A Convex Optimization Approach for Addressing Storage-Communication Tradeoffs in Multicast Encryption

Radha Poovendran
CS TR # 4082
{radha}@isr.umd.edu
Dept. of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742, USA

Abstract

In Eurocrypt'99, Canetti, Malkin, and Nissim [1] presented a new tree based key distribution algorithm that required $\mathcal{O}(\log n)$ (n is the group size) key update communications and key storage of $\mathcal{O}(\frac{n}{\log n})$. The results in [1] are known to be the first results presenting the sub-linear storage among the family of tree based key distribution schemes. The question of whether this storage was the possible optimal value while keeping the communication as $\mathcal{O}(\log n)$ was posed as a problem.

We show that the storage-communication tradeoff can be formulated as a convex optimization problem in terms of the size of the minimal storage parameter defined in [1]. In particular, we show that the optimal solution is parameterizable by the ratio of the communication and storage costs denoted λ, the degree of the tree denoted a, and the group size n. Using this design triplet (a, n, λ), we show that not only the results in [1] but also the results of the basic scheme of Wallner, Harder, and Agee [2] can be derived as specific Pareto optimal points for specific choice of the triplet (a, n, λ). We also present an exact design procedure for feasibility testing and constructing optimal key distribution tree of the type in [1].

We also show that if the communication and the storage are equally weighted, then the optimal value for storage and communication is $\mathcal{O}(\sqrt{n})$, a value noted in [1].

Key Words: Multicast Security, Key Distribution, Trees, Convex Optimization

1 Introduction

In order to reduce the sender and the network resources, secure multicast encryption requires all the group members to share a single data encryption key called the group key [2, 1, 18, 12, 13, 17, 11]. In order to protect the access to group communication, the group key needs to be updated whenever a member leaves or joins the group. Most of the recent literature assumes that there is a single group controller that is responsible for key distribution [2, 1, 3]. We make the same assumption in this paper. Whenever there is membership change, the group controller needs to perform re-keying to protect the integrity of the group communication. The focus of recent efforts has been to provide *efficient* schemes for re-keying.

1.1 Re-keying Problem

The re-keying problem is to distribute the new group keys such that only the valid members of the group receive the updated group keys. Since the new group key needs to be encrypted using a key encrypting key and then transmitted, the problem of group key updating is related to the problem of distributing the key encrypting keys [2, 1, 3]. We now illustrate the re-keying problem by a set of well known examples that illustrate the extremes in [2, 1, 3].

If the group is such that no member leaves the group during session lifetime, but new members may join the group, re-keying is solved by two key updates with the storage requirement of two keys at the controller and at the user side. The group center assigns a key K as the group key, and K' as the updated group key. The group center encrypts the updated group key K' with the public key of the new member, and with the group key K. These two encryptions will allow the key update for the entire group. Here, the number of keys to be stored by the sender and the receivers is two. The number of encryption and the updates is also two.

If the group has members that also leave in the middle of the session, this approach will not provide communication integrity for the valid members. Since all the valid members have access to the group key, if a member leaves the group, the group center has to use a key that is not accessible to the leaving member. In the absence of any other shared key, the group center can use individual public keys of the valid members. For a group size of n, this requires (n-1) encryptions each time a member leaves the group. Instead of using public keys, the group center may also establish unique individual key encrypting key with each member. At the time of a single member leave, the group center has to encrypt the new group with the shared key of the individual members. This approach requires two keys to be stored at the user end, (n +1) keys to be stored at the group center, and requires (n-1) encryptions when a member leaves.

If the number of encryption at the group center and the use of network resources have to be reduced at each member leave, the group center can construct all possible subsets of users. There are 2^n such subsets. Once these sets have been formed, each set is assigned a unique key encrypting key. When a member leaves the group, the group center identifies the subset that doesn't contain the leaving member and uses the corresponding key encrypting key to update the group keys to the rest of the members. This scheme requires a single encryption. However,it requires at least 2^n keys to be stored by the group center and 2^{n-1} keys to be stored by every user. Finding intermediate solutions that provide efficient tradeoffs between the storage and communication[1](related to the number of update encryptions) has been the most actively researched area in the literature [2, 1, 3].

The re-keying problem is thus reduced in [2, 1, 3] to the problem of finding efficient key encrypting key distributions.

1.2 Our Results

Our approach is based on the observation that the amount of communication (encryption) and the key storage requirements can be expressed as functions of the parameters defining a key distribution scheme. For any key distribution scheme, the cost for the total storage and the update communication also are known and need to be kept as low as possible. We use λ_1 to denote the unit cost of storage, and λ_2 to denote the unit cost of communication. We then define a triplet denoted (a, n, λ), where a is the degree of the tree, n is the group size, and λ is the ratio of the communication and the key storage cost. We then construct a weighted cost function that contains the costs of key storage and the update communication messages. We then show that this cost function is a *convex function* in terms of the parameters used in [1].

[1]*Network Related Efficiency Parameters:* We note that the storage and the communication are not the only parameters for measuring efficiency. Since the communication is over a network that possibly has multiple services, network parameters such as (1) bandwidth, (2) amount of the input queuing, and (3) queue priorities also play a major role in time efficiency of a re-keying scheme.

Since a convex function has unique minima with respect to the parameter of interest, we show that the proposed sub-linear storage approach in [1] corresponds to a unique optimal solution with respect the variable that optimizes the cost function. We then *analytically* solve for the parameter of interest *explicitly* inter-Ms of the triplet (a, n, λ). *Hence, given the triplet, the design of the optimal tree is completely specified if there is no additional relationship between the parameters.* This formulation helps not only in designing an optimal tree, but also in deciding the explicit costs of storage and the update communication of the optimal tree! Hence, given a tree design, one can identify if it is an optimal one and further improve the design. We illustrate the generality of our approach by showing how to design the currently known *optimal schemes*. We show that each of these schemes correspond to a specific cost choice on the Pareto curve parameterized by the triplet (a, n, λ). Interestingly, it is the relative ratio of the costs that plays a role in the optimal design and not the specific numerical value.

Using our results, we show how to derive the results in [2] as a special case. We then show how to construct a variety of sub-linear solutions. We note that if the storage and communication are equally weighted, the optimal solution for communication and storage is of the order of $\mathcal{O}(\sqrt{n})$. We also note that our approach will allow for systematic parameter selection for the tradeoffs. We then generalize the results and derive a condition under which it is possible to develop a unique optimal solution.

The paper is organized as follows: Section two presents related work excluding [1]. Section three present the modified tree based scheme proposed by [1]. Section four presents our formulation of the problem and the derivation of the optimal solution. Section five then shows how to derive the basic tree based scheme in [2] and square root storage-communication schemes. Section six shows how to construct optimal trees with sub-linear storage [1] and also how to test if a given sub-linear storage tree is really optimal with respect to the Pareto triplet (a, n, λ).

2 Prior Work

In [6, 7], communication and storage tradeoffs were studied with information rate as a defining quantity. In particular, these schemes use unconditional or information theoretic security and characterize the lower and upper bounds of the key distribution requirements. In [14], bounds on the broadcast encryption were developed using combinatorial methods. In particular, authors showed that there is a tradeoff between the communication and the storage with no simultaneous minimization of both these quantities being feasible.

In [9], Fiat and Naor presented a variety of techniques that captures several possible variations of broadcast key distribution. They also presented a broadcast encryption model that can be collusion free up to a pre-specified number of group members. Under their model, joining and leaving of members didn't involve computations for the rest of the members. The threshold number of members who can break the scheme was however tied to the model.

In [15], an extension to the approach in [9] was given using the theory of error correcting codes as the basis. The scheme required $\mathcal{O}(\log n)$ communications with polynomials and $\mathcal{O}(k^2)$ with Algebraic geometric codes where k being the number of excluded users. Due to the use of error correcting codes, the method in [15] can also correct errors in transmission under noise.

In [2, 17] a binary tree based scheme that provides key distribution was proposed. In this scheme, for a given set of n users, a logical tree of depth $\log n$ was constructed. Each member of the group was assigned to a unique leaf of the tree. All the nodes including the leafs and the root were treated as place holders of intermediate keys. The key assignment then proceeded as follows: for a member assigned to a leaf, the set of keys assigned to all the nodes along the path from that leaf to the root were assigned to a member.

Each of the members was assigned a unique set of keys so that the removal of every individual member was feasible. At the time of member removal, the keys assigned to the removed member were invalidated. Since these key are placed on the nodes along the path from the leaf to the root, all the node keys that are along the path from the leaf to the root are invalidated. The valid members whose path from the leaf to the root shared common nodes had to be updated with the new keys. This process involved $\mathcal{O}(\log n)$ to be stored by each user and involved $\mathcal{O}(\log n)$ update communications. An efficient key update scheme based on pseudo-random functions was proposed in [12] to improve the message update efficiency of the scheme in [2] by a factor of two under member deletion.

In [13], one-way function trees were used for constructing dynamic groups. In [16], an approach based on member deletion probabilities was proposed. Authors proposed to assign probabilities to each member deletion, and showed that on average, the optimal number of keys to be assigned to a member is equal to the entropy of the member deletion event. Results relating coding to the key assignment was also noted in [16]. They did not however provide any relationship between the group center storage and the key updated communication.

3 Review of the Model for Sub-Linear Solution

In what follows, since the first optimal sub-linear solution was pointed out by Canetti, Malkin and Nissim in [1], we will call such trees CMN trees. We will review them in section three. However, we note that the CMN trees have two positive parameters we call them the CMN pair, denoted as (γ, β). The value of γ is the scale constant in $\frac{\gamma n}{\log n}$ appearing in the storage term, and the value of β is the scale constant appearing in the communication as $\beta \log n$. We now describe the CMN tree construction. The CMN tree [1] is a hybrid scheme that combines minimal storage model and the basic tree scheme [2] described above.

3.1 Minimal Storage Scheme

In the minimal storage scheme of [1], each user holds the common group key K_s that is shared by all the members, and a unique key K_u that it shares only with the group center. The center uses a random seed r as an index into pseudo-random function [8] 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. Under member deletion, this approach requires $(n-1)$ encryptions, but requires each user to store only two keys. *The minimal storage scheme can be thought of as a m-ary tree with depth one.*

3.2 The CMN Trees

The CMN trees are constructed using a hybrid approach. The construction of the CMN tree is based on the observations that for n members, the basic tree of Wallner, Harder, and Agee [2], requires $\mathcal{O}(\log n)$ communication updates, but has $\mathcal{O}(n)$ key storage at the center, whereas the minimal storage scheme needs only two keys to be stored but has $(n-1)$ communication updates. The CMN trees try to construct a hybrid tree in which the given group of members is partitioned into subset sets of size m, and then a basic tree of degree a is constructed with n/m leafs. Each of the leaf node of the a-ary basic tree is then assigned a unique subset. Hence, this construct is similar to building an a-ary tree of depth $\log_a(n/m)$ and then at each leaf construct an m-ary tree of depth one.

Systematic construction of the CMN trees is presented in by 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 $\log_a \frac{n}{m}$.

As noted in [1] setting $m = 1$, this combined scheme is an a-ary generalization of the binary tree scheme in [2]. When $m > 1$, the resulting CMN tree is a non-trivial extension of the basic scheme in [2]. The group center has to have a unique index into the pseudo-random for each subset in the combined scheme. Since there are $\frac{n}{m}$ subsets, the group center has to store $\frac{n}{m}$ additional keys. Total number of keys that the group center needs to store for an a-ary CMN tree with minimal subset size m, and group size n is given by $\frac{an}{(a-1)m}$.

Since the depth of the a-ary basic tree is $\log_a \frac{n}{m}$, and each leaf has minimal storage subset of size m, each member is assigned $1 + \log_a \frac{n}{m}$ number of keys. We note that the leaf key is shared by m members. Hence, when a member from subset U_i is revoked, the group center has to update the $\log_a \frac{n}{m}$ keys for all the relevant members not in the subset U_i using the update techniques of the basic tree scheme, and the the members of the subset U_i receive the updated keys by $(m-1)$ individual key updates. The total number of communication updates is $(m-1) + (a-1)\log_a \frac{n}{m}$ as shown in [1]. The proof can be done using induction but we omit it since this result is already available in [1]. The table below summarizes some interesting parameter selections presented in [1].

	general m, a	case 1	case 2
user storage	$\log_a(\frac{n}{m})$	$\mathcal{O}(\log n)$	2
GC storage	$\frac{n}{m}\frac{a}{(a-1)}$	$\mathcal{O}(\frac{n}{\log n})$	$n^{1/2}+1$
Communication	$(m-1)+(a-1)\log_a(\frac{n}{m})$	$\mathcal{O}(\log n)$	$2n^{1/2}-2$

3.2.1 The CMN Conjucture

In [1], Canetti, Malkin, and Nissim noted that the value of $m = \log_a n$ leads to the key storage of the center as $\{\frac{an}{(a-1)\log_a n}\}$ while the update communication yields $\{\log_a -1 + (a-1)\log \frac{n}{\log_a n}\}$. This can be derived using the following argument based on [1].

Consider the product denoted by S, of the key storage at the center and the communication overhead:

$$\begin{aligned} S &= (\frac{an}{(a-1)m})(m-1+(a-1)\log\frac{n}{m}) \qquad (1) \\ & \frac{an(m-1)}{m} + \frac{an\log\frac{n}{m}}{m} \\ &\approx an + \frac{an\log\frac{n}{m}}{m} \end{aligned}$$

For large values of n, if $m \geq \log_a n$, the product $S = \Theta(n)$. However, in order to keep the communication as low as possible, the value of m in $\{m-1+(a-1)\log_a(\frac{n}{m})\}$ should satisfy $m \leq \log_a n$. Hence, the optimal value of m that keeps the communication as $\mathcal{O}(\log_a n)$ while minimizing the product S of the storage and communication, is $m = \log_a n$.

Authors did not use this argument for the proof that the value of $m = \log_a n$ is the best possible for which the storage is sub-linear in n while the communication is logarithmic in n. Since the choice of the product function was not shown to be derived from a natural cost selection process, finding an optimal value of m was left as an open problem in [1].

We now show that the optimality of a given CMN tree is driven by the system parameters such as the degree of the tree, ratio of the costs of communication and storage, and the group size. We do this by constructing a convex cost function in m and show that the optimal value of the cost function is the suitable value for m. We also show under what conditions the CMN trees with sub-linear storage and logarithmic communications can be constructed.

4 Convex Optimization Based Storage-Communication Minimization

We note that indeed it is possible to solve the storage-communication problem as an optimization problem and then solve for the values of the minimal storage subset size m analytically. Our solutions are exact and not asymptotic.

In order to construct the appropriate cost function for the CMN tree, we fist denote by $f_s(m)$ the number of keys that need to be stored by the group center. We also denote by $f_c(m)$, the number of update communications under a member deletion. From the previous computations, we know that the center storage is $f_s(m) = \frac{an}{(a-1)m}$. If we set $m = 1$, the storage size is of the order of n, which is the result for the basic a-ary tree [1, 2, 17]. The function $f_s(m)$ decreases at the rate of $\frac{1}{m}$.

The communication update function for the combined scheme is([1]) $f_c(m) = (m-1) + (a-1)\log_a(\frac{n}{m})$. For large values of m, it behaves as a linear function in m. Figure 1 shows the graph of these two functions for $1 \leq n \leq 3000$.

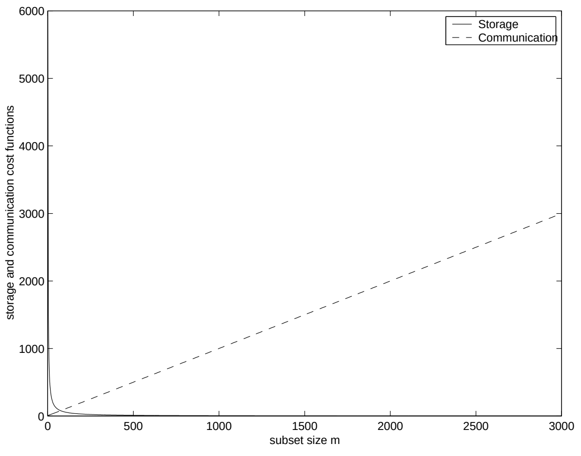

Figure 1: The graph of the group center storage and communication update functions

From the graph we note that these two functions intersect at a unique point. If we consider the sum of the communication function $f_c(m)$ and the storage function $f_s(m)$ as the total cost, then the optimal value of the total cost is the point of intersection of the functions $f_c(m)$ and $f_s(m)$. However, this simplistic interpretation assumes that the communication and the storage have equal weightage in the total cost. Since the storage is one time cost, and the communication is dynamic, one would expect the communication cost to weigh more in the construction of a total cost function.

Figure two shows the graph of $f_c(m)$ vs $f_s(m)$ as a function of m. This curve clearly shows the tradeoff between the storage and communication. This is the Pareto curve that defines the points of m.

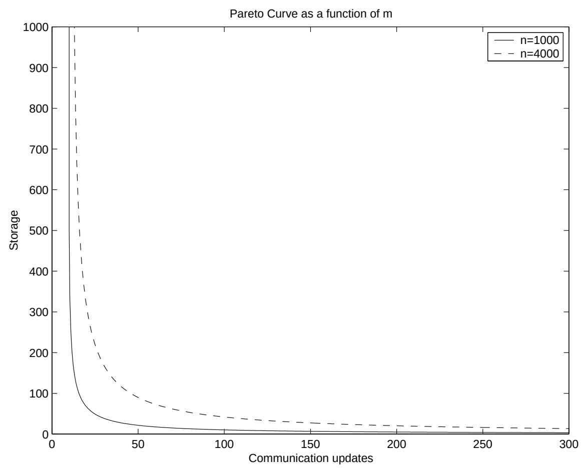

Figure 2: The Pareto curve of Communication vs Storage Tradeoff for binary tree

The cost function to be constructed needs to yield a unique minimal point. The convex functions have the property that the optimal point is also a unique minimal point. Hence, if we can construct the cost function such that it is convex in m regardless of the values of the parameters a, n, then we will be guaranteed a global minimum point for this cost function. The value of m that yields the global minimum will be the value that leads to the optimal total cost for the storage-communication tradeoff. We show that it is possible to find a cost function that considers the storage and communication, and is convex in m by constructive procedure.

We denote the weight of the storage function by $\lambda_1 \geq 0$ and the weight of the communication by $\lambda_2 \geq 0$. Then the weighted total cost function is given by

$$\begin{aligned} T &= \lambda_1 f_s(m) + \lambda_2 f_c(m) & (2) \\ &= \lambda_1 \frac{an}{(a-1)m} + \lambda_2 \{(m-1) + (a-1)\log_a(\frac{n}{m})\} & (3) \end{aligned}$$

In this formulation, we have not added any constraints on the values of λ_1 and λ_2. It is possible to set the relationship between these values by normalizing their sum and setting $\lambda_1 = 1 - \lambda_2$. We will however leave the values unconstrained except for claiming that they are non-negative in our formulation.

We first show that the total cost function constructed above is a convex function of the parameter m. In order to do so, we will use the following theorem [4] (pp. 27) without proof in a modified version for our problem:

Theorem: Let f be a twice continuously differentiable real-valued function of $m \in [1, n]$. Then f is convex *iff* its second derivative is non-negative for $m \in [1, n]$.

We now show the convexity of the total cost function by computing the derivatives with respect to m.

Computing the first and the second derivatives with respect to the variable m leads to

$$\begin{aligned} \frac{dT}{dm} &= -\lambda_1 \frac{an}{(a-1)m^2} + \lambda_2(1 - \frac{(a-1)}{m \log a}) && (4) \\ \frac{d^2T}{dm^2} &= \lambda_1 \frac{2an}{(a-1)m^3} + \lambda_2 \frac{(a-1)}{m^2 \log a} && (5) \end{aligned}$$

Since $\lambda_1, \lambda_2 > 0$, $m \geq 1$, and $a \geq 2$, we have the second derivative term $\frac{d^2T}{dm^2} = \lambda_1 \frac{2an}{(a-1)m^3} + \lambda_2 \frac{(a-1)}{m^2} > 0$. Hence, the weighted cost function $\lambda_1 f_s(m) + \lambda_2 f_c(m)$ is a convex function of m, and has a unique value of m that yields the minimum value for the weighted total cost. The minimum point is computed as the solution to the equation

$$\frac{dT}{dm} = -\lambda_1 \frac{an}{(a-1)m^2} + \lambda_2(1 - \frac{(a-1)}{m \log a}) = 0, \quad (6)$$

leading to the quadratic equation in m

$$\lambda_2(a-1)m^2 - \lambda_2(a-1)^2 \frac{m}{\log a} - \lambda_1 an = 0 \quad (7)$$

The optimal value of m, which is also the minimum is given by

$$m = \frac{(a-1)}{2 \log a}\{1 + \sqrt{1 + 4N\frac{\lambda_1}{\lambda_2}\frac{a}{(a-1)}}\} \quad (8)$$

Setting $\lambda = \frac{\lambda_2}{\lambda_1}$ leads to the optimal value of the minimal storage subset size m for the weighted total cost function as:

$$m = \frac{(a-1)}{2 \log a}\{1 + \sqrt{1 + 4\frac{N}{\lambda}\frac{a}{(a-1)}}\}. \quad (9)$$

Hence, we have shown how to compute the exact optimal value of m as a function of the group size n, degree a of the tree, and the ratio of the costs of storage and communication. We note that the optimal value of the minimal storage subset size is parameterized by the triplet (a, λ, n). We can also interpret this by considering the Pareto curve of $f_s(m)$ vs $f_c(m)$, for different values of m. Every point on this smooth curve is an optimal point for a given set of triplet (a, λ, n). For reasonably large values of n (not asymptotically large but say $n > 1000$), we can compute the optimal value of m by the approximation

$$m \approx \frac{1}{\log a}\sqrt{\frac{n}{\lambda}a(a-1)} \quad (10)$$

without losing the accuracy of the computed parameter m. We note that the computed solution for m, can be used for systematic design of several "optimal" schemes that are parameterized by the triplet (a, λ, n). We present some of the interesting schemes in the next few sections.

5 Systematic Design of Parameterized Optimal Schemes

Since the minimal storage subset size

$$m \approx \frac{1}{\log a}\sqrt{\frac{n}{\lambda}a(a-1)} \quad (11)$$

is a reasonable approximation for large values of n, it is of interest to relate this parameterized function to the known "optimal solutions" such as the one in [2]. We show that the parameterized model can be used to derive the basic scheme, and also to interpret the results with respect to the weighing functions of the communication and storage. We first interpret the basic tree scheme in [2].

5.1 Generating Basic Tree Scheme

The basic binary scheme in [2] has $a = 2$. We also know that the basic scheme has every member assigned to a unique leaf node. Hence, the minimal storage subset size is $m = 1$. From the equation $m \approx \frac{1}{\log a}\sqrt{\frac{n}{\lambda}a(a-1)}$, we have $m = 1 \approx \frac{1}{\log a}\sqrt{\frac{2n}{\lambda}}$. Hence, $\lambda = \frac{\lambda_2}{\lambda_1} = \frac{2n}{(\log 2)^2}$. If we compute the fractional weights of λ_1 and λ_2, we obtain $\frac{\lambda_2}{\lambda_1+\lambda_2} = \frac{2n}{2n+(\log 2)^2}$ and $\frac{\lambda_1}{\lambda_1+\lambda_2} = \frac{1}{2n+(\log 2)^2}$. For sufficiently large values of n, we have $\frac{\lambda_2}{\lambda_1+\lambda_2} \approx \frac{2n}{2n} = 1$, and hence $\frac{\lambda_1}{\lambda_1+\lambda_2} \approx 0$. From [2], we also note that the basic scheme did not consider the center storage as a parameter for minimization. The basic tree scheme tried to minimize only the overall communication. This is equivalent to setting the relative value of $\lambda_1 \approx 0$ (storage unit cost), and $\lambda_2 \approx 1$ in the total cost function. From the formulation above for the basic scheme, we note that as long as $\lambda_2 = \frac{2n}{(\log 2)^2}\lambda_1 > 0$, regardless of the total cost, the optimal size of the minimal storage subset is $m = 1$. The following lemma summarizes the characterization of the basic scheme:

Lemma 1. For a given binary tree with n members, if the ratio of the communication cost λ_2 and the storage cost λ_1 is chosen to be $\frac{2n}{(\log 2)^2}$, regardless of the explicit value of λ_1, λ_2, the optimal minimal storage subset size $m = 1$ for the convex total cost function $\lambda_1\frac{2n}{m} + \lambda_2 \log_2(\frac{n}{m})$. The depth of the tree is $d = \log_2 n$ and the communication updates is $\log_2 n$.

5.2 Generating Square Root Storage-Communication Schemes

There are a variety of ways to generate key schemes that will provide minimal subsets with $m = \sqrt{n}$. We list a few interesting cases here.

1. The first choice is obtained by noting that from $m \approx \frac{1}{\log a}\sqrt{\frac{a(a-1)n}{\lambda}}$, if we choose $\lambda = \frac{a(a-1)}{(\log a)^2}$, we have $m \approx \sqrt{n}$. For this value of m, the depth of the a-ary tree is $0.5\log_a n$, and the communication overhead is $(\sqrt{n} - 1 + 0.5\log_a n)$. Hence, though the storage is reduced by a factor of $\sqrt{n}$, the communication overhead grows as $\sqrt{n}$ for large values of n.

2. The second choice of a square root scheme is obtained when the key storage and the communication message updates have equal weights. Then, $\lambda = \frac{\lambda_2}{\lambda_1} = 1$. The size of the subset forming the minimal storage scheme is given by $m = \frac{\sqrt{a(a-1)n}}{\log a}$. When $a = 2$, optimal size of the minimal storage subset is $m = \frac{\sqrt{2n}}{\log 2}$. For this value, the depth of the binary tree is $d = 0.5(\log_2(n) - 1)$. The center storage is $\sqrt{2n}$, and the communication update required is $(\frac{\sqrt{2n}}{\log 2} - 1 + 0.5(\log_2(n) - 1))$.

3. The third set of square root schemes can also be generated if we set the value of $\lambda = \frac{a}{(a-1)}$, and let $a << \sqrt{n}$. Under this condition, the subset size $m = \frac{(a-1)}{\log a}\sqrt{n}$ is a linear function of the degree of the tree. The following lemma summarizes these three cases:

Lemma 2. The following cases provide a $\sqrt{n}$ growth in the storage and communication updates as the optimal choices for total cost function:

- $\lambda = \frac{a(a-1)}{(\log a)^2}$, $m = \sqrt{n}$.
- $\lambda = 1$, $m = \frac{\sqrt{2n}}{(\log a)^2}$.
- $\lambda = \frac{a}{(a-1)}$, $a << \sqrt{n}$, $m = \frac{(a-1)}{\log a}\sqrt{n}$

If the value of the $\lambda > \frac{a(a-1)n}{(\log_a n)^2}$, the derived solutions belong to a more interesting sub-linear storage family of key distribution schemes that preserve the logarithmic nature of key update communication. This is the CMN tree family introduced in [1] using asymptotic arguments. We will show how to construct such trees exactly using the triplet (a, n, λ).

6 The CMN Tree Family of Sub-linear Storage Schemes

In [1], asymptotic arguments were used for constructing a sub-linear storage scheme with logarithmic communication updates. In [1], value of $m = \mathcal{O}(\log_a n)$ was proposed for key storage minimization while keeping the communication as $\mathcal{O}(\log_a n)$. In proposing this value for m, CMN trees, authors posed the question if the sub-linear storage was the optimal scheme among all possible schemes that keep the communication overhead as $\mathcal{O}(\log_a n)$.

In terms of our model, the equivalent question is whether there is a triplet (a, λ, n) such that it is possible to design a sub-linear storage CMN tree scheme with the optimal value of the minimal subset $m = \frac{1}{\log a}\sqrt{\frac{a(a-1)n}{\lambda}}$, providing logarithmic communication overhead. Since this value of m is an *exact* (not asymptotic but computable) optimal solution for every Pareto triplet (a, n, λ), we compute the value of m that yields the optimal weighted cost. We then check if this value of m yields the storage function $f_s(m)$ that is sub-linear in n while keeping the communication overhead of the form $k_1 \log_a n$. This is sufficient for the constructive proof for the optimality of CMN scheme for a given triplet (a, n, λ). In particular, since we parameterized the minimal storage subset size m by the ratios of the computation and storage costs, there will be a family of CMN tree solutions, all of which have *exact* sub-linear storage and logarithmic communication bounds. We illustrate this below.

Existence of a sub-linear storage scheme with logarithmic update is equivalent to having positive coefficients (γ, β) such that the key storage is $f_s(m) \leq \frac{\gamma n}{\log_a n}$, and the communication updates $f_c(m) \leq \beta \log_a n$. These two conditions will relate the optimal value of m to the sub-linear storage scheme of CMN.

The constraints on the key storage at the center can be expressed as

$$f_s(m) = \frac{an}{(a-1)m} = \frac{\log a}{(a-1)}\sqrt{\frac{an\lambda}{(a-1)}} \leq \gamma \frac{n}{\log_a n}, \tag{12}$$

where $\gamma(> 0)$ is called the CMN storage constant. From equation (13), we derive the feasible value of γ as

$$\gamma \geq \frac{\log a \log_a n}{(a-1)}\sqrt{\frac{a\lambda}{(a-1)n}}. \tag{13}$$

Hence, given a triplet (a, n, λ), γ can not be a sub-linear storage scheme of CMN type if $\gamma < \frac{\log a \log_a n}{(a-1)}\sqrt{\frac{a\lambda}{(a-1)n}}$. Similarly, the constraint on the communication updates is given by

$$f_c(m) = \frac{1}{\log(a)}\sqrt{\frac{na(a-1)}{\lambda}} - 1 + (a-1)log_a \log(a) + 0.5 * \log_a(\lambda n) - 0.5 * \log_a a(a-1) \leq \beta \log_a n. \tag{14}$$

where $\beta(> 0)$ is called the CMN communication constant. This can be rewritten as a constraint on (γ, β) pair as

$$\beta \geq \frac{\frac{a}{(a-1)}\frac{\log_a n}{\gamma} - 1 + (a-1)\log_a \frac{a}{(a-1)} + (a-1)\log_a \gamma + (a-1)\log_a \frac{n}{\log_a n}}{\log_a n} \tag{15}$$

Using our approach of convex optimization, given a desirable pair (γ, β) for a candidate CMN tree, we relate the necessary conditions in terms of the triplet (a, n, λ). In doing so, we ask the following questions:

1. For a given triplet (a, n, λ), Will the chosen pair of (γ, β) satisfy the inequalities (14) and (16)?
2. If the CMN pair (γ, β) satisfies (14), and (16) is that the minimum possible pair that satisfies the equalities for (14) and (16)?

The first question tries to determine if a specific pair of positive constants (γ, β) is indeed a CMN pair. The second question tries to find if the CMN pair is indeed the optimal one for a given Pareto triplet (a, n, λ). i.e., given a Pareto triplet (a, n, λ), are the values $f_s(m) = \frac{\gamma n}{\log_a n}$, and $\beta \log_a n$ the smallest possible storage and communication updates feasible for the CMN tree.

We posed the question in two steps so that an explicit design procedure can be given for a given triplet (a, n, λ). The answer reduces to simple inequality checkings. Before deriving the inequality for testing, we present a procedure that allows explicit construction of CMN trees. Lets denote the candidate pair of the CMN pair as (γ_0, β_0). We note that the design of the optimal CMN tree for a given Pareto triplet (a, n, λ) is easier if we follow the procedure of the following type.

1. Choose the value of the Pareto triplet (n, a, λ).
2. Compute the minimum value of γ denoted as $\hat{\gamma}$, from equation (14) as $\gamma = \frac{\log a \log_a n}{(a-1)}\sqrt{\frac{a\lambda}{(a-1)n}}$.
3. If $\hat{\gamma} > \gamma_0$, it is not feasible to construct a CMN tree for a given triplet (a, n, λ) with candidate CMN pair (γ_0, β_0).
4. If $\hat{\gamma} < \gamma_0$, use $\hat{\gamma}$ in (16) to find the lowest value of β denoted as $\hat{\beta}$.
5. If $\hat{\gamma} < \gamma_0$, use γ_0 in (16) to find the lowest value on β denoted as $\hat{\beta}_0$.
6. If $\hat{\beta} > \beta_0$, then it is not feasible to construct a CMN tree for a given triplet (a, n, λ) for a candidate CMN pair (γ_0, β_0).

From the procedure, we can answer if a given pair of values (γ_0, β_0) is a CMN pair and if it is a CMN pair, if it is the optimal CMN pair for a given Pareto triplet (a, n, λ). The optimal CMN pair is the one that satisfies $\hat{\gamma} = \gamma_0$, and $\hat{\beta} = \beta_0$.

This completes the design feasibility analysis and specific design procedure of a sub-linear storage scheme with logarithmic communication overheads with candidate CMN parameters (γ, β).

7 Conclusion

In this paper, we showed that the design of key distribution trees can be formulated as a convex optimization problem with respect to the minimal storage subset size m. We then showed that the optimal value of the minimal storage subset can be analytically computed.

We showed that given the degree of the tree, group size and the ratios of the communication and the storage costs, the cost of optimal tree can be systematically designed.

We also showed how to design an important class of recently reported trees that have key storage as $\frac{\gamma n}{\log_a n}$ and the communication updates as $\beta \log_a n$, where the optimal values of (γ, β) is constraint by the triplet (a, n, λ). In doing so, we also provided an approach that will allow finding the minimum values for the parameters (γ, β). Our approach also allows the user to test if for a given triplet (a, n, λ) the candidate pair (γ, β) will form a sub-linear storage scheme with logarithmic updates. It further answers the question if this pair of (γ, β) is the minimal pair for a given triplet (a, n, λ).

We note that the formulation is exact and not asymptotic. Hence, we were also able to give specific design guidance for key distribution trees of different types. More importantly, all the lower bounds provided in [1] will be satisfied by the optimal tree designed using our approach since our approach takes into account the total cost of the tree design.

References

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

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

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

[4] R. T. Rockafeller, Convex Analysis, # 28, Princeton Matematical Series, Princeton, 1970.

[5] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication", 3rd *ACM Conf. on Computer and Communications Security"*, 1996.

[6] D. R. Stinson, and T. V. Trung, "Some New Results on Key Distribution Patterns and Broadcast Encryption", to appear in Design, Codes and Cryptography.

[7] D. R. Stinson, "On some methods for unconditionally secure key distribution and broadcast encryption", to appear in Design, Codes and Cryptography.

[8] O. Goldreich, S. Goldwasser, and S. Micali, How to construct random functions, JACM, vol.33, pp. 792-807, 1986.

[9] A. Fiat and M. Naor, "Broadcast Encryption", *Advances in Cryptology- Crypto'92*, Lecture Notes in Computer Science. vol. 773, pp. 481-491, Springer-Verlag, Berlin Germany, 1993.

[10] M. Brumester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System", *Advances in Cryptology- Eurocrypt'94*, Lecture Notes in Computer Science. vol. 950, pp. 275-286, Springer-Verlag, Berlin Germany, 1994.

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

[12] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, Multicast Security: A Taxonomy and Efficient Reconstructions, Proceedings of *IEEE Infocom'99.*

[13] D. A. McGrew and A. Sherman, Key Establishment in Large Dynamic Groups Using One-Way Function Trees, *Manuscript, 1998.*

[14] M. Luby, and J. Staddon, Combinatorial Bounds for Broadcast Encryption, Eurocrypt'98, pp. 512-526, LNCS. 1403.

[15] R. Kumar, S. Rajagopalan, A. Sahai, Coding Constructions for Balcklisting Problems without Computational Assumptions, Crypto'99, pp. 609-623, LNCS 1666, Springer.

[16] R. Poovendran, and J. Baras, An Information Theoretic Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes, Crypto'99, pp.624-638, LNCS, 1666, Springer.

[17] C. K. Wong, M. Gouda, S. S. Lam,Secure Group Communications Using Key Graphs, *ACM SIGCOMM'98*, September 2-4, Vancouver, Canada.

[18] R. Canetti, P-C. Cheng, D. Pendarakis, J. R. Rao, P. Rohatgi, D. Saha, An Architecture for Secure Internet Multicast, *Internet Draft*, November 1998.