On the communication-storage minimization for a class of secure multicast protocols

R. Poovendran
Institute for Systems Research
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742, USA

Abstract

Developing cryptographic key management protocols that have scalability in terms of the key storage as well as key update communication is an important problem in many secure multicast applications [1], [9], [10]. Wong *et al.* [10] and Wallner *et al.* [9] independently presented the first set of key distribution models where the key update communication grows as $\mathcal{O}(\log N)$ for group of size N. However, the storage requirement of these models were $\mathcal{O}(\mathcal{N})$. Recently [2], a new model based on clustering of the group members was proposed in order to lower the key storage while maintaining the update communication growth as $\mathcal{O}(\log N)$. For the new model, by considering the product of the storage and the communication as the cost function, the optimal cluster size M was conjectured to be $M = \mathcal{O}(\log N)$. In this paper, we show that the optimal value of the cluster can be computed without the product function due the monotonicity of the storage with respect to the cluster size. We show that the optimal cluster size selection of the model in [2] can be formulated as a constraint optimization problem, and then transform it to a fixed point equation of the form $M - \lambda \log_e M = (\beta_2 - \lambda)\log_e N$, where β_2, λ are model parameters. We first show that the largest root of this equation is the optimal solution, and then compute it by two different techniques. We then show that the first order approximation of the solution is of the form $M \approx (\beta_2 - \lambda)\log_e N + \lambda \log_e \log_e N$, leading to $M \approx (\beta_2 - \lambda)\log_e N$ for large values of N. We make a case for use of the estimate $M = (\beta_2 - \lambda)\log_e N + \lambda \log_e \log_e N$ instead of $M = \log_e N$ by showing that even for group size up to 2^{32}, the value $M = \log_e N + \lambda \log_e \log_e N$ provides significantly lower value of key storage compared to the value $M = \log_e N$. We also show that the best estimate of M using the product function in [2] does not exceed $M = \nu \log_e N$ for a constant ν.

Keywords

Security, Optimization, Multicast Communications

I. Introduction

Use of multicast communications reduce the sender and the network overhead in applications where a set of identical messages have to be sent to multiple receivers. A single message can be sent to the entire group using the multicast communications mode. In the case of secure multicast, all the multicast group members have to share a common session encryption key called the group key [1]-[10]. In order to protect the past, present and future communications, the session key needs to be updated whenever a member leaves or joins the group.

Since the session key is known to every valid member of the group, additional keys called Key Encrypting Keys(KEKs) need to be distributed [9], [10], [2], [4] among members for key updates under member dynamics. When the session key has to be updated, KEKs that are shared among the valid members can be used for encryption and transportation. The KEK distribution should be (a) scalable in key update communication and/or key storage requirements of sender and receivers with respect to group size, (b) resistant to illegal collaboration (*collusion* free) of present or past members and (c) able to guarantee that deletion of one or more members does not invalidate the keys of valid members.

Wong *et al.* [10] and Wallner *et al.* [9] proposed the first tree based key update protocols with user storage and the key update communications that scaled as $\mathcal{O}(\log N)$, where N is the group size. The sender storage requirements of these two models were $\mathcal{O}(N)$. The sender storage constraints were bottlenecks of this model. In [2] a hybrid tree based scheme that reduced the sender storage from $\mathcal{O}(N)$ to (*sub-linear*) $\mathcal{O}(\frac{N}{\log N})$ while preserving the communication updates as $\mathcal{O}(\log N)$ was proposed. This was achieved by forming clusters of size M and then building a virtual tree of [9], [10]. By considering the product of the communication and the storage as the cost function, it was shown that for $M = \log N$, the storage requirements grow as sub-linear in N. The value of $M = \log N$, and the sub-linear storage were conjectured to be optimal for the hybrid scheme.

In this paper we formulate the sender storage minimization as a constraint optimization problem and show that there are only two solutions to the constraint optimization problem. We then show that the solution is obtained as the largest root of a fixed point equation. In computing the largest root, we show that for small values of N, our estimates will yield smaller storage values. We present a series approximation to the solution and show that under first order approximation, asymptotic (large N) solution of our formulation shows that the sub-linear solution is indeed optimal.

The paper is organized as follows: Section II presents the original tree based key distribution schemes [9], [10]. Section III presents the model of Canetti *et al.* [2] and identifies the problem posed. Section IV shows our formulation of the problem and the derivation of the optimal cluster size M, and the required proof of optimality of the cluster size. Section V presents numerical comparisons of the estimates of M presented in this paper and that in [2]. Section VI discusses the feasible maximum estimate of M for the product cost function used in [2].

II. Virtual Tree Based Key Distribution Protocols

The first use of rooted tree based key distribution approach for secure multicast communication was independently proposed in [9] and [10]. A rooted binary tree was used in [9] and key graphs were used in [10]. Both these approaches construct a logical tree or key graph based on the size of the group without making any assumption regarding the relationship among the keys. The group controller (GC) key storage requirements of these two schemes grow as $\mathcal{O}(N)$ while the user key storage and the update communication requirements grow as $\mathcal{O}(\log N)$.

A. Distribution of Keys on the Virtual Tree

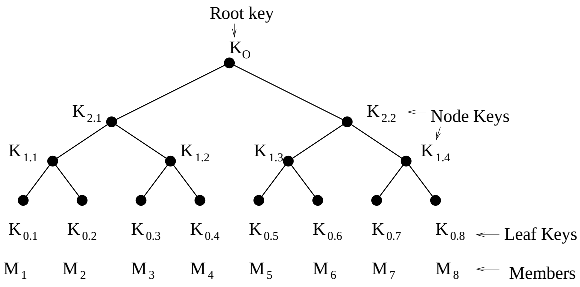

Fig. 1. The Logical or Virtual Key Tree of [9], [10].

The Figure 1 presents a rooted binary key distribution tree for a group with eight members. The logical tree is constructed such that each group member is assigned to a unique leaf node of the tree. Every node of the logical tree is assigned a key. 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 Figure 1 is assigned key encrypting keys $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$. Since the root key K_O is also shared by all the members, if there is no change in group membership, K_O can be used to update the session key (SK) for all the members.

The tree based structure also induces a natural hierarchical grouping among the members. By assigning the members to appropriate nodes, the group controller can form desired hierarchical clusters of members and selectively update, if needed, the keys of the group. For example, in Figure 1, members M_5, M_6, M_7, and M_8 exclusively share the key $K_{2.2}$. The GC can use the key $K_{2.2}$ to selectively communicate with members M_5, M_6, M_7, and M_8. Such clustering of the members on the tree may be decided by the GC based on application specific needs. In order to be able to selectively disseminate information to a subset of group members, the GC has to ensure that the common key assigned to a subset is not assigned to any member not belonging to that subset.

Using the notation $\{m\}_K$ to denote the encryption of message m with key K, and the notation A $\longrightarrow$ B : $\{m\}_K$ to denote the secure exchange of message m from A to B, GC can selectively send a message m to members $M_5, \cdots M_8$ by the following transmission:

GC $\longrightarrow M_5, M_6, M_7, M_8 : \{m\}_{K_{2.2}}$

If, however the key $K_{2.2}$ is invalidated for any reason, GC needs to update the key $K_{2.2}$ before being able to use a common key for members M_5, M_6, M_7, and M_8. It can do so by first generating a new version of $K_{2.2}$, denoted $\hat{K}_{2.2}$, and then performing two encryptions, one with $K_{1.3}$ and the other with $K_{1.4}$. The following two messages are needed to update key $\hat{K}_{2.2}$ to the relevant members of the group.

GC $\longrightarrow M_5, M_6 : \{\hat{K}_{2.2}\}_{K_{1.3}}$

GC $\longrightarrow M_7, M_8 : \{\hat{K}_{2.2}\}_{K_{1.4}}$

B. Member Deletion on Trees

Since the session key and the root key encrypting key K_O are common to all the members in the group, they have to be invalidated each time a member is deleted. Apart from these two keys, all the intermediate key encrypting keys assigned to the deleted member need to be invalidated. In the event there is bulk member deletion, the GC has to (a) identify *all* the invalid keys, (b) find the minimal number of valid keys that need to be used to transmit the updated keys, and (c) update the valid members with the new keys.

The general principle behind the member deletion is discussed below using member M_1 as example. Member M_1 in Figure 1 is indexed by the set of four keys $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$. Deleting member M_1 leads to invalidating these four keys and the session key, generating new keys, and updating these keys of the appropriate valid members who shared the invalidated keys with member M_1. When M_1 is deleted, the following updates are necessary: (a) all member need new root key K_O and new session key SK, (b) members $M_2 - M_4$ need to update $\{K_{2.1}\}$, (c) members $M_3 - M_4$ need to update $\{K_{1.2}\}$, and (d) member M_2 needs to update $\{K_{1.1}\}$.

The re-keying problem is thus reduced in [1], [9], [10] to the problem of finding efficient key encrypting key distributions.

Hence, the user storage as well as the update communications scale as $\mathcal{O}(\log N)$ for the tree based schemes in [9], [10]. However, these two approaches do not provide a mechanism for reducing the storage requirements of the group controller. The group controller has to store *all* the keys corresponding to the nodes of the entire tree. It can be shown that for any rooted tree, the storage requirement of the group controller is $\mathcal{O}(N)$[1]. Hence, the group controller key storage is a bottleneck in this model. For many applications with limited storage, it is desirable to reduce the group controller storage requirements as well. We now analyze the model presented in [2] and formulate the optimal cluster size estimation problem.

III. Model for Key Storage Reduction

In order to reduce the storage requirements of the group controller, Canetti *et al.* [2] purposed to assign a set of M members to each leaf node of the rooted trees of [9], [10]. Hence, each leaf node of the rooted tree has a cluster of M members. Within each of these clusters, following key scheme, called the minimal storage scheme, was proposed.

A. Minimal Storage Scheme

In the minimal storage scheme, every member needs to store only two keys. Every user holds the *common cluster* key K_s that is shared by all the cluster members, and a unique key K_u that it shares only with the group controller. The controller uses a random seed r as an index into pseudo-random function [2] f_r to generate the key K_u for member u as $K_u = f_r(u)$. Under this model, when a member leaves, the center generates the new common group key, encrypts it with the individual shared keys of the valid members and transmits. When a member is deleted within the cluster, the group controller has to perform $(M-1)$ individual encryptions to update the *common cluster key*.

[1]Exact value of the storage for a $d-ary$ tree is $\frac{dN-1}{d-1}$ and scales as N.

B. A Hybrid Scheme

The construction of a hybrid tree consists of (a) partitioning of group into clusters of size M and (b) constructing a virtual rooted tree of degree a with N/M leafs. Each cluster is assigned to a unique leaf of the $a-ary$ tree. Hence, this construct builds an $a-ary$ tree of depth $\log_a(N/M)$ followed by a $M-ary$ tree of depth one.

The crucial observation is that by performing clustering, the height of the rooted tree has been expressed as a function of the cluster size M. Hence, choosing the cluster size appropriately one can minimize the number of keys to be stored by the group controller. Selection of the cluster size 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 group controller scales better than $\mathcal{O}(N)$. Systematic construction of the hybrid trees is presented in [2] and is repeated in the following steps:

- The users are partitioned into subsets of size M, denoted as $U_i; i=1,\cdots\frac{N}{M}$.
- An $a-ary$ tree of depth $\log_a\frac{N}{M}$ is constructed.
- Each subset U_i of size m is assigned to a unique leaf in the $a-ary$ tree.
- Each user subset U_i of size M uses a minimal storage scheme for key assignment.
- Additional keys are assigned to each node of the $a-ary$ tree of depth $\log_a$

Total number of keys stored by the group controller in the hybrid scheme consists of the keys corresponding to the nodes of the tree as well as the seeds for $\frac{N}{M}$ clusters. Counting the nodes of the $a-ary$ tree of depth $\log_a\frac{N}{M}$ (which includes the $\frac{N}{M}$ seeds), the storage is given by

$$S \approx \frac{aN}{(a-1)M}. \tag{1}$$

When a member from cluster U_i is revoked, the group center has to update the $\log_a\frac{N}{M}$ keys on the $a-ary$ tree and also the $(M-1)$ cluster members with the common cluster key. The total number of update communications is $C=(M-1)+(a-1)\log_a\frac{N}{M}$.

C. Optimality Conjecture

The product of S and C was considered as the cost function in [2]. The resulting cost function denoted by F, is given by

$$\begin{aligned} F &= \left(\frac{aN}{(a-1)M}\right)\left(M-1+(a-1)\log\frac{N}{M}\right) \\ & \frac{aN(M-1)}{M}+\frac{aN\log\frac{N}{M}}{M} \\ &\approx aN+\frac{aN\log\frac{N}{M}}{M} \end{aligned} \tag{2}$$

It was noted in [2] hat if $M=\log_a N$, then

$$\begin{aligned} C &= \log_a N+(a-1)\log_a\frac{N}{\log_a N} \\ S &= \frac{aN}{(a-1)\log_a N} \end{aligned} \tag{3}$$

Hence, the update communication grows as $\mathcal{O}(\log N)$ while the key storage of the center grows as $\mathcal{O}(\frac{N}{\log N})$. For the hybrid model that has the update communication growth as $\mathcal{O}(\log N)$, the choice of cluster size $M=\log_a N$ and the corresponding storage $S=\frac{aN}{(a-1)\log_a N}$ were *conjectured to be optimal.*
We now show how to reformulate the problem as a constraint optimization problem that results in an explicit equation in variable M. We analyze this equation to find the optimal solution for cluster size.

IV. Reformulation of the Problem

Our approach is based on the observation that the optimal clustering problem can be restated as *minimize* S such that $C = \mathcal{O}(\log N)$.

Based on this observation, we claim that the following theorem characterizes the optimal cluster size selection problem.

Theorem 1: Optimal cluster size M that Minimizes the storage function $S = \frac{aN}{(a-1)M}$ while satisfying $M - 1 + (a-1)\log_a \frac{N}{M} = \mathcal{O}(\log N)$ is obtained by the largest root of the equation $M - \lambda \log_e M = \mu$, where $\lambda = \frac{(a-1)}{\log_e a}$ and $\mu > \lambda(1 - \log_e \lambda)$.

We first study the behavior of the functions C and S and then show how to convert the constraint into an explicit equation in M.

Under the hybrid scheme, the storage is $S = \frac{aN}{(a-1)M}$. The storage is a monotonically decreasing convex function of the cluster size M. Hence, in the absence of any additional constraint, choosing $M = N$ will yield the minimum storage of $S_{min} = \frac{a}{(a-1)}$. Figure 2 represents the storage as a function of M for $N = 1000$ and different values of a.

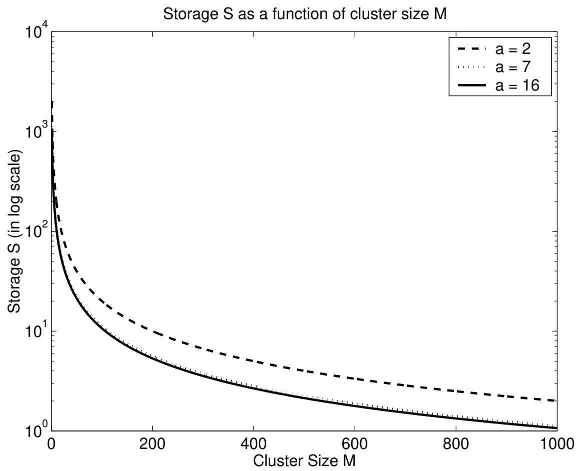

Fig. 2. The graph of storage vs cluster size M

The update communication for the hybrid model is $C = (M-1) + (a-1)\log_a \frac{N}{M}$. Setting $\lambda = \frac{(a-1)}{\log_e a}$, the update communication can be rewritten as

$$C = M - \lambda \log_e M + (\lambda \log_e N - 1). \tag{4}$$

Noting that $\lambda > 0$ and taking the derivatives of C with respect to M leads to

$$\frac{dC}{dM} = 1 - \frac{\lambda}{M} \tag{5}$$

$$\frac{d^2C}{d^2M} = \frac{\lambda}{M^2}. \tag{6}$$

The second derivative is positive independent of the values of M, whereas the first derivative changes sign at $M = \lambda$ and there is only one root at the stationary point. Hence the function is convex and has a unique minimum at $M = \lambda$. The corresponding minimum value of the update communication for the hybrid model is

$$C_{min} = \lambda(1 + \log_e \frac{N}{\lambda}) - 1. \tag{7}$$

Since $\lambda = \frac{(a-1)}{\log_e a}$ is a small constant for a given $a-ary$ tree, the minimum value of the update communication grows as $\mathcal{O}(\log N)$. However, the cluster size $M = \lambda$ leads to the key storage $S = \frac{aN}{(a-1)\lambda}$, which grows as $\mathcal{O}(N)$. Hence, when the cluster size $M = \lambda$, the function C grows as $\log N$ with the smallest possible overheads while the storage grows as N. Figure 3 represents the update communication as a function of cluster size M for $N = 1000$ and different values of a.

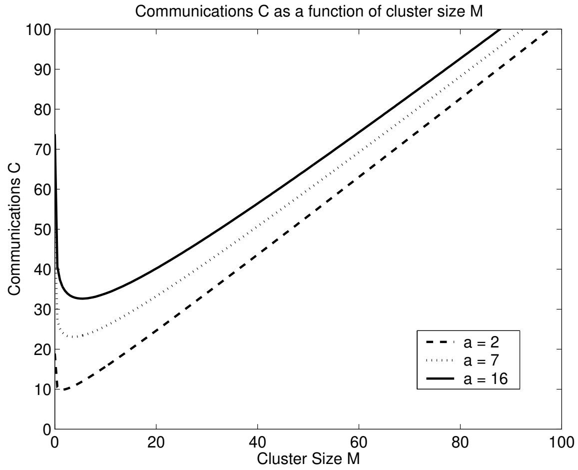

Fig. 3. The graph of communication vs. cluster size M.

Table 1 presents the values of storage and communication functions when $M = N$ and $M = \lambda$.

M	Storage	Communication
N	$\frac{a}{(a-1)}$	$N-1$
λ	$\frac{a\lambda}{(a-1)\lambda}$	$\lambda(1+\log_e \frac{N}{\lambda})-1$

Since λ is a constant, we note that when the communication is minimized the storage grows as a linear function of N and when the storage is minimized communication grows as a linear function of N. Figure 4 presents the graph of communication vs storage as a function of cluster size M, illustrating the tradeoff between C and S. We now present the main rationale behind our solution to the problem.

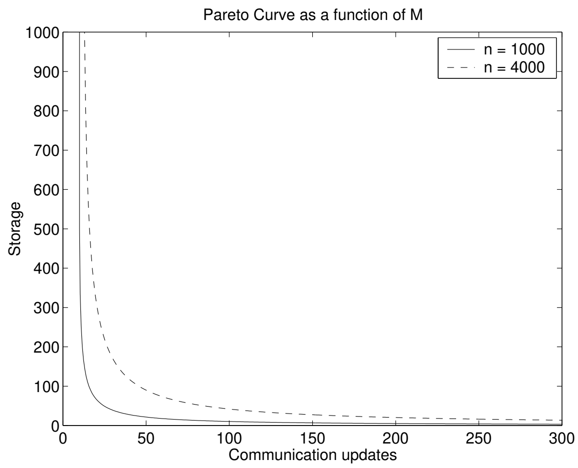

Fig. 4. The graph of Communication vs Storage Tradeoff for binary tree

A. Solution Approach

The Virtual trees of Wong *et al.* [10] and Wallner *et al.* [9] have the update communication growth of $\mathcal{O}(\log N)$. Hence, any improvement to the tree scheme needs to maintain update communication growth at most as $\mathcal{O}(\log N)$ while trying to minimize the key storage of the group controller from $\mathcal{O}(N)$. This observation can be stated as a constraint that *minimize* S with respect to M subject to the constraint $C = \mathcal{O}(\log N)$. In general, this constraint optimization can be made unconstraint optimization by constructing a cost function using C and S. We note that it is not necessary for this model since the storage is a monotonic function of M.

Due to the monotonicity of the storage S with respect to the variable M, it is sufficient to find the values of M that satisfy the update communication constraint. Since the storage is a monotonically decreasing function, the largest value of M satisfying the update communication constraint will be the solution of this constraint optimization.

In order to minimize the storage, we note that the logarithmic growth condition on update communication implies that there is a pair of positive real numbers β_1, β_2 such that the update communication is lower and upper bounded by $\beta_1 \log_e N$ and $\beta_2 \log_e N$ for all permitted values of M. This can be mathematically expressed as:

$$\beta_1 \log_e N \quad \leq \quad C \leq \beta_2 \log_e N, \tag{8}$$

Since $C_{min} = \lambda(1 + \log_e \frac{N}{\lambda}) - 1$, choosing $\beta_1 = \frac{C_{min}}{\log_e N}$ will ensure that $\beta_1 \log_e N \leq C$.

If $\beta_2 > \frac{C_{min}}{\log_e N}$, then the graph of the line $\beta_2 \log_e N$ will intersect the convex function C at two different values. Although all the values of M that lie between these two values of M satisfy the update communication constraint, the largest value of M is the one that will yield the lowest value of the storage S. However, the largest value of M satisfying the update communication constraint is also the largest value of M corresponding to the intersection of the graph of C and the line $\beta_2 \log_e N$.

Hence, the optimal value of M is obtained by the equation solving for the largest value of M at the intersection of the graph of C and line $\beta_2 \log_e N$. Clearly, values of M at the intersecting points are given by $C = \beta_2 \log_e N$ leading to the equation

$$M - \lambda \log_e M + (\lambda \log_e N - 1) = \beta_2 \log_e N \tag{9}$$

This can be rewritten as

$$\begin{aligned} M - \lambda \log_e M &= 1 + (\beta_2 - \lambda) \log_e N \qquad (10) \\ \Rightarrow M - \lambda \log_e M &= \mu \end{aligned}$$

where $\mu = 1 + (\beta_2 - \lambda) \log_e N$.

B. Computing Cluster Size M

Since the function $M - \lambda \log_e M + (\lambda \log_e N - 1)$ is convex with the minimum value $\lambda(1 + \log_e \frac{N}{\lambda}) - 1$, if $\beta_2 > \frac{\lambda(1+\log_e \frac{N}{\lambda})-1}{\log_e N}$, then the equation (9) has two solutions. Since $M - \lambda \log_e M$ has the minimum at $M = \lambda$ and $\lambda > 0$, the roots of the equation $M - \lambda \log_e M = \mu$ lie below and above $M = \lambda$. Moreover, if $M > \lambda$, the gradient of $M - \lambda \log_e M$ is $1 - \frac{\lambda}{M} > 0$. Hence the fixed point equation

$$M - \lambda \log_e M = \mu \tag{11}$$

is a contraction mapping with the largest root as the fixed point solution if we start the iteration with an initial value $M_0 > \lambda$. Since $\lambda, \mu > 0$, $M = \mu + \lambda \log_e M > \mu$, if we set the initial value of M to be $M_0 = \mu$, after some algebra, a series approximation to M is given by

$$M = \mu \prod_{i=1}^{\infty} (1 + \left(\frac{\lambda}{\mu}\right)^i \log_e \mu), \tag{12}$$

Since $\lambda > 0$ is fixed, and $\log_e \mu < \mu$ as $\mu \to \infty$, if we denote the asymptotic value of M by M_∞, the limiting value is given by

$$\begin{aligned} M_\infty &= \lim_{\mu \to \infty} \mu \prod_{i=1}^{\infty} (1 + \left(\frac{\lambda}{\mu}\right)^i \log_e \mu) \qquad (13) \\ &= \mu + \lambda \log_e \mu \\ &\approx \mu \qquad (14) \end{aligned}$$

But $M_0 = \mu \approx (\beta_2 - \lambda)\log_e N$. Hence, the asymptotic value of the largest root of the equation $M - \lambda \log_e M = \mu$ is $M_\infty \approx (\beta_2 - \lambda)\log_e N$.
We now show that the same results can be derived using the first order Taylor series approximation using Newton's method. Setting the first approximate solution to the equation $M - \lambda \log_e M = \mu$ for $M > \lambda$ as $M_0 = \mu$, the first approximation is $M_1 = \mu + \lambda \log_e \mu$. This is indeed the asymptotic solution we obtained using the fixed point iterations earlier. Letting $N \to \infty$ leads to $M_\infty \to \mu + \log_e \mu \approx \mu$. It can be shown that even if the series is computed for higher order terms, for large values of N, the largest root M of the equation $M - \lambda \log_e M = \mu$ converges to $M_\infty = \mu + \lambda \log_e \mu$ and grows as $\mathcal{O}(\log N)$.

C. Computing Minimal Storage

We showed that the asymptotic value of the largest root of the equation $M - \lambda \log_e M = (\beta_2 - \lambda)\log_e N$ is $M = \mu = (\beta_2 - \lambda)\log_e N$ by two different approaches. The corresponding value of the storage denoted S_∞ is

$$\begin{aligned} \lim_{N\to\infty} S_N &= \frac{aN}{(a-1)M_\infty} \qquad (15) \\ S_\infty &= \frac{aN}{(a-1)\{\mu + \lambda \log_e \mu\}} \\ &\approx \frac{aN}{(a-1)(\beta_2 - \lambda)\log_e N}. \end{aligned}$$

Hence, the constraint optimization leads to optimal growth of storage as $\mathcal{O}(\frac{N}{\log N})$ when the update communication is constrained to grow as $\mathcal{O}(\log N)$. We now formally state it as the proof of *Theorem 1*.
Proof of Theorem 1: The storage $S = \frac{aN}{(a-1)M}$ is a monotonically decreasing function of M. We showed that the minimal storage is obtained by the *largest cluster size* M that satisfies the logarithmic growth constraint of the update communications. We showed that to find the minimal value of the storage, the constraint on the update communication $C = M - 1 + (a-1)\log_a \frac{N}{M} \le \beta_2 \log_a N$ for $\beta_2 > \frac{\lambda(1+\log_a \frac{N}{\lambda})-1}{\log_e N}$ can be converted as an equation of the form $M - \lambda \log_e M = \mu$ involving the cluster size M. We then showed that this equation has two roots and the asymptotic form of the largest root is $M_\infty = (\beta_2 - \lambda)\log_e N$. Hence the optimal storage under communication constraint is $S = \frac{aN}{(a-1)(\beta_2-\lambda)\log_e N}$.
We note that the proof of the theorem 1 also completes the proof of the conjecture in [2] which stated that for the hybrid tree based scheme with update communication growth of $\mathcal{O}(\log N)$, the minimum value attained by the storage is $\mathcal{O}(\frac{N}{\log N})$.

V. Numerical Comparisons

We formulated the storage minimization with communication constraint as an optimization problem. We did this without choosing a pre specified cost function. We showed that the optimal cluster size M is given by $M_\infty = \mu + \lambda \log_e \mu$ which may be approximated to $M = \mu$ when the value of μ is significantly larger than that of $\lambda \log_e \mu$. We also showed that this approximation leads to results in [2]. We now show by numerical illustration that the cluster size computed as $M = \log_e N$ is a significant underestimation compared to $M_\infty = \log_e N + \lambda \log_e \log_e N$ for group sizes ranging from few hundreds to several millions.
We recall that $\mu = (\beta_2 - \lambda)\log_e N$ where the parameter β_2 needs to satisfy $\beta_2 > \frac{\lambda(1+\log_a \frac{N}{\lambda})-1}{\log_e N}$. Although β_2 can be chosen as large as possible, for simplicity, we set $(\beta_2 - \lambda) = 1$ which leads to $\mu = \log_e N$. The optimal cluster size in this case is $M = \mu + \lambda \log_e \mu = \log_e N + \lambda \log_e \log_e N$.
If we denote $M^* = \log_e N$, the fraction of improvement in estimate of the cluster size is

$$\frac{M_\infty - M^*}{M^*} = \frac{\lambda \log_e \log_e N}{\log_e N}. \qquad (16)$$

For $a = 2$, $N = 2^{20}$, we compute $\lambda = 1.442$, $M^* = \log_e N = 14$, and $M_\infty = (\log_e N + \lambda \log_e \log_e N) = 18$. Since [2] uses M^* for estimates, results in [2] under estimate the cluster size by 27%. Using M^* leads to the storage of $S = \frac{aN}{(a-1)\log_e N} = 149796$ keys whereas the use of M_∞ leads to the storage of $S = \frac{aN}{(a-1)(\log_e N + \lambda \log_e \log_e N)} = 116508$ keys. Using M^* instead of M_∞ leads to an additional 20% storage requirement.

Similarly, for $a = 4$, $N = 2^{14}$, we compute $\lambda = 2.164$, $M^* = 10$, and $M_\infty = 15$. The improvement in estimate of cluster size is 50%. Using M^* leads to storage of 2184 keys and the use of M_∞ leads to storage of 1456 keys. Using M^* leads to an additional 33% storage requirement. Table 2 presents the improvement in estimates for several pairs of (a, N). From the table 2, we also note that the use of the estimate $M_\infty = \log_e N + \lambda \log_e N$ yields significantly lower values of keys to be stored even for group size up to 2^{32}. Hence, for all practical purposes, the first order estimate value $M_\infty = \log_e N + \lambda \log_e N$ will yield minimal storage.

(a, N)	$\frac{\lambda \log_e \log_e N}{\log_e N}$	Improvement in %
$(2, 2^{10})$	0.40	40%
$(2, 2^{15})$	0.32	32%
$(2, 2^{20})$	0.27	27%
$(2, 2^{32})$	0.20	20%
$(3, 2^{10})$	0.50	50%
$(3, 2^{15})$	0.40	40%
$(3, 2^{20})$	0.34	34%
$(3, 2^{32})$	0.25	25%
$(4, 2^{10})$	0.59	59%
$(4, 2^{15})$	0.47	47%
$(4, 2^{20})$	0.40	40%
$(4, 2^{32})$	0.29	29%

Figure 5 presents the function $\frac{\lambda \log_e \log_e N}{\log_e N}$ as a function of group size N for various values of a. The fraction of improvement monotonically decreases at the rate of $\frac{\lambda(1-\log_e \log_e N)}{N(\log_e N)^2}$. However, for the group size values from 100 to 3 million, the improvement is at least around 20%.

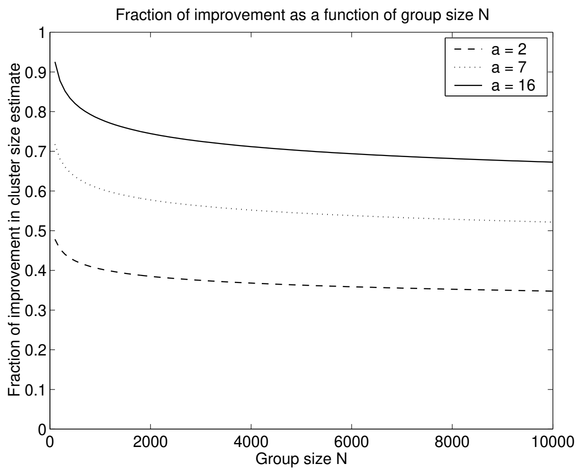

Fig. 5. The graph of the improvement $\frac{\lambda \log_e \log_e N}{\log_e N}$ as a function of the group size N.

We note that for the term $\lambda \log_e N$ to be negligible, the improvement in estimate given by $\frac{\lambda \log_e \log_e N}{\log_e N}$ should be negligible. If we denote the percentage improvement by ϵ, then we require $\frac{\lambda \log_e \log_e N}{\log_e N} \leq \epsilon$. However, if the value of ϵ is small, say 1%, then the value of N needs to be extremely large even for a binary tree. For example, letting $a = 2$, $\epsilon = 1\%$, leads to the condition that $\frac{\log_e \log_e N}{\log_e N} \leq 0.007$. It can be checked that even the value of $N = 2^{100}$ will yield the improvement of 8% for a binary tree. Hence, for all practical cases, the estimate given by $M^* = \log_e N$ significantly underestimates the optimal cluster size.

In conclusion, we note that the since $M_\infty = \log_e N + \lambda \log_e N$ is not only optimal but also yields a significantly lower number of keys to be stored by the center. This can be verified by noting that $M_\infty > M^* \Rightarrow \frac{aN}{(a-1)M_\infty} < \frac{aN}{(a-1)M^*}$.

A. Choice of the free Parameter β_2

In the virtual tree scheme [9], [10], the update communication cast was $\log N$ [3]. Note that the amount of additional update communications needed for the hybrid model is given by the parameter β_2 in equation (10). Ideally, the choice of β_2 should be as small as possible to keep the update communications as close to the virtual trees as possible. However, we note that $S_\infty = \frac{aN}{(a-1)(\beta_2-\lambda)\log_e N}$ indicates the choice of larger values of β_2 will lead to lower values of storage. We note that the parameter $\beta_2 > \frac{C_{min}}{\log_e N}$ is a design choice as in the case of the degree a of the key distribution tree [9], [10].

VI. Choosing an Appropriate Cost Function

When there is a tradeoff between the variables under consideration, often a cost function is constructed based on the product of the variables or suitable functions of these variables. We now show that the value of $M^* = \log_e N$ is indeed the best estimate for the product function considered in [2]. In order to show this we repeat the equation (2) $F = aN + \frac{aN \log_a \frac{N}{M}}{M}$. Since the aim in [2] was to ensure that the growth of F is $\mathcal{O}(N)$, we have the constraint $\frac{aN \log_a \frac{N}{M}}{M} = \gamma N$ for some non-negative real number γ. This can be written as

$$M = \nu \log_e \frac{N}{M} \tag{17}$$

for some positive constant ν. This equation can be further expanded as

$$M + \nu \log_e N = \nu \log_e N. \tag{18}$$

Since $N > M$, $\log_e N > \log_e M$ and ν being a positive constant, $\nu(\log_e N - \log_e M) = M > 0$ as expected. Moreover, since $M > 1$ for a meaningful solution, $M = \nu(\log_e N - \log_e M) \Rightarrow M < \nu \log_e N$. Noting that this equation is similar to the one we derived earlier with change of sign of the term $\nu \log_e M$, we can write the asymptotic solution of this equation as

$$M = \nu \log_e N - \nu \log_e \log_e N - \nu \log_e \nu. \tag{19}$$

From this expression, we note that the value of M is always over estimated by the amount of $\log_e(\nu \log_e N)$ if we set $M = \log_e N$. Hence, the estimate $M = \nu \log_e N$ is indeed the best possible estimate if we use the product of the storage and communication of the hybrid model as the cost function. As noted earlier, even this approximation falls short of the best possible cluster size for the hybrid model by the factor $\frac{\lambda \log_e \log_e N}{\log_e N}$.

VII. Conclusions

In this paper, we showed that optimal cluster size M that minimizes the key storage while preserving the update communication of a hybrid model for secure multicast communication can be formulated as a constraint optimization problem. We then showed that this constraint optimization problem can be converted to an equation of the form $M - \lambda \log_e N = \mu$ where λ and μ are model parameters. We also showed that the largest root of this equation is the desired optimal solution and computed it as $M_\infty = \log_e N + \lambda \log_e N$ for a first order approximation. We showed that as $N \to \infty$, $M_\infty = \log_e N$ and this value shows that the conjecture in [2] is true for asymptotic values of N. We also showed that the estimate of $M = \log_e N + \lambda \log_e \log_e N$ produces significantly less storage even for group sizes of the order of 2^{32}.

We note that developing models that will lead to scalability of update communications lower than $\mathcal{O}(\log N)$ while requiring storage lower than $\mathcal{O}(\frac{N}{\log N})$ is a useful open problem. Such a model may require additional relationship among the keys used. Development of such a scheme also needs to consider the collusion resistance as a feature.

VIII. Acknowledgments

I would like to acknowledge the discussions with Drs. Eric Harder and C. Berenstein.

References

[1] R. Canetti, and B. Pinkas, "A taxonomy of multicast security issues", *Internet draft*, April, 1999.

[2] R. Canetti, T. Malkin, and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption", Eurocrypt 99, pp. 456 - 470.

[3] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, "Multicast Security: A Taxonomy and Efficient Reconstructions", In *Proceedings of IEEE Infocom'99*, pp. 708-716.

[4] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, "Key Management for Secure Internet Multicast Using Boolean Function Minimization Techniques", *Proceedings of IEEE Infocom'99*, pp. 689-698.

[5] A. Fiat and M. Naor, "Broadcast Encryption", *Advances in Cryptology- CRYPTO'92*, LNCS. vol. 773, pp. 481-491, Springer-Verlag, Berlin Germany, 1993.

[6] R. Kumar, S. Rajagopalan, A. Sahai, "Coding Constructions for Blacklisting Problems without Computational Assumptions", *Advances in Cryptology- CRYPTO'99*, LNCS. vol 1666, pp. 609-623, Springer-Verlag, Berlin, Germany, 1999.

[7] S. Mittra, "Iolus: A framework for Scalable Secure Multicasting", In *Proceedings of ACM SIGGCOM'97*, pages 277–288, September 1997.

[8] R. Poovendran, "On the Communication-Storage Minimization for a Class of Secure Multicast Protocols", Technical Report, Department of Computer Science, University of Maryland, College Park, June, 2000.

[9] D. M. Wallner, E. C. Harder, and R. C. Agee, "Key Management for Multicast: Issues and Architectures", Internet Draft, September 1998.

[10] C. K. Wong, M. Gouda, S. S. Lam, "Secure Group Communications Using Key Graphs", IEEE/ACM Trans. on Networking, Vol.8, No.1, pp.16-31, Feb. 2000. (Also in *Proceedings of ACM SIGCOMM'98*, September 2-4, Vancouver, Canada.)