An Information Theoretic Analysis of Rooted-Tree Based Secure Multicast Key Distribution Schemes

R. Poovendran* J. S. Baras **

Dept. of Electrical Engineering & Institute for Systems Research
University of Maryland, College Park, MD 20742, USA

Abstract. Several variations of rooted tree based solutions have been recently proposed for member revocation in multicast communications [18, 19, 20, 21]. In this paper, we show that by assigning probabilities for member revocations, the optimality, correctness, and the system requirements of some of these schemes [18, 19, 20, 21] can be systematically studied using information theoretic concepts. Specifically, we show that the optimal average number of keys per member in a rooted tree is related to the entropy of the member revocation event. Using our derivations we show that (a) the key assignments in [18, 21, 20, 19] correspond to the maximum entropy solution, (b) and direct application of source coding will lead to member collusion (we present recently proposed solutions [21, 20] as examples of this) and a general criteria that admits member collusion. We also show the relationship between entropy of member revocation event and key length.

1 Introduction

Recent research in multicast communication has created the possibility of several new business applications with potential need for secrecy and integrity of the communication ([18]-[26]). Potential commercial applications are stock quotes, special sporting events, Internet news and multimedia related applications such as conferences, etc. Due to the distributed nature and the involvement of more than two parties in these applications, there are some *unique* security related issues that are relevant only to secure multicast communications. Issues that pose significant challenges are: (a) preserving the integrity and secrecy of the communication, (b) dealing with the dynamic nature of the group membership, (c) being able to secure the intermediate nodes such as the routers, (d) graceful failure of administrative nodes, and (e) member addition/deletion.

In secure multicast communication, all the members share a common Session Encrypting Key (SK). Members of the group should also have Key Encrypting Key(s) (KEK) that can be used to update the SK in the event of membership

* Email: radha@isr.umd.edu, http://www.ece.umd.edu/~radha
** Email: baras@isr.umd.edu, http://www.ece.umd.edu/~baras

change due to any of the following reasons (a) a new member admission, (b) expiration of the SK, (c) member compromise, (d) voluntary leave, and (e) member revocation. When the membership increases, the SK may be changed to protect the back traffic and in all other cases, the SK is changed in an effort to protect future traffic. Developing efficient key update schemes while attempting to prevent member collusion and allowing the group center to perform member revocation has been the focus of several recent efforts [13, 23, 27, 28, 18, 19, 11, 21, 20]. We review two extremes of the non-tree based methods below and then focus on the rooted tree based scheme in the rest of the paper. A list of relevant papers are given in the reference section. The group size is denoted by N throughout this paper.

1.1 Non-Tree Based Key Distribution Approaches

The secure group communication requires KEKs to securely distribute the updated SK. If every member has an individual public key, for a group consisting N members, the SK update will involve $\mathcal{O}(N)$ encryption by the GC. The linear increase of the required number of encryptions in group size is not suitable for very large scale applications common in Internet, due to the amount of computational burden on the GC.

A simple way to reduce the number of encryption by the GC at the time of SK update is to provide a common KEK to all the members of the group as suggested in [23]. If the SK is to be updated due to its lifetime expiration the GC can perform a single encryption and update all the group members. If the SK is to be updated due to a new member admission, before admitting the new member, the GC may choose a new SK and the future KEK, encrypt both using the current KEK and update all the members. The newly admitted member is given the new SK and the KEK separately. However, this approach fails to support the secure communication if a single member is to be deleted/revoked. Since the whole group, including the deleted/revoked member share a single KEK, a revoked member will have access to all future key updates. Hence, this approach doesn't provide an efficient recovery mechanism for the valid members in the presence of single member failure.

In [13, 18], an approach that partition the set of keys assigned to a member into two groups was proposed. One of these sets is called the complement set and contains keys that are not distributed to a particular member. If each member has a unique complementary set, this set can be used for key updates in the event the corresponding member is revoked. The GC associates a KEK and a member in a one-to-one manner. If there are N members in the group, there will be N KEKs each representing a single member. The GC then distributes these N KEKs such that a member is given all the KEKs except the one associated with him/her. Hence, the complementary set contains a single KEK for each member. If the GC wants to delete/revoke a member, it needs to broadcast only the member index to the rest of the group. Since all members except the revoked one has the associated KEK of the revoked member, they can use that KEK for the future SK updates. This approach requires only one encryption at

the GC and allows the GC to update the SK under single member compromise. In fact this approach seem to allow even multiple member deletion/revocation. However, considering the complementary sets of any two members reveals that all the KEKs of the group are covered by the KEKs held by any two members. Hence, any two deleted/revoked members can collaborate and have access to all future conversations. Thus, under user collusion, this key scheme does not scale beyond two members. Thus the scheme doesn't have perfect forward secrecy under collusion of revoked members. This approach requires KEK storage that scales as $\mathcal{O}(N)$.

Above mentioned schemes are two extremes of KEK distribution. Depending on the degree of user collusion, a large variety of key management schemes with different number of KEKs per member can be generated.

Recently a series of papers utilizing rooted-trees for key distribution have been proposed to minimize the storage at the group controller and the members while providing a reduction in the amount of encryptions required to update the session key [11, 12, 18, 19, 20, 21, 28]. Some efficient schemes based on oneway functions also have been used on the trees for member revocation. Many of these tree based schemes seem to present different solutions to the problem with different values for the required keys to be stored at the GC and the user node. Aim of this paper to unify these results and analyze them.

1.2 Organization of the Paper

In section 2, we review the seminal work on currently known rooted tree based revocation schemes [18, 19]. We show that the approach in [18, 19, 20, 21] is related to well-known *prefix* coding in the rooted trees. In section 3, we define an appropriate notion of *member revocation* event and the associated probabilities. Using this probabilistic modeling, we show that the optimal average number of keys per member (and hence the average number of keys to be updated at the time of a member revocation) is equal to the *entropy* of the *member revocation event*. We further show that the optimal strategy is to assign a member with higher revocation probability less number of keys. Section 4 shows that although the basic idea of information theory can be used to find the **optimal number** of keys to be given to a member, trying to directly use optimal coding scheme, namely the Huffman coding, for key allocation will lead to member collusion. In order to justify our claims, we use the results in [20, 21] to show that the approaches in these two schemes can be interpreted as the Huffman coding and then show that both the schemes can be broken if two *appropriate* members collude or are compromised, regardless of the group size N. *Using the source coding part of the information theoretic approach in analyzing the key allocations in [20, 21] allows us to characterize the collusion problem with these approaches for any N and D.* In section 5 we show how to use the entropy of member revocation event to average hardware requirements of the key generating system and bound the length of the key that can be generated.

2 Review of the Logical Key Tree

Given a set group of N members and a number base D, $\log_D N$ D-ary digits are sufficient to uniquely index each of the N members in base D. This D-ary representation can also be viewed as a rooted tree representation with each member being the leaf of a D-ary tree of depth $\log_D N$. (For illustrations, we use $D = 2$ in the figures, leading to binary trees).

2.1 Distribution of Keys on the Tree

As a concrete illustration, figure 1 presents a KEK distribution based on a binary rooted tree for 8 members. In this approach, each leaf of the tree represents a unique member of the group; i.e. the leafs are in one-to-one correspondence with members. Each node of the tree represents a key. The set of keys along the path from the root to a particular leaf node are assigned to the member represented by that leaf node. For example, member M_1 in figure 1 is assigned KEKs $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$.

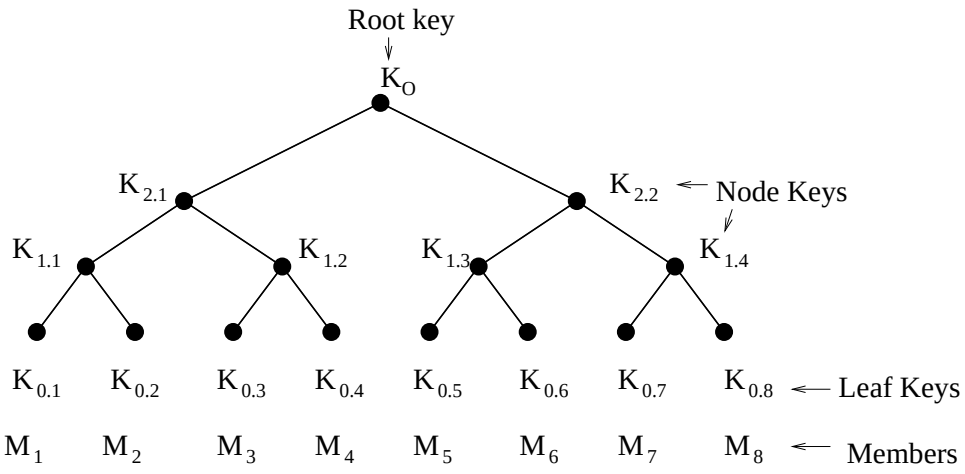

Fig. 1. The Logical Key Tree of [11, 18, 19, 20, 21]

If there is no member deletion/revocation or compromise, the common KEK denoted by K_O can be used to update the SK for all the members. The tree based structure also induces a natural hierarchical grouping among the members. By logically placing the members appropriately, the GC can choose the appropriate keys and hence selectively update, if need be, 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. Hence, the local grouping of the members and the keys shared 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 m with key K, and the notation A $\longrightarrow$ B : $\{m\}_K$ to denote the secure exchange of message m from A to B, the GC can selectively send a message m to members five through eight by the following transmission:
GC $\longrightarrow M_5, M_6, M_7, M_8 : \{m\}_{K_{2.4}}$

If the key $K_{2.2}$ is invalidated due any reason, the 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 $K_{2.2}$, 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 $K_{2.2}$ to the relevant members of the group.
GC $\longrightarrow M_5, M_6 : \{K_{2.2}\}_{K_{1.3}}$
GC $\longrightarrow M_7, M_8 : \{K_{2.2}\}_{K_{1.4}}$

2.2 Member Revocation in Rooted Trees

From now on, we will use the term keys to denote SK or KEKs unless there is a need for clarification. Since the SK and the root KEK are common to all the members in the multicast group, they have to be invalidated at each time a member is revoked. Apart from these two keys, all the intermediate KEKs of the revoked member need to be invalidated. In the event there is bulk member revocation, the GC has to

- Identify *all* the invalid keys,
- Find the minimal number of valid keys that need to be used to transmit the updated keys.

For an arbitrary tree that may not hold members in all the leafs these two problems need to be solved by exhaustive search. The principle behind the member revocation is discussed below by an example.

Member M_1 in figure 1 is indexed by the set of four keys $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$. Revoking M_1 is equivalent to invalidating these four keys, generating four new keys, and updating these keys of the appropriate valid members. When M_1 is revoked, the following key updates need to be performed: (a) all member need new K_O, (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 $\{K_{1.1}\}$.

Following observations can be made towards the rooted tree based key distributions.

- Since each member is assigned $(2 + \log_d N) = \log_d Nd^2$ keys, deletion of a single member requires $(2 + \log_d N)$ keys to be invalidated.
- Since there are $(1 + \log_d N)$ nodes between the root and a leaf and $\log_d N$ nodes are shared with other members, and for each common node one encryption is required, the GC needs to perform a total of $\log_d N$ encryptions.
- For a $d-ary$ tree with depth $h = \log_d N$, the GC has to store $1+1+d+d^2+\cdots+d^h = \frac{d(N+1)-2}{(d-1)}$ number of keys. Setting $d = 2$ leads to the binary tree

for which the required amount of storage works out to be $\frac{2(N+1)-2}{2-1} = 2N$. This result can be independently checked by noting that a binary tree with N leafs has $2N-1$ nodes. Hence the GC has to store the SK and $(2N-1)$ KEKs, leading to $2N$ keys that need to be stored.

In [20, 21], binary rooted tree based key distributions which require the GC to store a total of $2\log_2 N$ distinct keys were proposed. The generalized version of this result requires $d\log_d N$ keys to be stored at the GC. Each member needs to store only $(2+\log_d N)$ keys in this scheme. However, the number of keys to be updated remain at $\log_d N$ as in [18, 19]. Hence, at first glance, the results in [21] seem to reduce the storage requirements at the GC by

$$\frac{d(N+1)-2}{d-1} - d\log_d N = \frac{d(N+1-(d-1)\log_d N)-2}{(d-1)} \tag{1}$$

number of keys without increasing the key storage requirements at the end user node.

In the next section we present our analytical formulation to study these models in a systematic manner.

2.3 Preliminary Observations: Properties of keys on Rooted Trees

We use the approach that uses Kraft inequality in this section since we intend to derive the optimal number of keys for individual members as well. We note that the approach based on [4] can be used if we are interested in average performance analysis for a given rooted tree based scheme.

Relationship between Prefix Coding and Member Revocation Unless the set of keys held by each member differ by at least one key, the group center will not be able to successfully update the keys of the group after revoking a member. More importantly, the keys held by any member should not form a subset of the keys held by another member in rooted-tree. This is equivalent to the condition that no internal node should index a member and no two members should be indexed by the same leaf. If we view the concatenation of keys given to a member as a Key Index (KID), then each of the member should have a distinct KID. *In our definition, we consider any permutation of the KID elements as identical to the original KID. Hence there are L! equivalent KIDs for a member with L KEKs. This distinction is important in dealing with the user collusion.* (The KID for member 1 in figure 1 is represented by $K_{2.1}K_{1.1}K_{0.1}$.)

To illustrate the collusion problem by an example, let members i, j, and k have sets of keys denoted by S_i, S_j and S_k respectively, where, $S_i = \{K_0, K_1, K_2\}$, $S_j = \{K_0, K_1, K_2, K_3, K_4, K_5\}$, and $S_k = \{K_3, K_4, K_5\}$, respectively. Clearly, $S_i \subset S_j$, and $S_k \subset S_j$. In the event member j is compromised, all the keys of members i, and k are to be treated compromised. If the group center tries to use any one of the keys in the set S_i or S_k to encrypt the new set of keys for member i, and k, revoked member j can decrypt and access all the communication since

it has all the keys of i and k. In such cases, removal of one or more members, who have *all* the keys of one or more valid members, will compromise the integrity of the key updates and the privacy of the entire future communication.

Choosing unique KIDs on the rooted-trees is equivalent to the requirement that is equivalent to the statement in source coding that *no codeword should be a prefix to another codeword*. This condition can be restated as the KID corresponding (or a set of keys assigned) to a member should not be a prefix to the KIDs corresponding (or a set of keys assigned) to any other member. If not, mapping between the set of keys and the members will not be unique. Assigning a unique *prefix code* to each member on the tree leads to the following (more or less) well known important theorem that will be used later in this paper.

Theorem 1. *Kraft Inequality for KIDs*
For a D-ary Logical key tree with N members and a *codeword* generated by the concatenation of a set of keys such that no two members have the same *codewords* (set of keys) and the codeword of anyone member is not a prefix of the codeword of any other member, if we denote the codeword length (number of keys held by that member) for member i by l_i, the sequence $\{l_1, l_2, \cdots l_N\}$ satisfies the Kraft inequality given by

$$\sum_{i=1}^{i=N} D^{-l_i} \leq 1. \tag{2}$$

Conversely, given a set of numbers $\{l_1, l_2, \cdots l_N\}$ satisfying this inequality, there is a rooted tree that can be constructed such that each member has a unique set of concatenated keys with no-prefixing.

Proof: Standard and can be found in [2, 3].

Relationship Between Member Collusion and Codewords In a group of more than two members, *member collusion* needs to be prevented to preserve integrity of the group communication. We illustrate collusion using the same set of members above. If the group center needs to revoke members i and k, and update the keys for member j, revoked members i and k can collude and construct the set S_j since $\boxed{S_j = S_i \cup S_k}$. Hence, any rooted-tree structure should not permit a member to have a set of keys is a subset of the keys of other members or can be obtained as a concatenation of keys of other members. In this example, we could concatenate the key sets of members i and k to get the key set of member j. We note that a variation of rooted-tree presented in [20, 21] does suffer from member collusion. In a later section we will prove this claim and characterize the type of collusion of rooted trees [20, 21] using information theory. We note however that the codeword representation of the keys is not enough to characterize *all* types of collusions. We will discuss this point in the section 4.

3 Probabilistic Modeling

Since the key updates are performed in response to revocation of members, statistics of *member revocation events*, are very appropriate for system design and performance characterization. We denote p_i as the probability of revocation of member i with $\boxed{\sum_{i=1}^{i=N} p_i = 1}$. If the revocation were to have zero probabilities, then there is no issue of revocation of keys in the first place at all[3]. Hence, this assignment of probabilities is consistent with the motivation of key revocation.

3.1 Optimizing the Rooted Tree Structure: Optimal Codeword Length Selection

Optimization of average number of keys per members with the length of the KIDs satisfying Kraft inequality is identical to the optimal codeword length selection in the prefix coding in the context of information theory. This problem is well studied and the optimal strategy is known to yield the Shannon entropy as the average codeword length [2]. *Interpreted in the context of KID assignment, average number of keys per member is equal to the entropy of the member revocation event.* Theorem below summarizes the main results without the proof. Proof is standard in information theory and can be found in chapter 5 of [2].

Theorem 2. For a key assignment satisfying Kraft inequality, optimal average number of keys, excluding the root key and the SK, held by a member is given by the $d-ary$ entropy $H_d = -\sum_{i=1}^{i=N} p_i \log_D p_i$ of the *member revocation* event. For a member i with probability of revocation p_i, satisfying the optimization criteria, the optimal number of keys l_i, excluding the root key and the SK, is given by

$$l_i = -\log_D p_i. \tag{3}$$

The following properties which are important for member revocation and grouping of valid members to find minimal number of keys for transmission are summarized in the form of the lemma below. These are also part of standard information theory results, and are valid for the "codewords" formed by the concatenation of the keys as well:

Lemma 2.

1. Given two members of the group, the member with higher probability of revocation should be assigned larger number of keys. If $p_i > p_j$, then $l_i(= -\log_D p_i) > l_j(= -\log_D p_j)$.
2. There must be at least two members with the largest number of keys.
3. The largest number of keys held differ only by one key and these two sets correspond to the members with the least probabilities of revocation.
4. The average number of keys held by a member is never less than the entropy H_D of the member revocation event. It is equal to the entropy of the member revocation event *iff* $p_i = D^{-i}$. i.e. the probabilities are D-adic.

[3] Member addition may change some keys but does not necessarily force the change of old keys other than the traffic encrypting key.

Sketch of the Proofs:

1. If $p_i > p_j$, log being a monotone function, $\log_d p_i > \log_D p_j$. Hence $-\log_D p_i < -\log_D p_j$, leading to $l_i(= -\log_D p_i) < l_j(= -\log_D p_j)$.
2. If there are no two members with the largest codeword, then we can reduce the largest codeword by at least one more bit and still ensure that all members have unique codeword assigned. This will violate the proof of optimality of the individual codeword lengths.
3. Results follow from the fact that the optimal value of the average number of keys held by a member is a minima with the value equal to the entropy of the member revocation event.

From these statements, we note that a member with higher probability of being revoked should be given fewer number of keys. Since the number of nodes along the path connecting the leaf and the root of the logical tree represents the number of keys held by the member, the member with higher probability is closer to the root of the logical tree. The following observation summarizes the nature of the rooted key key distribution architectures in [18, 19, 21, 20].

Among all the efficient rooted tree based key revocation strategies that satisfy Kraft inequality, results in [18, 19, 21, 20] have the maximum entropy, and hence corresponds to the maximum average number of keys held by a member for a given tree size.

Upper Bounds on the Integer Values of keys Since the optimal number of keys held by a member i with probability of revocation p_i which is given by $\boxed{l_i = -\log_D p_i}$ needs to be an integer, we need to compute the exact bounds on the total number of keys (including the root key and the traffic key), assuming that the probability of member revocations can be ordered in an ascending order at the time of key assignment. This value is given by the following theorem.

Theorem 3. The optimal average number of keys held by a member satisfies $\boxed{H_D + 2 \leq \hat{l} + 2 \leq H_D + 3}$.

Proof: Result showing $H_D \leq \hat{l} < H_D + 1$ is standard [2] and is not repeated here. Adding 2 to makeup for the total number of keys $H_D + 2$) yields the desired result $\boxed{H_D + 2 \leq \hat{l} + 2 < H_D + 3}$.

Since the average number of keys per member is $(\hat{l} + 2)$, we note that the *optimal* number of average keys per member is at most 3 D-ary digits more than, and is at least 2 D-ary digits more than the *entropy of the member revocation event.*

4 Characterization of Collusion in Schemes [20, 21] Using Optimal Source Coding

Noting that we used member revocation probabilities and derived the optimal rooted-tree based key revocation schemes that eliminate redundancies in [18,

19, 21, 20], one may be tempted to conclude it may be appropriate to use the *deterministic* optimal coding techniques like the Huffman coding to develop a one-to-one map between the members and the keys assigned to them. Since the optimal number of keys led to rooted trees often called the Huffman trees, choosing the codes based on Huffman coding appears attractive from the point of using *minimal number of individual keys to construct codewords.*

We assert claim that using the Huffman coding is not the right approach when collusion needs to be avoided. To justify/make our point, we will first review the key assignment methods discussed in [20, 21] and then show that the results presented in [20, 21] for binary trees (a) fall under the category of the optimal Huffman coding and (b) provide one of the lowest possible integrity levels for member collusion.

The authors in [20] noted that given the binary index of a member, each bit in the index takes two values, namely 0 or 1. To follow the example given in [20], when $N = 8$, $\log_2 8 = 3$ bits are needed to uniquely index *all* 8 members. The authors then proceeded to claim that since each bit takes two values, it can be *symbolically* mapped to a distinct pairs of keys. The table below reproduces the mapping between the ID bit # and the key mapping for the case in [20] for $N = 8$:

ID Bit #0	K_{00}	K_{01}
ID Bit #1	K_{10}	K_{11}
ID Bit #2	K_{20}	K_{21}

where, the key pair (K_{i0}, K_{i1}) symbolically represents the two possible values of the ith bit of the member index. Although this table does provides a one-to-one mapping between the set of keys and the member index using only eight keys, the problem with this approach becomes clear if we map the table to the rooted tree structure. Figure 2 shows the mapping of the keys on the tree. (For the sake of clarity, not all the keys corresponding to the leafs are shown in figure 2). Adjacent leafs have K_{30}, K_{31} as the keys and this pair is repeated across the level. In fact, at any depth only two specific keys have been used and duplicated across the depth. If members corresponding to leafs 1, and 8 are compromised or to collude, entire set of eight keys will be exposed by these two members, i.e., this system will be *completely* broken independent of the size N of the group. Hence, this scheme is ranked *very* low in providing guarantees of privacy or integrity against collusion.

There are three different ways to interpret the collusion problems with approaches in [20, 21] based on rooted trees. We present them in the order of generality:

4.1 Interpretation based on Minimal number of Key Requirements

A simple way to interpret the shortcomings of results in [20, 21] is to note that $\boxed{2\log_2 N < N, \forall N > 4}$. In order to prevent member collusion from being able to break the rest of the system, there must be at least N keys so that each member

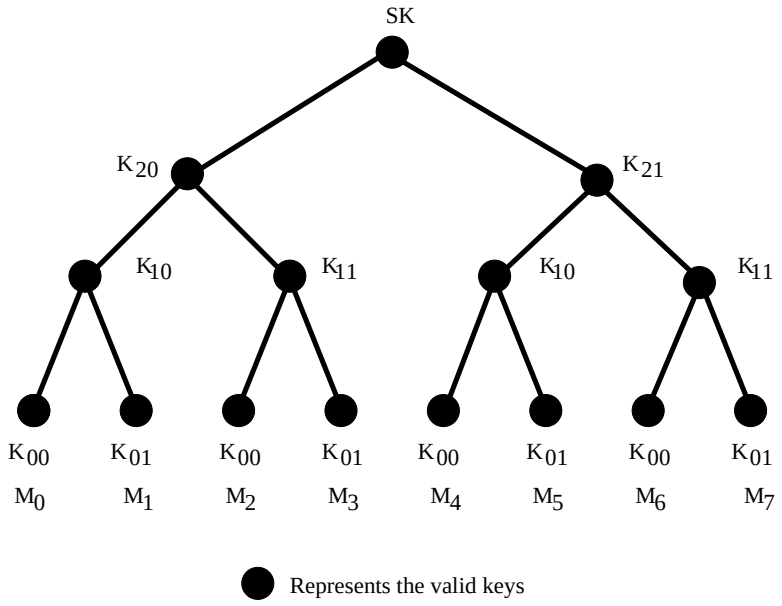

Fig. 2. The Logical Key Tree of [21, 20]

has a unique key and can be contacted at the time of member revocation. since $2\log_2 N < N$ $(N > 4)$ is the number of distinct keys used by the variation of rooted tree presented in [20, 21], and can be completely or partially compromised depending on the colluding members.

4.2 Interpretation based on Source Coding

For simplicity, we assume that the group size N is a dyadic number. Since we showed that the traditional binary rooted tree based rooted-tree [18, 19, 21, 20] leads to the maximum entropy of the member revocation event, the number of keys per member, $\log_2 N$, is same as the average number of keys per member. Also, the member indices each need $\log_2 N$ bits. The scheme in [20, 21] used a unique pair of keys to symbolically map each of bit positions of the the member index. Hence, a total of $2\log_2 N$ keys are used to uniquely represent each member index. This selection of keys can create a set of N unique indices and the codewords generated by concatenating $\log_2 N$ keys satisfy the Kraft inequality. Hence, this mapping of a unique pair of keys to each bit location corresponds to performing a Huffman coding with $2H_2(U)$ distinct keys, where $H_2(U) = \log_2 N$. If we use the notation $(k_j, \hat{k}_j)$ to denote the unique key pair representing the two possible binary values taken by the jth bit, we note that the collusion or compromise of two members holding keys k_j and $\hat{k}_j$ respectively will compromise the integrity of the key pair $(k_j, \hat{k}_j)$. The following lemmas summarize our observations:

Lemma 3. If the binary rooted key tree uses the optimal Huffman Coding for assigning members a set of keys based on $2\log_2 N$ ($N > 4$) (here N is dyadic) distinct keys as in [20, 21], the whole system can be broken if any two members whose "codewords" or KIDs (NOT UIDs as in many other recent papers) are one's complement of each other collude or are compromised. Hence, the integrity systems in [20, 21] do not scale beyond 4 members in the presence of colluding members.

In a $D-ary$ tree, each digit takes D values and the sum of these values is given by $\frac{D(D-1)}{2}$. Hence, if a set of k ($k \geq D$) members whose ith bit values when summed lead to $\frac{D(D-1)}{2}$ collude, they will be able to fully compromise the ith bit location. This result is summarized by:

Lemma 4. For a $D-ary$ tree with N members, the key corresponding to bit location b will be compromised by a subset of k ($k \geq D$) members whose symbolic value of the bit location b denoted by the set $\{b_1, b_2, \cdots, b_k\}$ satisfy $\boxed{b_1 + b_2 \cdots b_k \equiv 0 \bmod \frac{D(D-1)}{2}}$.

4.3 Interpretation Based on Complementary variables

The third interpretation is based on the notion of sets and includes a larger definition of collusion discussed under the category of complementary variables in [18]. The approach in [20, 21] is a special case of the complementary variable approach. If the secure group membership is a set such that every member is denoted by a unique key and that key is given to all other members but the member itself, at the time the member is to be revoked, all other members can use the key denoting the revoked member as the new key. For a set of N members, all the members will have $(N-1)$ keys that corresponding to other members and no member will have the key denoting itself. Clearly, if two members collude, between them they will have *all* the future keys of the group. Hence, this kind of key assignment does not scale beyond 2 members.

4.4 Appropriate mapping of the keys to the members

The main problem with the approach presented in [20, 21] is that the direct mapping of the values taken by each bit in the "codeword" to a unique pair of keys. As mentioned earlier, given a bit location h from the root of the tree, $\boxed{2^{h-1}2^{h-1} = 2^{2h-2}}$ possible combinations of members can collude to compromise the keys corresponding to that bit. In order to avoid such repeated assignment of keys, only one internal node or bit location at a given depth should be assigned a particular key. This will guarantee that only the descendant leafs of that node will have access to the specific key assigned to that node. Hence, when a member is revoked, specific key will not be part of any "unrevoked" path.

5 Entropy Based Bounds on Average Key Generation Requirements and Conditions for Clustering

We showed that on average, at the time of member revocation $(2 + H_D)$ keys need to be updated. If each key is L bits long, then the average number of bits that need to be generated by the hardware after key revocation is $L(2 + H_D)$ bits. Since $H_D \leq \log_D N$ with equality attained *iff* all the members have equal revocation probabilities, the hardware need to be able to generate a worst case average of $L(2 + \log_D N)$ bits within the next unit of time of update to let the session continue.

Theorem 4. For a binary rooted tree based rooted-tree family of systems with keys of length L bits, the average number of bits B that need to be generated by the hardware at the time of member revocation, should satisfy $\boxed{B \geq L(\log_D N + 2)}$, with the average lower bound being attained *iff* all the members have equal probability of being revoked.

Proof: As shown earlier, average number of keys to be generated in the event of member revocation is given by $\boxed{(2 + H_D) = 2 + \sum_{i=1}^{i=N} p_i l_i}$. Hence, the hardware should be able to generate a total of $L(H_D + 1)$ bits of *suitable quality*[4] in unit of time to let the session continue without delays in the average sense. Desired lower bound follows from the observation that $H_D \leq H_D(U) = \log_D N$, with equality *iff* all the members have the same revocation probabilities.

From the above given theorem, if membership is too large for a single hardware to support the key generation, there need to be at least $\lceil \frac{L(H_D+2)}{B} \rceil$ units of hardware with capability of generating B bits in a unit of time. This result can also be interpreted from the point of view of splitting a group into clusters. If a group center can update only B $(L < B < L(2 + H_D))$ bits in a unit of time, it may be appropriate to split the group center into a panel consisting of at least $\lceil \frac{L(H_D+2)}{B} \rceil$ centers each of which can update B $(L < B < L(2 + H_D))$ bits in a unit of time.

6 Conclusions and Future Work

This paper showed that several properties of the recently proposed [18, 19, 20, 21] rooted tree based secure multicast key management schemes can be systematically studied using information theoretic concepts. By using the member revocation event as the basis of our formulation, we showed that the optimal number of average keys per member is related to the *entropy* of the member revocation event. We then proved that the currently available known rooted tree based strategies [18, 19, 20, 21] yield the maximum entropy among all the rooted tree based strategies and hence opt for the maximal average number of keys per member regardless of the values of the revocation probabilities. Using the optimal source coding strategy, we identified the *collusion* problem in [20, 21] resulting

[4] Based on the application specific use of the key.

from performing the Huffman coding with $D \log_D N$ symbols. We also showed which subset of members need to collude or be compromised to break schemes such as the ones in [20, 21], regardless of the size of N. We showed that for a group with uniform revocation probabilities and using a binary tree, it is enough for two members with complementary keys to collude to break the scheme. We then showed that using the entropy of the member revocation event, we can set a bound for the minimal *average case* hardware key generation requirements. We also provided a simple rule for deciding group size based on hardware availability (or the number of hardwares required to support a size N, on average).

Acknowledgments

We would like to thank professors Anthony Ephremides and Jim Massey for early references on the rooted-tree algorithms by Massey. Relating key length to the entropy was due to observation by A. Ephremides that the keys are not shown to be related to classical information theory. We would like to thank Benny Pinkas for providing reference [1].

References

1. R. Canetti, T. Malkin, and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption", In Eurocrypt 99, pp. 456 - 470.
2. T. Cover, J. Thomas, Elements of Information Theory, John Wiley & Sons, Inc, NY, 1991.
3. R. Gallager, Information theory and reliable communication, Wiley, NY, 1968.
4. J. L. Massey, "An Information-Theoretic Approach to Algorithms", Impact of Processing Techniques in Communications, In NATO Advanced Study Institutes Series E91, pp. 3-20, 1985.
5. J. L. Massey, "Some Applications of Source Coding to Cryptography", In European Trans. on Telecom., Vol. 5, pp. 421-429, July-August 1994.
6. H. N. Jendal, Y. J. B. Khun, and J. L. Massey, "An Information-Theoretic Approach to Homomorphic Substitution", In Advances in Cryptology-Eurocrypt'89, LNCS-434, pp. 382-394, 1990.
7. U. M. Maurer, "Secret Key Agreement by Public Discussion from Common Information", In IEEE Trans. IT, Vol 39, No. 3, 1993, pp 733- 742.
8. R. Canetti, and B. Pinkas, "A taxonomy of multicast security issues", *Internet draft*, April 1999.
9. Y. Desmedt, Y. Frankel, and M. Yung, " Multi-receiver/Multi-sender network security: efficient authenticated multicast feedback", *IEEE Infocom'92*, pp. 2045-2054.
10. M. steiner, G. Tsudik, and M. Waidner, " Diffie-Hellman key distribution extended to group communication", 3rd *ACM Conf. on Computer and Communications Security"*, 1996.
11. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, "Multicast Security: A Taxonomy and Efficient Reconstructions", *Proceedings of IEEE Infocom'99.*

12. D. A. McGrew and A. Sherman, " Key Establishment in Large Dynamic Groups Using One-Way Function Trees", *Manuscript, 1998*.
13. 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.
14. A. Menezes, P. van Oorschot, and A. Vanstone, "Handbook of Applied Cryptography", CRC Press, Boca Raton, 1997.
15. M. Naor and O. Reingold, "From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs", *Advances in Cryptology-Crypto'98*, Lecture Notes in Computer Science. vol. 1462, pp. 267-282, Springer-Verlag, Berlin Germany, 1998.
16. M. Luby, Pseudo-Random Functions and Applications, Princeton University Press, 1996.
17. 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.
18. D. M. Wallner, E. C. Harder, and R. C. Agee, "Key Management for Multicast: Issues and Architectures", Internet Draft, September 1998.
19. C. K. Wong, M. Gouda, S. S. Lam,"Secure Group Communications Using Key Graphs", In *Proceedings of ACM SIGCOMM'98*, September 2-4, Vancouver, Canada.
20. G. Caronni, M. Waldvogel, D. Sun, and B. Plattner, "Efficient Security for Large and Dynamic Groups", In *Proc. of the Seventh Workshop on Enabling Technologies*, IEEE Computer Society Press, 1998.
21. I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, "Key Management for Secure Internet Multicast Using Boolean Function Minimization Techniques", To apper in Proceedings of IEEE Infocom'99.
22. S. Mittra, "Iolus: A framework for Scalable Secure Multicasting", In *Proceedings of ACM SIGGCOM'97*, pages 277–288, September 1997.
23. H. Harney and C. Muckenhirn, "GKMP Architecture", *Request for Comments(RFC)* 2093, July 1997.
24. R. Canetti, P-C. Cheng, D. Pendarakis, J. R. Rao, P. Rohatgi, D. Saha, "An Architecture for Secure Internet Multicast", *Internet Draft*, Novenber 1998.
25. T. Hardjono, B. Cain, and N. Doraswamy, "A Framework for Group Key Management for Multicast Security", *Internet draft*, July 1998.
26. B. Quinn, "IP Multicast Applications: Challenges and Solutions", *Internet draft*, November 1998.
27. H. Harney and C. Muckenhirn. "GKMP Specification". Internet RFC 2094, July 1997.
28. A. Ballardie. "Scalable Multicast Key Distribution". Internet RFC 1949, May 1996.