th IEEE Internat. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), 2003.

[5] M. Heyman, private communication.

[6] G. Horng, Cryptanalysis of a key management scheme for secure multicast communications, IEICE Trans. Commun. E85-B (5) (2002) 1050–1051.

[7] W.C. Ku, S.M. Chen, An improved key management scheme for large dynamic groups using one-way function trees, in: Proc. Internat. Conf. Parallel Processing Workshop (ICPPW'03), 2003.

[8] M.Y. Li, R. Poovendran, C. Berenstein, Optimization of key storage for secure multicast, in: Proc. Conf. on Information Science and Systems, 2001, pp. 771–774.

[9] D. Micciancio, S. Panjwani, Optimal communication complexity of generic multicast key distribution, in: Advances in Cryptology-Eurocrypt'04, 2004, pp. 153–170.

[10] H. Seba, A. Bouabdallah, H. Bettahar, N. Badache, D. Tandjaoui, A hybrid approach to group key management, in: Proc. Internat. Network Conf., 2002.

[11] A.T. Sherman, D.A. McGrew, Key establishment in large dynamic groups: using one-way function trees, IEEE Trans. on Software Engrg. 29 (5) (2003) 444–458.

[12] J. Snoeyink, S. Suri, G. Varghese, A lower bound for multicast key distribution, in: Proc. IEEE Infocom'01, 2001.

[13] D.M. Wallner, E.J. Harder, R.C. Agee, Key management for multicast: issues and architectures, RFC 2627, 1999.

[14] C.K. Wong, M. Gouda, S.S. Lam, Secure group communications using key graphs, IEEE/ACM Trans. on Networking 8 (1) (2000) 16–31.