An Information Theoretic Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes

by R. Poovendran and John S. Baras

CSHCN T.R. 99-18
(ISR T.R. 99-36)

JB 99-24

Entitled:

An Information Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes

Authors:

R. Poovendran and J.S. Baras

Journal:

Journal of IEEE Transactions on Information Theory

An Information Theoretic Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes*

R. Poovendran, J. S. Baras
{*radha, baras*}*@isr.umd.edu*
Dept. of Electrical and Computer Engineering & Institute for Systems Research
University of Maryland, College Park, MD 20742, USA

Abstract

Recent literature presents several rooted tree based member deletion/revocation schemes [20, 21, 22, 24, 4, 5] trying to simultaneously minimize the key storage while providing efficient member deletion/revocation. Many of these approaches have different solutions and provide different values for the number of keys to be stored and distributed. In this paper, we show that many of these papers can be systematically studied using basic concepts from information theory. In particular, we show that the entropy of member revocation event, plays a major role in defining the key allocation requirements. We then relate the entropy of member revocation event to provide bounds on the key length. We also show that the optimal Huffman coding strategy used in [22, 24] leads to security weakness. A method for generating Key management schemes with varying degrees of member collusion is also presented in this paper.

Key Words: Multicast Security, Collusion, Member Deletion/Revocation, Key Length, Entropy.

1 Introduction

Many of the distributed applications like Internet newscast, stock quote updates, and distributed conferencing registration may benefit from secure group communications. Providing an effective key management scheme for these applications is complicated by the nature of a group that is *netgraphically* distributed and exhibits varying degrees of trust, i.e. different parts of the group may have different security strengths that can be assumed in key management.

A centralized Group Controller (GC) is assumed to be responsible for distributing all the required keys to the group members. In general, two or more members can use their public keys to communicate. However, if the number of messages and the number of members participating in the communication are very large, it is convenient and efficient to use shared keys. The key that is used for session encryption by the participating members is called the Session Key (SK). If the SK needs to be updated over a period of time for a variety

*Parts of this work appears in CRYPTO'99 [16] and IEEE Information theory and Networking Workshop [17].

of reasons including key lifetime expiration, compromise of the key, and/or temporary failure of one of the members, there has to be a mechanism to securely update the SK of all valid members. Although the use of public keys is one approach to achieve this goal, a specific key called the Key Encrypting Key (KEK) can also be used for this purpose. Instead of using a single KEK, each member is given a variable number of KEKs for broadcast efficiency while optimizing the storage requirements. The main focus of the current research has been to find efficient key distribution schemes without introducing vulnerabilities such as user collusion.

In a group communication model, removal or addition of one or more members does not necessarily terminate the session. Since there is more than one member involved in the communications, the group size may vary during the session due to a variety of reasons. These changes in turn prompt the SK update to prevent unauthorized access to group communications. The SK may have to be updated due to any of the following reasons:

- Expiration of the lifetime of the session key.
- Join/Admission of a member.
- Deletion/Revocation of a member.
- Voluntary leave of a group member.

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 of 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 the 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 [26]. 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, 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 one 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 a single member failure.

In [6, 20], an approach called complementary key assignment is discussed. In this approach, the set of keys are partitioned into two groups with respect to each member. 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 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. 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 $\mathcal{O}(N)$.

The 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 amounts 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 [4, 5, 20, 21, 22, 24, 31]. Some efficient schemes based on one-way 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.

The main contributions of this paper are the following:

- We show that it is possible to unify these approaches using a common analysis technique.
- We also show that the design of an optimal tree is closely related to the Huffman trees and the *entropy of member revocation event.*
- We then show that this entropy provides a bound on the providable key length if all the keys are of same length.
- We perform security analysis using entropy and show that these schemes correspond to optimal

Huffman coding and any scheme using Huffman coding for key assignment has security vulnerabilities.

- We then show how to generate a key management scheme which will safeguard against a specific amount of user collusion.

The paper is organized in the following manner. Section 2 presents the review of basic concepts behind the rooted-trees based key distribution. Section 3 presents some necessary preliminaries. Section 4 presents our formulation based on the event of member revocation and the design of the optimal rooted tree. Section 5 presents the analysis bounding the key length and the entropy of member revocation event and proposes a strategy under worst case key generation scenario. Section 6 shows that seemingly very attractive results in [22, 24] are essentially the "symbolic" Huffman coding on the full trees and then show that these two methods will completely fail if two appropriate nodes collude or are compromised. A general criteria for choosing the set of colluding members is also presented in section 6. In We conclude the paper by reviewing the one-way function based key update schemes and potential problems with them.

2 Review of the Rooted Tree Based Key Generation Schemes

Hierarchical rooted-tree based keys have been used for different applications. The first attempt at using a rooted-tree based key distribution approach for efficient member revocation was independently proposed in [20] and [21]. Modifications to reduce the computational and storage requirements of these two methods were later presented in [4, 5, 24, 22]. We will briefly review the basic concepts behind the rooted-tree based key distribution in this section.

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 16 members. In this approach, each leaf of the tree represents a unique member of the group; i.e. the leaves are in a 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 $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$.

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 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. Hence, the local grouping of the members and the keys

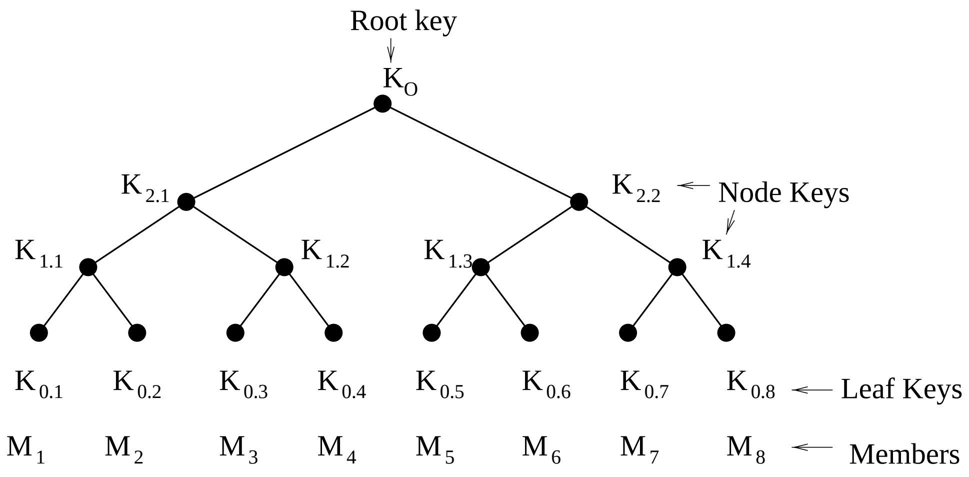
Root key
K_O
K_2.1
K_2.2
Node Keys
K_1.1
K_1.2
K_1.3
K_1.4
K_0.1 K_0.2 K_0.3 K_0.4 K_0.5 K_0.6 K_0.7 K_0.8
Leaf Keys
M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8
Members

Figure 1: The Logical Key Tree of [4, 20, 21, 22, 24]

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, GC can selectively send a message m to members five through eight by the following transmission:
GC $\longrightarrow M_5, M_6, M_7, M_8 : \{m\}_{K_{2.2}}$

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 $K_{2.2}$, 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 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 leaves these two problems need to be solved by exhaustive search. The general principle behind the member revocation is discussed below.

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 $\{K_{1.1}\}$.

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

- Since each member is assigned $(2 + \log_d N) = \log_d N d^2$ 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 leaves 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 [22, 24], binary rooted tree based key distributions which require 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 that need to be updated remain at $\log_d N$ as in [20, 21]. Hence, at first glance, the results in [24] seem to reduce the storage requirements at 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.

3 Preliminary Observations

We first show the need to optimize the rooted-tree using a worst case example. Lets consider the binary rooted-tree shown in Figure 2.

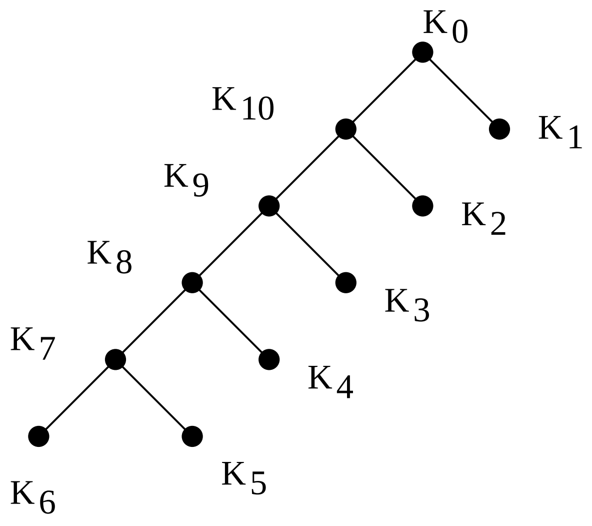

Figure 2: An Unbalanced Key Distribution

Since the SK and the root key are common to all the members, they will be invalidated each time a member is revoked. In this tree, if all the member have equal probability of being revoked, the average number of keys to be invalidated when a member is revoked is given by

$$\begin{aligned} \frac{3+4+\cdots+(N+1)+(N+1)}{N} &= \frac{N+1+N(N+1)/2}{N} \\ &= \frac{(N+1)(N+2)}{2N}. \end{aligned} \tag{2}$$

Hence, the average number of keys to be invalidated grows as $\mathcal{O}(N)$ in this model. In balanced rooted trees from [20, 21, 22, 24],the number of keys to be invalidated are of the order of $\mathcal{O}(\log_d N)$.

The key assignment problem in [4, 5, 20, 21, 24, 22] has been related to the number of members alone. The number of keys per member was assigned based on the observation that for N members $\log_d N$ keys are enough for a rooted tree.

We, however, will show that the problem of key assignment can be related to the physical process of member revocation and can be intimately related to a suitably defined *Shannon entropy* of the member revocation event. We further show some interesting properties using this approach, including security analysis. In order to do derive our formulation, we first define the terminology and show that the well-known prefix coding (and hence the Kraft inequality) plays a critical role in recovery under single member revocation.

3.1 Member Indexing

Let $X_{n-1}X_{n-2}\cdots X_1X_0$ be the binary index sequence to representing N users. Following the conventional network terminology, we call this indexing the User Index (UID). In order for the GC to be able to revoke each member and invalidate the keys, the GC has to store a member index and the corresponding set of keys assigned to that member. Hence, UID for a member has to be in one-to-one correspondence with the set of keys assigned to that member. This requirement implies that each member should be indexed using the set of keys assigned to that member. We now define the Key Index (KID) in the following manner:

Definition: Key Index (KID) of a member i is defined as the string generated by the concatenation of the keys assigned to the member i, taken in any order. If the number of keys assigned to member i is denoted by L_i, then there are L_i possible different sequences that can be generated using these L_i keys. All these KIDs are equivalent. *Hence, the KID of a member is a equivalence class with the number of $L_i!$ elements in it, where L_i is the number of keys assigned to member i.*

M_1 in Figure 1 has four KEKs and is represented by the string $K_O K_{2.1} K_{1.1} K_{0.1}$. Since there are 24 different ways to concatenate these keys, there are 23 additional strings generated by rearranging and concatenating the keys assigned to M_1. This equivalence class is the feature that separates the conventional prefix codes from what we need in this paper. Use of UID alone as in [20, 21, 22, 24] doesn't provide insights into the problems due to user collusion. The discussions on user collusion are revisited in a later section under security analysis.

3.2 Reachability and Kraft Inequality

At the time of member revocation,the GC has to be able to uniquely identify the set of keys assigned to the revoked member and invalidate the keys. After revoking a member, securely reaching the rest of the group requires that the valid member has one or more keys that are not in the set of keys assigned to the revoked member. We will call the ability of the GC to reach the valid members under user revocation as *reachability* condition. Unlike in other works that emphasize UID, we note that the KID plays a major role since it is the keys that need to be invalidated and generated.

One important necessary condition for the rooted tree based key assignment is that the KID of any member should not be a prefix *of the KID of any other member. On the rooted-tree, this leads to the well known Kraft inequality given below.*

Theorem 1: *Kraft Inequality for KIDs*
For a $d-ary$ rooted key tree with N members and KIDs satisfying prefix condition, if we denote the 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

$$\sum_{i=1}^{N} d^{-l_i} \leq 1. \qquad (3)$$

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 KID with no-prefixing.

Proof: Well known and available in [9, 10].

3.3 Limitations of Prefix Coding in Key Distribution

We now demonstrate the limitation of the prefix coding in security. Although there are many examples, the following is used to show the interplay between user collusion and the desire to prevent it. Let the set of keys $\{K_1, K_2, K_3, K_4\}$ be used to form the KIDs $\{K_2K_3K_4\}, \{K_1K_3K_4\}, \{K_1K_2K_4\}$, and $\{K_1K_2K_3\}$. These are assigned to members A, B, C, and D respectively. It can be verified that no KID is a prefix of another. Also the KID length is three and satisfies the Kraft inequality since $(2^{-3}.4) = 0.5 < 1)$. Since no KID is a prefix of another, if a single member is revoked, there is at least one key for each of the remaining member that is not in the set of the compromised member. Hence, a single user deletion/revocation does not invalidate *all* KEKs of other members.

However, if two or more members are to be simultaneously revoked, the above mentioned key scheme completely fails since there will be no a single KEK that is not invalidated.

Hence, the selection of KIDs satisfying prefix condition does not guarantee any safe guarding against failure of the key management scheme or user collusion

On the other hand, the **KIDs satisfying the prefix condition do help to solve another important problem, namely the optimal key allocation per member**. We present the needed formulation in the next section. This optimal assignment is very closely tied to the underlying physical process of member revocation as shown in the next section.

4 Probabilistic Modeling of Member Revocation

Since the key updates are performed in response to member revocation, statistics of *member revocation event*, is appropriate for system design and performance characterization. Hence, the statistics of member revocation should be linked to the assignment of KIDs of a member. It may be noted that we are not making any claim about the specific selection of any key at this stage. We denote p_i as the probability of revocation of member i.

4.1 Relating the Probability of Member Revocation to the Keys on the Rooted Tree

The physical process of member revocation is related to the rooted trees via the leaf nodes using the following observations.

- Since each member in the rooted tree is assigned to a unique leaf, the probability of revocation of a member is *equivalent* to the probability of revocation of the corresponding leaf node.
- Since all the nodes of the rooted tree are assigned a unique key, the probability of revocation of the leaf node is also the probability of revocation of the key represented by the leaf node. Hence, we note that the probability p_i of revoking member i is equivalent to having the probability p_i of revoking the key at the leaf i.
- Since every member has a unique KID and the KIDs are formed by concatenating the keys assigned to a member, when the member is deleted/revoked the KID is also revoked/invalidated. Hence, the event of member revocation is equivalent to the event of revocation/invalidation of the KID of member.

The following assumptions are implicit in the models presented in [20, 24, 22] and are useful in the derivation of the optimal number of keys to be assigned to each member.

- *Assumption 1:* Revocation of members is an independent event.
- *Assumption 2:* The number of members N is a fixed quantity.

We recognize that the second assumption is somewhat restrictive and can at best satisfy only one temporal "snap shot" of the real world requirement. Implicit in this assumption is the property that the tree structure is fixed over the entire session. In deriving the optimal number of keys to be assigned per member, however we will continue to assume that N represents the number of members in the group (One way to remove this constraint is to set N as the maximal allowed members though there are additional technical details about the KIDs with infinite length is needed. We will ignore this issue in this paper since it is not critical in the analysis).

The assumption that the member revocation event is indepent allows simple computation of the probabilities of revocation of all the intermediate node keys on the tree. Let the kth child node of an intermediate node i have the probability of revocation p_{ik}. If the individual member revocations are statistically independent, the following equation presents the probability of revocation p_i of the intermediate node i of a $d-ary$ rooted tree.

$$p_i = \sum_{k=1}^{d} p_{ik} \qquad (4)$$

Hence, starting from the revocation probabilities of the leaf nodes, one can compute the probabilities of revocation of all the intermediate nodes. Using the recursive nature of the rooted tree structure, every probability of revocation of any key corresponding to an internal node can be expressed in terms of the probabilities of the member revocation.

4.2 Defining the Shannon Entropy of Member Revocation Event

In physical processes that involve probabilistic modeling, one can often define the uncertainty of the occurance of an event using a suitably defined entropy of the process. We will use Shannon entropy [9] to express the amount of uncertainty as to which member will be revoked. We first state the definition of the *Shannon entropy* in the context of member revocation event.

Definition: We define the $d - ary$ entropy H_d of the member revocation event by

$$H_d = -\sum_{i=1}^{N} p_i \log_d p_i \tag{5}$$

where p_i is the probability of revocation of member i. As mentioned earlier, the entropy expresses the uncertainty as to which member will be revoked in $d - ary$ digits.

A word of caution is in place since the Shannon entropy is often used to describe the rates in the source coding literature. We use it in the context of its physical interpretation which is the amount of uncertanity about the occurance of an event.

Since the member revocation event and the leaf node key revocation event are probabilistically identical, entropy of the member revocation event is same as the entropy of the leaf key revocation event. This important observation is summarized as a theorem below.

Theorem 2. *Leaf Key Revocation Entropy* is the entropy or uncertainty as to which of the leaf key will be revoked. Since the leaf key revocation probability is in one-to-one correspondence with the member revocation probabilities, Leaf Key Revocation entropy is identical to the entropy of the member revocation event.

Hereafter we will use the term entropy of member revocation event instead of leaf key revocation entropy since they are equivalent.

Another very useful observation is that since the member revocation event is also probabilistically equivalent to the KID revocation event, the entropy of member revocation event is identical to the entropy of the KID revocation event.

A main outcome of these observations is that the entropy of the KID revocation event is identical, and can be completely characterized once the entropy of the leaf key revocation event (which is equivalent to the member revocation event) is known.

With the probabilistic model, below we show that we can

- derive the optimal number of keys per member.
- analyze the collusion properties of some schemes.

- derive a bound on the length of the keys.
- determine if a given rooted key scheme can sustain its key generation rates

4.3 Assigning Optimal Number of Keys per Member

Since the SK and the root key are common to all the members, these two keys don't contribute to the optimal key assignment strategies. In formulating the optimal number of keys per members, the GC should try to minimize the storage requirements without making any explicit assumptions about the nature of the keys to be chosen. If such a formulation is possible, then that optimal key assignment strategy may be used to relate the storage requirements to system parameters such as the probabilities of member revocation.

From the view point of GC, one strategy is to minimize the average number of keys per member with the additional conditions that the KIDs of the members should satify the Kraft inequality[1].

Optimization of the 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 [9]. *Interpreted in the context of KID assignment, the average number of keys per member is equal to the entropy of the member revocation event.* We summarize the result as Theorem 3 without repeating the proofs [9].

Theorem 3. 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 $H_d = -\sum_{i=1}^{N} p_i \log_d p_i$ 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 TEK, is given by

$$l_i{}^* = -\log_d p_i. \tag{6}$$

Since the SK and the root key are common to all the members, optimal average number of keys per member is given by $H_d + 2$, and the number of keys assigned to member i with revocation probability p_i, including the SK and the root key is given by

$$l_i^* + 2 = -\log_d p_i + 2 = \log_d \frac{d^2}{p_i}. \tag{7}$$

The following properties that are very useful in identification of the minimal number of keys that can be used after member revocation are summarized in the form of the lemma below. They are also satisfied by the optimal number of keys held by a member.

[1]Although the prefix strategy provides protection against only a single member failure, it is the only one that to our knowledge is mathematically viable to analysis.

Lemma 2.

1. A member with higher probability of revocation should be given fewer keys compared to a member with lower probability of being revoked. 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. Since the number of keys assigned per member needs to be integer, *true* average number of keys per member is more than the optimal value, and is not more than d additional keys.

In order to derive the last part of the lemma, we need the following definition from information theory [9].
Definition: The *relative entropy* or the *Kullback Leibler* distance between two probability mass functions $p(x)$ and $q(x)$ is defined as

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}. \tag{8}$$

Sketch of the Proofs:

1. The logarithm being a monotone function, if $p_i > p_j$, then $\log_d p_i > \log_d p_j$. 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 number of keys, then we can reduce the largest number of keys held by at least one and still ensure that all members have unique set of keys assigned. However this reduction will violate the proof of optimality of the individual codeword lengths. Hence, at least two members should be assigned largest number of keys.

3. In the earlier derivation we showed that the entropy is the point of optimality, and is indeed a global minimum of the average number of keys held by a member. Since the number of keys need to be integer, the average number of extra keys is given by

$$\begin{aligned}
\sum_{i=1}^{i=N} p_i l_i - H_d &= \sum_{i=1}^{i=N} p_i l_i + \sum_{i=1}^{i=N} p_i \log_d pi & (9)\\
&= \sum_{i=1}^{i=N} p_i \log_d \frac{p_i}{d^{-l_i}} & (10)\\
&= \sum_{i=1}^{i=N} p_i \log_d (\frac{p_i}{\frac{d^{-l_i}}{\sum_{j=1}^{j=N} d^{-l_j}}}) + \log_d (\frac{1}{\sum_{j=1}^{j=N} d^{-l_j}}) & (11)\\
&= D(p||q) + \log_d (\frac{1}{\sum_{j=1}^{j=N} d^{-l_j}}) & (12)
\end{aligned}$$

where $q_i = \frac{d^{-l_i}}{\sum_{j=1}^{j=N} d^{-l_j}}$.

We don't repeat the well known result that the information divergence is non negative [9].

In summary, the optimal key allocation strategy requires that the member with revocation probability p_i be assigned $\boxed{(2 - \log_d p_i)}$ number of keys. In the case of binary rooted trees, the optimal number of keys for a member with probability of revocation p_i needs to be assigned $(2 - \log p_i)$ keys.

The results indicate that there are at least two members with the largest KID also indicates that the tree is *packed.* i.e., if a member is revoked, all the complementary keys of that member are needed to securely reach the rest of the members. If there are bulk member removals, the set of keys that are in the complementary set of the revoked members can be used to securely update the valid members. Depending on the nature of the specific key choices it is possible to develop a fast algorithm for key updates.

4.4 Maximum Entropy and the Key Assignment

The results reported in [20, 21, 4, 24, 22] present a rooted tree with all members having the same number of keys. Since the optimal number of keys for a member i with probability or revocation p_i is $(2 - \log_d p_i)$,this assignment is equivalent to treating $(2 - \log_d p_i = \text{constant})$ for all values of i. Hence, the results in [20, 21, 24, 22, 4] assume that the probability of revocation is uniform for the entire group. Since the uniform distribution *maximizes* the entropy and entropy is the average number of keys per member under a optimal strategy, the schemes in [20, 21, 4] assign maximal set of keys per member. We summarize these results by the following theorem.

Theorem 4. Average number of keys per member is upper bounded by $(2 - \log_d N)$ and this value is reached when all the members have equal probabilities of being deleted/revoked.

Proof We showed that the average number of keys per member is $(2 + H_d)$ where $H_d = -\sum_{i=1}^{i=N} p_i \log_d p_i$. The Shannon entropy H_d is maximized [9] when the probabilities of the event is uniform. Hence, the entropy of member revocation event is maximized when all the members have equal probability of being revoked/deleted.

Since the schemes in [4, 20, 21, 22, 24] implicitly assume uniform member revocation probabilities, these schemes correspond to the worst case or the maximum entropy solution for key assignments for individual members. These schemes assign $(2 + \log_d N)$ keys per member, where N is the group size.

We note here that the use of maximal number of keys per member does not imply that the key distribution scheme is free of any possible member collusion or even secure. We elaborate on this point later.

4.5 Relationship to the Oneway Function Based Key Selection Schemes on the Tree

In [6], Fiat and Naor presented a set of broadcast key distribution schemes with pre-specified degree of collusion. One of their scheme used one-way functions for generating the keys on the rooted tree for the group members. New results on using pseudorandom, and one-way functions were reported recently in [5, 4] in the context of rooted tree based multicast key management schemes. These schemes use either oneway or pseudorandom functions to generate the keys. They don't address the issue of finding the optimal number of key assignment. Recent work reported in [23] parameterized the minimal number of messages as a function of the number of keys. The bounds derived in [23] can be tightened using the optimal key lengths we derived in this paper.

Since there can't be any more entropy than that provided by the *member revocation event* and we have incorporated that in our optimization problem and minimized the average number of keys needed to be stored, for a very large multicast group, our formulation based on entropy will yield the lowest average cost of key generation. Only way to further reduce the communication overhead in key generation is to introduce relationships (for example using pseudorandom functions) among the keys generated as in [4].

Hence, among *all* rooted-tree based key schemes, our formulation will yiled the lowest average number of keys per member. From information theoretic view point, it is a very well known result that the optimal code lengths of the prefix codes is obtained by minimizing the average codeword length and the optimal average codeword length is given by the entropy.

4.6 Upper Bounds on the Integer Values of keys

The optimal number of keys for a member with probability of revocation p_i in a $d-ary$ rooted tree key management scheme was shown to be

$$2 - \log_d p_i. \tag{13}$$

Since this quantity corresponds to the number of physical keys, it has to be an integer value. The following theorem summarizes the bound on the optimal number of keys to be held by a member. If we denote the integer value of the average number of keys, excluding the SK and the root key, held by members by $\hat{l}^*$, the bounds on the optimal number of keys per member are given by the following inequalities:

Theorem 5. The optimal average number of keys held by a member satisfies

$$H_d + 2 \leq \hat{l}^* + 2 < H_d + 3. \tag{14}$$

Proof: Using the notation $\lceil -\log_d p_i \rceil$ to represent the smallest integer greater than or equal to $-\log_d p_i$, we have the integer value of l_i^* as

$$l_i^* = \lceil -\log_d p_i \rceil. \tag{15}$$

For this choice of l_i^*, it can be shown [9], that $\boxed{\Rightarrow H_d \leq \hat{l}^* < H_d + 1}$. Hence, $\boxed{H_d + 2 \leq \hat{l}^* + 2 < H_d + 3}$.

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.*

4.7 Effect of Using Incorrect Entropy on Key Length

In Figure 2 we presented the effect of an unbalanced rooted tree on the number of keys to be assigned and to be invalidated. We note that this quantity can be completely characterized using basic results from information theory as well.

Lets us assume that the true revocation probability of member i is p_i and the used probability of revocation for member i is q_i. Hence, the optimal number of keys to be assigned to that member is given by

$$l_i^* = \lceil -\log_d q_i \rceil \tag{16}$$

Using an incorrect distribution introduces redundancy in the number of keys that are assigned to the members. This redundancy is given by the following theorem.

Theorem 6. The average number of keys per member under the true distribution p with the number of key selection based on $l_i = -\log_d q_i$ satisfies the following bounds

$$H_d(p) + D(p||q) \leq L < H_d(q) + D(p||q) + 1, \tag{17}$$

where $L = \sum_{i=1}^{i=N} p_i l_i$.

Proof: Standard and can be found in [9].

Hence, on average the number of redundant keys assigned to a member due to the use of an incorrect distribution is given by the inequalities $\boxed{D(p||q) \leq \{L - H_d(p)\} < D(p||q) + 1}$.

Apart from being closely related to the optimal key assignments, the entropy of member revocation event is also related to the sustainable key length of the secure multicast group, as shown next.

5 Impact of Revocation on the Key Length

In the current public literature of key generation, the key length is shown to be a function of the amount of computational power an adversary has or the duration of the use of the key. Assuming the quality of the distributed keys being good, a recent study report presents the choice of key length as a function of the computational power of the adversary alone.

We now show that the admissible key lengths in rooted-tree based key distributions can be bounded by the system parameters like the entropy of the member revocation event and the group size.

5.1 Bounds on Average Key Length

We showed that the member revocation event is probabilistically equivalent to the KID revocation event. We also showed that the entropy of the member revocation event is identical to the entropy of the KID revocation event. In this section, we will present the relationship between the entropy of member revocation event and the sustainable key length of the rooted tree based key distribution schemes.

When there is a member revocation, the average number of keys to be invalidated is given by $(2 + H_d)$. If each key is $L\ d - ary$ digits long, then in order to update these keys, the total number of digits that need to be generated by the GC after member revocation is $L(2 + H_d)$ digits. 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 on average of $L(2 + \log_d N)$ digits within the next unit of time of update to let the session continue. The following theorem summarizes this result.

Theorem 7. For a $d - ary$ rooted tree key distribution scheme in which each key is of length L digits, if the hardware digit generation rate is given by B, then the key length L is bounded by the following inequalities: $\frac{B}{2-\log_d p_{min}} \leq L \leq \frac{B}{2-\log_d p_{max}}$.

Proof: As shown earlier, number of keys to be regenerated in revoking member i with probability of revocation p_i is $\boxed{(2 - \log_d p_i)}$. Hence, the hardware should be able to generate $L(2 - \log_d p_i)$ digits of *suitable quality*[2] in unit of time to let the session continue without interruptions. Hence, the hardware digit generation rate B must satisfy $B \geq L(2 - \log_d p_i)$.

$$\begin{aligned} p_{min} &\leq \frac{1}{N} \leq p_{max} \qquad (18) \\ \log_d p_{min} &\leq \log_d \frac{1}{N} \leq \log_d p_{max} \\ (2 - \log_d p_{min}) &\geq (2 + \log_d N) \geq (2 - \log_d p_{max}) \\ \frac{B}{2 - \log_d p_{min}} &\leq \frac{B}{2 + \log_d N} \leq \frac{B}{2 - \log_d p_{max}} \end{aligned}$$

Hence the key length that can be generated by the hardware is bounded by $\boxed{\frac{B}{2-\log_d p_{max}} \leq L \leq \frac{B}{2-\log_d p_{min}}}$.

Since it is of interest to make sure that the system sustains the secure communication mode, the system should satisfy the worst case key generation scenario. Hence the hardware digit generation rate B needs to satisfy $\boxed{B \geq L(2 - \log_d p_{max})}$.

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

If all the members have equal probabilities of being revoked, $p_{min} = p_{max} = \frac{1}{N}$. Hence, the needed hardware bit generation rate is given by $B \geq L(2+\log_d N)$. Most of the current solutions [20, 21, 24, 22, 4, 5] assume this case.

Often key length is chosen as a function of external parameters such as the available computational power of the communicating parties as well as the attacker and risk vs. time trade off. In light of the key length bounds presented using the optimal number of key assignments for rooted-trees, any key length chosen based on other external considerations for rooted-trees should also satisfy these bounds.

In the next section, we show how to make use of the entropy of member revocation event to characterize and interpret the vulnerabilities of some recently reported schemes.

6 Security Analysis of Some Recent Results Using Member Revocation Entropy

Noting that we used member revocation probabilities and derived optimal rooted tree based key assignments which are also known as Huffman trees, one may be tempted to conclude that it may be appropriate to use *deterministic* optimal coding techniques like 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 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 now show this is not the case. In doing so, we use two recent schemes [22, 24] which use optimal Huffman coding for key assignment. We first revisit the issue of user collusion and then present and analyze these two schemes.

6.1 User Collusion Problem Revisited

We noted that the KID of one member should not be a subset of the KID of any other member. This condition ensures that the revocation of one member does not expose all the keys of a valid member. However, this is not the only case under which member revocation will expose the keys of valid member(s). It is possible that one or two members are simultaneously revoked or compromised. If the set of keys held by the revoked members can cover the set of keys held by one or more valid members, the corresponding keys of the valid members should be treated as exposed.

For example, let members i, j, and k have the set of keys $S_i = \{K_1, K_2, K_3\}$, $S_j = \{K_1, K_2, K_3, K_4, K_5, K_6\}$, $S_k = \{K_2, K_4, K_5, K_6, K_7\}$ respectively. In this case, the group controller can use K_2 to reach all the members when no member needs to be revoked. If the member i were to be revoked, the keys $\{K_2, K_4, K_5, K_6\}$ will be exposed. The group controller can still securely contact members j, and k using keys not in the set S_i. If the member j were to be revoked, the keys of member i will be exposed to member j, but the group controller can still securely communicate with member k. If the members i and j were to be revoked simultaneously, although the sets S_i, and S_k can't individually cover the set S_j, if the revoked users were to collude they can cover the set S_j. Moreover, the group controller can never securely communicate with member j without having either the member i or k also receiving the message. In this example, the $S_i \in S_j$ and hence, KIDs satisfy $\{K_1, K_2, K_3\} \in \{K_1, K_2, K_3, K_4, K_5, K_6\}$ Although $S_j \in S_i \cup S_k$, we can't directly use the KIDs to express this regardless of how we order the individual keys in the KID string. Hence, the KIDs can be used to express only a limited type of security problem.

We now show that assigning keys such that the KIDs formed by keys satisfying Kraft inequality is not sufficient to prevent additional security problems by the following example.(We assume that i, j, and k are three members of a larger group). To do so, we modify the set of keys assigned to member j, to $S_j = \{K_1, K_2, K_4, K_5, K_6\}$. It can be checked by inspection that none of the sets S_i, S_j and S_k are subsets of the other. Since $(2^{-3} + 2^{-5} + 2^{-4}) < 1$, by the converse of the Kraft inequality, we can also construct a binary tree for distributing the keys. This tree does ensure that each member has unique KID and if any one of the members is compromised, the group controller can still securely communicate with the other two members. However, if the members i and k are to collaborate, between them they can cover the keys of the members j. Not only can they cover the keys of member j, they can still ensure the integrity of their communication with the group controller if they don't expose the keys K_3 and K_7. Hence, we note that the Kraft KID satisfying prefix coding doesn't imply that the key assignment scheme is free of security vulnerabilities.

6.2 Description of the Schemes in [22, 24]

The authors in [22] 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 [22], 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 [22] 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 $(K_{i,0}, K_{i,1})$ 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 3 shows the mapping of the keys on the tree. (For the sake of clarity, not all the keys corresponding to the leaves are shown in Figure 3). Adjacent leaves have K_{20}, K_{21} 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.

In approaches such as [22, 24] that use UID to optimal Huffman coding, a special case of member revocation brings these key management scheme to halt. This happens if the members M_0 and M_7 need to be revoked. The corresponding keys to be revoked are shown in Figure 4. These two members have only the session key in common. However, if these two members need to be simultaneously revoked, the group controller is left with *no key* to securely communicate with the rest of the valid members. This reduces the rooted tree to the GKMP [26]. The compromise recovery for this case requires that the entire group re-key itself by contacting one member at a time.

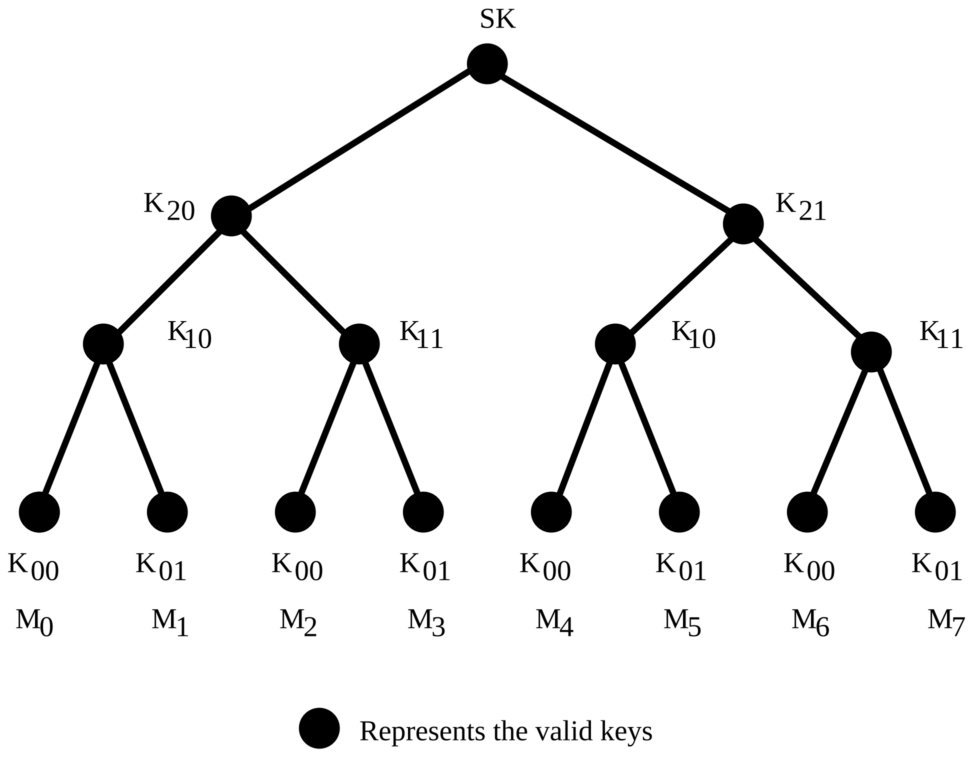

Figure 3: The Key Distribution in [24, 22]

The key assignments in [22, 24] and their variations also allow the members to collaborate and break the system. We now discuss the user collusion on the rooted tree in [22, 24].

6.3 Impact of Member Collusion on Rooted Trees

We showed that if more than one member were to be revoked, the whole key scheme may be compromised. There are three different ways to interpret the collusion problems with approaches in [22, 24] based on rooted trees. We present them in the order of generality:

6.3.1 Interpretation based on Minimal number of Key Requirements

A simple way to interpret the shortcomings of results in [22, 24] 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 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 [22, 24], and can be completely or partially compromised depending on the colluding members.) However, when $N = 4$, $2\log_2 N = 4$. The scheme in [22, 24] breaks down if members M_0 and M_7 were to be simultaneously compromised. This is illustrated in Figure 4.

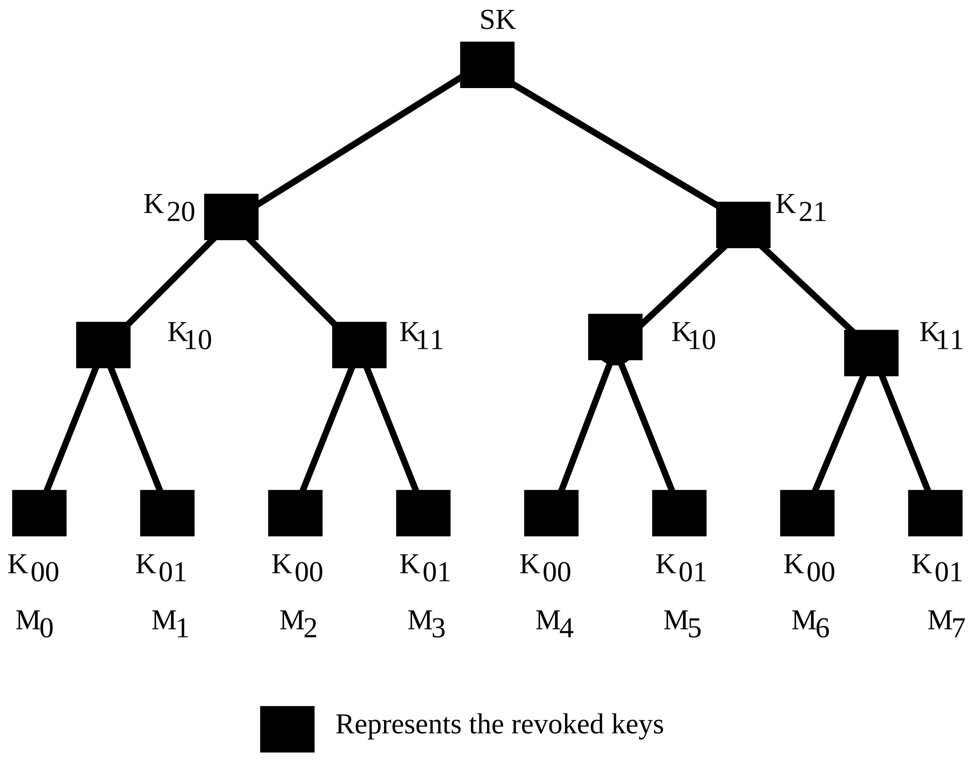

Figure 4: Revocation of Members M_0, M_7 in [22, 24].

6.3.2 Interpretation Based on Complementary variables

The second interpretation is based on the notion of sets and includes a larger definition of collusion discussed under the category of complementary variables in [6, 20]. The approach in [22, 24] 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 correspond 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.

6.3.3 Interpretation based on Huffman Coding

We showed that the average number of keys per member is given by entropy. We also showed that if the distribution is uniform, the average number of keys per member attains its maximum value. When the member revocation probabilities are equal, the number of keys assigned to a member is the same as the average number of keys per member. We also showed that this strategy is used in [20, 21, 24, 22].

The schemes in [22, 24] mapped the UIDs to KIDs directly. Since the number of bits needed for N members is $\log_2 N$, the schemes in [22, 24] used a unique pair of keys to symbolically map each of the 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 $H_2(U) = \log_2 N$. However, the problem with Huffman coding is that it is uniquely decodable! Hence, a key assignment based on the direct mapping of bit locations to keys will lead to serious security exposure. In fact, an attacker can break the whole system by breaking the members whose indices are *all* ones and *all* zeros. These two members represent all possible bit patterns and hence have all the $2\log_2 N$ keys among themselves.

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 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 [22, 24], the whole system can be broken if any two members whose "codewords" (and hence indices) are one's complement of each other collude or are compromised. Hence, the integrity systems in [22, 24] 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 $\boxed{b_1 + b_2 \cdots b_k \equiv 0 \bmod \frac{d(d-1)}{2}}$.

6.4 On Generating a Large Class of Key Management Schemes with Varying Degree of Collusion

From our analysis of the tree based schemes, we note that many different key management schemes with different levels of protection against the user collusion can be made. On one extreme, the keys representing the rooted tree have no relationship, leading to a very high degree of integrity but also higher storage requirements. On the other extreme, all members share the same keys as in GKMP [26] leading to the system failure in the event of a single member failure. The schemes in [22, 24] fail with the collusion of two members or can fail at different bit levels depending on the index of the colluding members. Depending on how many digit locations are represented as $k - ary$ digits. The Figure 5 shows the comparison between various schemes.

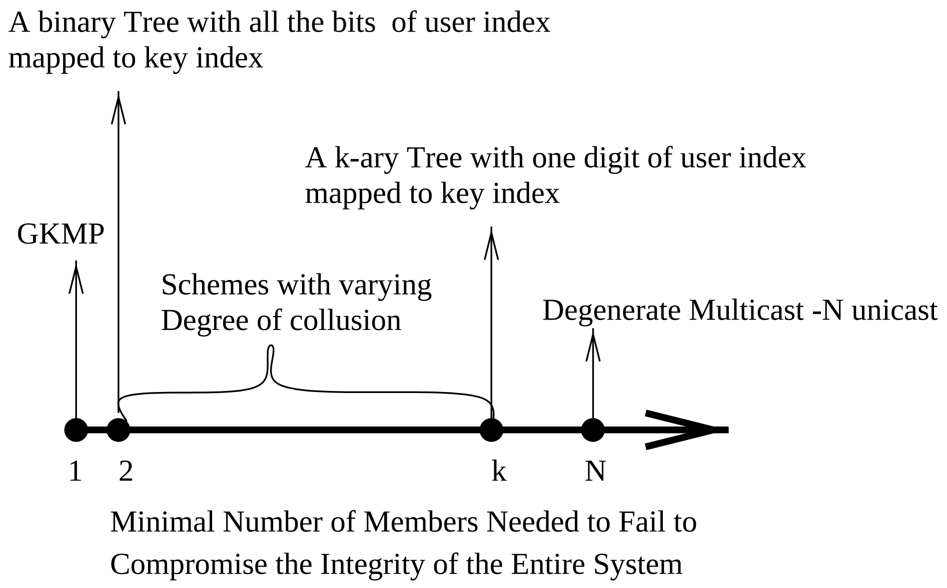
A binary Tree with all the bits of user index mapped to key index
A k-ary Tree with one digit of user index mapped to key index
GKMP
Schemes with varying Degree of collusion
Degenerate Multicast -N unicast
1
2
k
N
Minimal Number of Members Needed to Fail to Compromise the Integrity of the Entire System

Figure 5: Effect of User failure of different schemes

7 Conclusions and Future Work

This paper showed that several important properties of the recently proposed [20, 21, 22, 24] rooted tree based secure multicast key management schemes can be systematically studied using basic 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 [20, 21, 22, 24] 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 [22, 24] resulting 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 [22, 24], 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 worst case* hardware key generation requirements. We also showed that our approach can be used to reduce the number of keys to be generated using oneway or pseudorandom functions. Future work will address developing fast algorithms for member index grouping and utilization factor analysis.

Acknowledgements

We would like to thank Sanjeev Khudanpur, Himanshu Khurana, and Diane Lee for their comments on the papers. We thank Benny Pinkas for pointers to the paper by Canetti, Malkin, and Nissim. We also thank A. Ephremides and J. Massey for providing early references on the rooted-tree based analysis of algorithms by Massey.

References

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

[2] Y. Desmedt, Y. Frankel, and M. Yung, "Multi-receiver/Multi-sender network security: efficient authenticated multicast feedback", *IEEE Infocom'92*, pp. 2045-2054.

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

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

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

[6] 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.

[7] A. Menezes, P. van Oorschot, and A. Vanstone, "Handbook of Applied Cryptography", CRC Press, Boca Raton, 1997.

[8] 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.

[9] T. Cover, J. Thomas, Elements of Information Theory, John Wiley & Sons, Inc, NY, 1991.

[10] R. Gallager, Information theory and reliable communication, Wiley, NY, 1968.

[11] 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.

[12] J. L. Massey, "Some Applications of Source Coding to Cryptography", In European Trans. on Telecom., Vol. 5, pp. 421-429, July-August 1994.

[13] 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.

[14] D. R. Stinson, and T. V. Trung, "Some New Results on Key Distribution Patterns and Broadcast Encryption", Manuscript, November 11, 1997.

[15] U. M. Maurer, "Secret Key Agreement by Public Discussion from Common Information", In IEEE Trans. IT, Vol 39, No. 3, 1993, pp 733- 742.

[16] R. Poovendran, and J. S. Baras, "An Information Theoretic Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes", in CRYPTO'99, August 1999, Santa Barbara, USA.

[17] R. Poovendran, and J. S. Baras, "An Information Theoretic Approach to Multicast Key Management", in Proceedings of IEEE Information theory and Networking Workshop, Metsovo, Greece, June, 1999.

[18] M. Luby, Pseudo-Random Functions and Applications, Princeton University Press, 1996.

[19] 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.

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

[21] 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.

[22] 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.

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

[24] 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.

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

[26] H. Harney and C. Muckenhirn, "GKMP Architecture", *Request for Comments(RFC)* 2093, July 1997.

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

[28] T. Hardjono, B. Cain, and N. Doraswamy, "A Framework for Group Key Management for Multicast Security", *Internet draft*, July 1998.

[29] B. Quinn, "IP Multicast Applications: Challenges and Solutions", *Internet draft*, November 1998.

[30] H. Harney and C. Muckenhirn. "GKMP Specification". Internet RFC 2094, July 1997.

[31] A. Ballardie. "Scalable Multicast Key Distribution". Internet RFC 1949, May 1996.