Design of Secure Multicast Key Management Schemes With Communication Budget Constraint

Mingyan Li, R. Poovendran, *Member, IEEE*, and C. Berenstein

***Abstract*—We study the problem of distributing cryptographic keys to a secure multicast group with a single sender and multiple receivers. We show that the problem of designing key distribution model with specific communication overhead can be posed as a constraint optimization problem. Using the formulation, we show how to minimize the number of keys to be stored by the group controller. An explicit design algorithm with given key update communication budget is also presented.**

***Index Terms*—Group communications, key management, multicast, optimization.**

I. Introduction

FOR point-to-multiplepoint group communications, multicasting is more efficient than multiple unicasts because it allows simultaneous delivery of data to multiple users and hence reduces computational overhead of a sender and network bandwidth requirement. However, the mainstream adaption 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 approach to control access to multicast communication is to use cryptography with a common shared session encryption key (SEK, also called traffic encryption key) which is known only by valid members at any time instant. Whenever there is a change in group membership, the SEK has to be updated to protect past, present, and future communications. When a member is removed from the group, the current SEK is compromised and cannot be used to encrypt the future data. To update *only* valid members with the new SEK, there need to be an additional set of keys called key encryption keys (KEK) to encrypt and distribute the new SEK. Then the problem of access control to multicast communication reduces to the secure distribution of KEK's to ensure only valid members have access to cryptographic keys at any given instant. This is the key management problem. Key storage and key update communication are two important overheads in key management.

In [1], [2], a tree-based key management scheme is presented, in which key update communication and user key storage grow as $\mathcal{O}(\log N)$ with the group size N. However, the key storage requirement of the centralized group controller (GC) grows linearly with N. A variation of tree-based key management is proposed in [3] to reduce the GC key storage. We adopt the model in [3] for study. However, the approach we have taken is different as our formulation presents the key distribution design under communication constraints as an optimization problem.

In this letter, we address the problem of minimizing the GC key storage while preserving the logarithmic user key storage and key update communication. We first show that the model needs to be hybrid and present an analytical formulation. Then the problem of key storage minimization with given communication budget can be posed as a constraint optimization problem with a design parameter as a variable. We convert the constraint optimization problem to a fixed point equation and show that the optimal design parameter is the largest root of the fixed point equation. Based on the optimal solution, an explicit design algorithm is described which allows designers to synthesize the key management parameters. We also present numerical computation of design examples.

II. Key Management Protocols

A. Minimal Key Storage Scheme

One key management scheme is to assign a unique KEK K_i to member M_i, where i is the member index. Every member stores two keys, its K_i and the SEK. When a member leaves, the GC has to encrypt the new SEK individually with K_is of the remaining $(N-1)$ members. Hence, key update communication overhead is $\mathcal{O}(N)$. To minimize the GC storage, a pseudorandom function f_r is used with a random seed r as an index to generate the key K_i as $K_i = f_r(i)$. The GC in this model only needs to store two keys, the SEK and the seed r.

B. Logical Key Hierarchy

Wallner *et al.* [1] and Wong *et al.* [2] proposed a scalable key management scheme by constructing a logical tree of KEKs for a given group. Fig. 1 illustrates a rooted binary key distribution tree for a group of eight members. In the tree, each member is assigned to a unique leaf node, thus fixing the number of leaves to be the group size N. Since the number of leaves determines the height of a tree, the height of the tree is also fixed and so is the total number of tree nodes in this model. Every node of the logical tree is assigned a KEK. The set of keys assigned to the nodes along the path from a leaf node to the root are assigned to the member associated with that particular leaf node. For example, member M_1 in Fig. 1 is assigned KEK's $\{K_O, K_{1.1}, K_{2.1}, K_{3.1}\}$. The member storage is thus

Manuscript received June 28, 2001. The associate editor coordinating the review of this letter and approving it for publication was Dr. L. Chen. This work was supported by the National Science Foundation under NSF Faculty Career Development Award ANI 00-93187.

M. Li and R. Poovendran are with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: radha@ee.washington.edu).

C. Berenstein is with the Institute of System Research and Department of Mathematics, University of Maryland, College Park, MD 20742 USA.

Publisher Item Identifier S 1089-7798(02)02988-5.

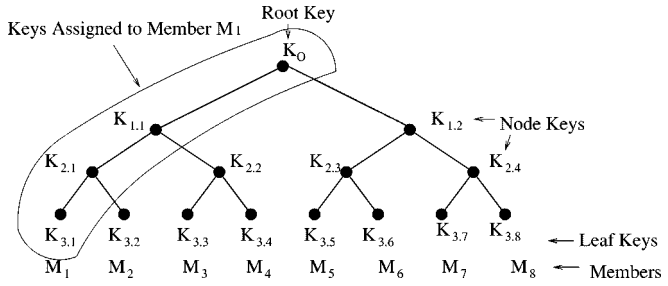

Fig. 1. A binary logical key tree for a group of eight members.

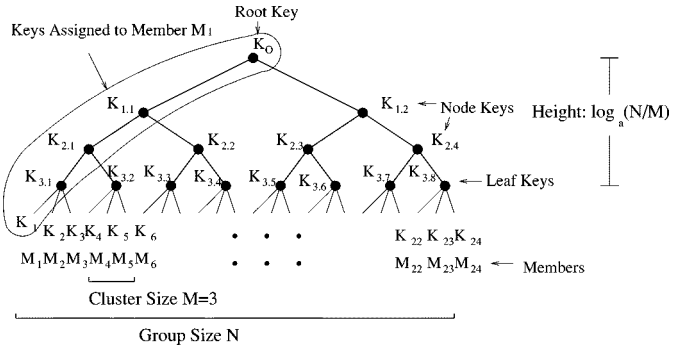

Fig. 2. A binary hybrid tree with cluster size $M = 3$ and group size $N = 24$.

the height of the tree, given as $(1 + \log_a N)$ for a tree of given degree a.

Since a member shares the root key and all the intermediate KEK's with other users, all the keys possessed by the member except the one at the leaf node have to be updated when the member is deleted. For example, when M_1 leaves, $\{K_O, K_{1.1}, K_{2.1}\}$ plus the SEK need to be updated. The number of key update messages [2] is given as $(a-1)\log_a N$. For this logical key hierarchy, the GC has to store *all* the keys corresponding to the nodes of the entire tree, which is $(aN-1)/(a-1)$ and scales as $\mathcal{O}(N)$. Hence, the key storage requirement of the GC is a bottleneck in this model.

The minimal storage scheme has constant GC storage but $\mathcal{O}(N)$ update communication cost, while the logical key hierarchy has $\mathcal{O}(\log N)$ key update communication but $\mathcal{O}(N)$ GC storage requirement. We need to have a hybrid model that can take advantage of both models.

C. Hybrid Tree Key Distribution

Since the number of leaves determines the total number of nodes in a tree of given degree, if we can set the number of leaves as a variable, then we can control the total number of keys. One approach [3] is to cluster the members and assign multiple members to a leaf, then by controlling the number of members assigned to a leaf node, we can vary the total number of nodes in the tree and thus the number of keys stored in the GC. We use the hybrid tree model in [3] to develop the design algorithm for a given amount of update communication.

The main idea of the hybrid tree is to divide the group into clusters of size M with every cluster assigned to a unique leaf node. Then there are N/M clusters (also leaves), and we need to build a tree of depth $\log_a(N/M)$. Fig. 2 illustrates this for a binary tree with cluster size $M = 3$ and a group of 24 members.

We notice that the structure in Fig. 2 consists of two parts, the logical tree, and the clusters. The logical key tree is used as inter-cluster key management scheme to limit key update communication, and the minimal storage used as the intra-cluster scheme to reduce GC storage requirement.

In the hybrid tree presented in Fig. 2, a user needs to store $(1+\log_a(N/M))$ KEK's required by the logical key tree 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 $(a-1)\log_a(N/M)$ 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 is computed as the keys on the tree plus seeds for (N/M) clusters, which is

$$S = \sum_{i=0}^{\log_a (N/M)} a^i + \frac{N}{M} = \left(1 + \frac{a}{a-1}\right)\frac{N}{M} - \left(\frac{1}{a-1}\right). \tag{2}$$

The last term $1/(a-1)$ is at most 1 since $a \geq 2$.

Since the logical key tree schemes have logarithmic update communication [1], [2], in the hybrid tree model, we want to keep the update communication as $\mathcal{O}(\log N)$ except some scale factor β. This can be expressed as:

$$M - 1 + (a-1)\log_a \frac{N}{M} \leq \beta \log_a N \tag{3}$$

where the communication scale factor β indicates how much communication can be alloted for key updates. The choice of parameter β should satisfy the inequality given later in (6).

III. Minimization of Key Storage With Communication Constraint

In the hybrid tree 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 $\mathcal{O}(\log N)$ while the key storage of the GC is better than $\mathcal{O}(N)$. Hence the optimization problem is posed as

$$\mathbf{min}\left[\left(1 + \frac{a}{a-1}\right)\frac{N}{M}\right] \quad \text{w.r.t.} \quad M \tag{4}$$

subject to the communication constraint given in (3). Note the storage in (4) is obtained from (2) by ignoring the last term, without affecting the solution of M.

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

Theorem 1: Optimal cluster size M that minimizes the storage function $S = ((2a-1)N)/((a-1)M)$ while satisfying the update communication budget $C = M - 1 + (a-1)\log_a(N/M) \leq \beta \log_a N$ is obtained by the largest root of the equation $M - \lambda \ln M = \mu$, where $\lambda = ((a-1)/\ln a)$ and $\mu = 1 + (\beta - a + 1)\log_a N$.

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 of this constraint op-

timization. Hence, the optimal value of the cluster size is computed by the equation:

$$M - \lambda \ln M + \lambda \ln N - 1 = \beta \log_a N. \tag{5}$$

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

$$\beta \geq \left(\frac{\lambda \left(1 + \ln \frac{N}{\lambda}\right) - 1}{\log_a N} \right). \tag{6}$$

With some algebra, it can be shown that for large values of N, the asymptotic lower bound of β approaches $(a - 1)$.

Equation (5) can be rewritten as

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

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

A. Computing Cluster Size M

The fixed point equation (7) is a contraction mapping with the largest root as the fixed point solution, if we start the iteration with an initial value $M_0 > \lambda$. We derive the solution using Newton's method [4]. By setting $M_0 = \mu$, the first-order approximation is $M_1 = \mu + \lambda \ln \mu$. Letting $N \to \infty$ yields

$$M_\infty = \mu + \lambda \ln \mu \approx \mu = 1 + (\beta - (a - 1)) \log_a N. \tag{8}$$

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

B. Computing Minimal Storage

We showed that the asymptotic solution to the optimization problem (4) is given in (8) for $N \to \infty$. The corresponding value of the GC storage denoted by S_∞ is

$$S_\infty = \frac{2a - 1}{(a - 1)(\beta - (a - 1))} \left(\frac{N}{\log_a N} \right). \tag{9}$$

Hence, the constraint optimization leads to the optimal growth of the GC storage as $\mathcal{O}(N/\log N)$ which is far better than $\mathcal{O}(N)$ growth, when the update communication is constrained to grow as $\mathcal{O}(\log N)$.

IV. A Design Example

For simulation, we set $\beta = (1 + \lambda) * \ln a$ which leads to $M_\infty = \ln N$ and $S_\infty = ((2a - 1)N/((a - 1) \ln N)$. As a specific design example, we are given the group size $N = 1000$ and the degree of the tree $a = 3$. The communication budget factor β is set to be 3. The cluster size M is computed as 8 by (8). A 3-ary tree with cluster size 8 requires 313 keys to be stored in the GC, while the update communication is less than $3 \log_3 N$.

TABLE I
COMPARISON OF GC STORAGE OF LOGICAL KEY HIERARCHY AND HYBRID TREE SCHEMES

(Degree a, Group size N)	# of keys in GC by logical tree	# of keys in GC by Eq (9)	Storage reduction in %
$(2, 2^{10})$	2047	444	78.3
$(2, 2^{20})$	2097151	226920	89.2
$(3, 2^{10})$	1535	370	75.9
$(3, 2^{32})$	6.44×10^9	4.84×10^8	92.5
$(4, 2^{10})$	1365	345	74.7
$(4, 2^{20})$	1398101	176490	87.4

Table I presents a numerical comparison between the logical key hierarchy and the hybrid tree scheme 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_∞ can lead to significant improvements in GC storage over values obtained in [1], [2] for a given communication budget.

V. Algorithm Summary

An explicit design procedure is given as follows.

1) Initial design data: group size N, tree degree a, and communication scale factor β
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 (8)
4) Construct a hybrid tree of degree a and cluster size M

Acknowledgment

The authors thank P. Dinsmore, R. Sampigethaya, A. Perrig, and E. Harder for their useful comments.

References

[1] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key management for multicast: Issues and architectures,", RFC 2627, June 1999.
[2] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Networking*, vol. 8, pp. 16–31, Feb. 2000.
[3] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," in *Eurocrypt'99*, pp. 456–470.
[4] M. Y. Li, R. Poovendran, and C. Berenstein, "Optimization of key storage for secure multicast," presented at the Conf. on Information Science and Systems, Baltimore, MD, Mar. 2001.