99-03

1999 IEEE

International Symposium on Information Theory

Metsovo, Greece
June 27- July 1, 1999

Sponsored by
the IEEE Information Theory Society
Supported by
Aristotle University of Thessaloniki
INRIA, NSF, Intracom, Panafon

IEEE NSF INRIA

An Information Theoretic Approach to Secure Multicast Key Management

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

Abstract — **New results are presented for recently proposed rooted tree based secure multicast key revocation schemes [1, 2] by studying the information theoretic properties of *member revocation* events. It is shown that the optimal average number of keys per person is given by the entropy of the member revocation event and the currently available solutions correspond to the worst case or the maximum entropy scenario. It is shown that the proposed key assignment in [2] corresponds to optimal source coding and is susceptible to attack by compromise or collusion of multiple members.**

I. Introduction

Secure multicast communication requires all the members to share a common key to encrypt the traffic. In the event of a member compromise, it becomes necessary to revoke the old traffic keys and securely distribute the new traffic keys to remaining members of the group. Another important constraint is that members (revoked or otherwise) should not be able to *collude* among themselves and construct the (future or other) keys used by the group. Contacting individual members and securely updating the traffic keys takes $\mathcal{O}(N)$ computations at the group center node. Noting that all N members can be uniquely indexed using $\log_d N$ d-ary digits, a scheme based on d-ary rooted tree of depth $\log_d N$ has been proposed [1, 2]. Each of the member indices are directly mapped to a unique leaf of a d-ary tree and a set of keys representing the intermediate nodes of the tree between the root and each leaf is assigned to individual members.

Figure 1 illustrates the rooted tree scheme for $N = 16$ with the leaf key indices also representing the indices of the group members. For example, member 1 is indexed by the set of five keys $\{K_O, K_M, K_I, K_A, K_1\}$. In this structure, the whole group can update its keys in $\mathcal{O}(\log_d N)$ encryptions at the group center and one decryption at each leaf node. *e.g.*, if member 1 is to be revoked, keys $\{K_O, K_M, K_I, K_A, K_1\}$ need to be updated for relevant members. The following encryptions and transmissions achieve this task: (a) members with indices 9-16 receive new K_O encrypted with key K_N, (b) members with indices 5-8 receive new $\{K_O, K_M\}$ encrypted with key K_J, (c) members with indices 3-4 receive new $\{K_O, K_M, K_I\}$ encrypted with key K_B, (d) member with index 2 receives new $\{K_O, K_M, K_I, K_1\}$ encrypted with key K_2.

II. Information Theoretic Formulation

Since key updates are done in response to revocation of members, statistics of *member revocation events* are very appropriate for system design and performance characterization. We define p_i as the probability of revocation of member i; $1 \leq i \leq N$ with $\sum_{i=1}^{i=N} p_i = 1$.

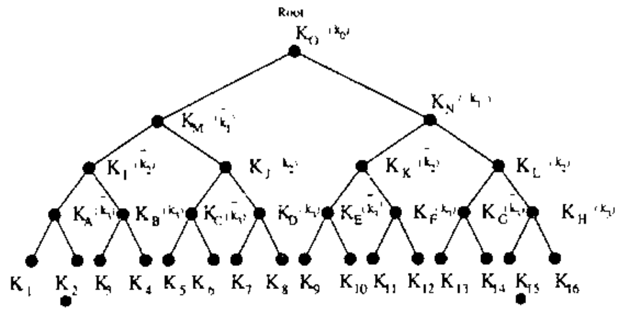

Figure 1: The rooted key distribution tree
Key allocation corresponding to [2] are shown in paranthesis
A possible collusion pair is shown by members 2 and 15

III. Main Results

In order to make sure that revocation of a single member does not render *all* keys held by any other member invalid, no member should have its keys embedded in the set of keys held by another member. This is equivalent to unique decodability in source coding.

Theorem 1: If the concatenation of the set of keys held by each member is viewed as a "codeword" then this collection of codewords satisfy the Kraft inequality.

Theorem 2: Optimal average number of keys per member is given by the *entropy* $H_d = -\sum_{i=1}^{i=N} p_i \log_d p_i$ of the member revocation event.

This source-coding viewpoint reveals that currently proposed key management schemes [1, 2] have redundant key allocation. They correspond to the worst case or *maximum entropy* situation where each member is equally likely to be revoked.

Attack by colluding members: The key allocation scheme of [2] uses a set of $2\log_2 N$ distinct keys as shown in Figure 1. This is shown to correspond to an optimal (Huffman) source coding procedure with $2H_d$ distinct keys. We show that this scheme can be attacked by two colluding or compromised members binary representations of whose indices are *one's compliments* of each other.

Theorem 3: Independent of the value of N or d, the optimal key assignment discussed above allows the integrity of the whole key management scheme to be broken by compromise of or collusion among appropriate subsets of members.

References

[1] D. M. Wallner, E. C. Harder, and R. C. Agee. "Key Management for Multicast: Issues and Architectures". expired Internet Draft, July 1997.

[2] 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.

[3] 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.

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

R. Poovendran, J. S. Baras
Dept. of Electrical and Computer Engineering
Center for Satellite and Hybrid Communication Networks
Institute for Systems Research
University of Maryland, College Park, MD 20742, USA

Abstract

Recent literature presents several seemingly different approaches to rooted tree based key distribution schemes [3, 4, 5, 22, 23] that try to minimize the user key storage while providing efficient member deletion. In this paper, we show that the user key storage on the rooted trees can be systematically studied using basic concepts from information theory. We show that the key distribution problem can be posed as an optimization problem that is closely related to the optimal codeword length selection problem in information theory. In particular, we show that the *entropy of member deletion event* quantifies the average number of keys to be assigned to a member. We then relate the sustainable key length to statistics of member deletion event and the hardware bit generation rate. We also present an example of a key distribution scheme that attains the optimality criteria but fails to prevent user collusion [4, 5].

Key Words: Multicast Security, Collusion, Member Deletion, 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. In order to minimize the sender as well as the network resources all members of the group should share a single Session Key (SK) for data encryption. In order to preserve the secrecy of the communication, at any instant of secure communication, only the valid group members must posses the session key [22, 23]. Unlike a pairwise communication model, the secure group communication has to consider additional factors to ensure that only the valid members have access to the communications. We present the details of this claim below.

In case there are two communicating members, when any one of them leaves the group, the session terminates and the session key can be invalidated. If the group consists of more than two members, "leave" or "join" or failure of a single member may not necessarily imply the termination of the group communication.

*Parts of this work appears in CRYPTO'99 [18] and IEEE Information theory and Networking Workshop [19]. Contact: *radha, baras* isr.umd.edu

Since the secrecy of the communication requires that only the valid members of the group possess the session key, the task is to develop key distribution methods that will allow the group to update the new session key without having security vulnerabilities such as illegal collaboration among present or past members to obtain all or parts of the keys of the valid members. This illegal collaboration among present or past members is noted as *user collusion* in this paper. Another important problem that a key distribution has to address is that the deletion of one or more members should not invalidate all the keys of a valid member.

The use of public keys is a solution to ensure that only the valid members have the session keys at any time. Under this assumption, when there is a need to change the session key, the entire group is individually contacted and provided with the new session key. This solution however, has computation that grows as $\mathcal{O}(N)$, where N denotes the group size throughout this paper. Since the desire is to use the minimum amount of sender and network resources, more *efficient* solutions need to be developed for key distribution for large groups.

In this paper, we study the model in which there is a single Group Controller (GC) that is assumed to be responsible for generating and distributing all the required suitable cryptographic keys [15] to the group members. We use the term Key Encrypting Keys (KEK) for keys that are used to encrypt other cryptographic keys. Efforts to develop scalable session key update techniques with minimum key storage requirements for the user as well as the Group Controller (GC) have been reported in the recent past [1-5, 7, 8, 10, 11, 16-22]. Among different session key update approaches, one family of key distribution approach [22, 23] based on rooted trees has been studied in detailed due to its scalability. Focus of this paper is to show that the key distribution on the rooted trees can be related to well founded results in information theory. We first present some of the non-tree based approaches for group communications.

1.1 Non-Tree Based Key Distribution Approaches

In [10], an approach that makes use of a common key encrypting key was proposed. Under this model, if the session key needs to be updated due to its lifetime expiration, the GC generates a new session key, encrypts it with the common key encrypting key, and broadcasts it to the entire group. If a new member joins the group, the GC generates new SK and key encrypting key, encrypts them with the current key encrypting key and updates the entire group. The new member is then individually updated with the new SK and the key encrypting key. This approach prevents the new member from getting direct access to past group communication. Although this approach handles the member join and the key updates due to key life time expiration, it was not designed to address the key updates under member deletion. If a single member is deleted[1], the session key has to be updated. However, the key encrypting key that is used to update the session key is shared by all members including the deleted member. Hence, the GC has to

[1]In this paper, we use deletion for member removal, voluntary leave of a member or deletion after detection of a compromise since every one of these will invalidate the keys assigned to that member and the session key of the group.

individually contact the members of the group to re-key. Although this approach requires one encryption and two key storage at the GC as well as the user for a static group, it has to rely on (N-1) additional individual keys every time a member is deleted.

In [22], an approach called complementary key assignment is discussed. Under this approach, each member is uniquely identified by a single key. The key corresponding to a member is distributed to all other members. Hence, no member will have the knowledge of the key that uniquely identifies that member. When a member is deleted, the GC simply broadcasts the index of the deleted member to the group. The group then sets the key corresponds to the deleted member as the new session key. Since the deleted member does not have access to this key, the future communications is not directly accessible to the deleted member. This approach requires (N-1) keys to be stored by every member. The GC doesn't need to perform any encryption under this model. Under this key distribution model, if two or members are deleted simultaneously, the group is left with no key that will be secure for future communication.

Another model is to form all possible sets of members and assign a unique key to every set. Members belonging to a set store the corresponding key. For a group of size N, there are 2^N possible set of members and hence there are 2^N possible keys to be stored by the GC. Since every member is in $2^{(N-1)}$ of these sets, every member has to store $2^{(N-1)}$ keys. In this model, if a member is deleted, then the GC has to identify the set that contains all the valid members and broadcast the index of that member. This will allow the valid members to use the appropriate key. Since the deleted member didn't have access to the specified group, there will be not direct access to the future communications.

The above mentioned schemes represent the extremes of KEK distribution. In [22, 23], tree based schemes that require each member to store $\log_d N$ number of keys for a d-ary tree were proposed. Several variations [3, 2, 4, 5] have been developed with additional modifications. Efforts have been made to develop key distribution schemes that will provide a scalable solution while providing acceptable tradeoffs in storage and communications [2]. The reference [2] also represents a detailed study towards understanding the communication-storage tradeoffs for the GC and is the first known document that provides a sub-linear storage. The scheme uses a hybrid approach and uses pseudo-random functions [12] in establishing some of the keys. We study methods that don't make assumptions about the keys in our derivations and hence don't include results in [2] in our analysis.

We note that the use of information theory for key agreement has been studied in [16]. Use of information theory in the context of homomorphic substitutions and coding have been studied by Massey [13]. Use of information theory in alayzing algorithms was presented in [13]. Stinson used information theoretic techniques in the context of boradcast key distribution [21].

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 deletion 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 necessary preliminaries. Section 4 presents our formulation based on the event of member deletion and the design of the optimal rooted tree. Section 5 presents the analysis bounding the key length and the statistics of member deletion event and proposes a strategy under worst case key generation scenario. Section 6 demonstrates the user collusion on the optimal trees using [4, 5]. A general criteria for choosing the set of colluding members is also presented in section 6.

2 Review of the Rooted Tree Based Key Distribution Schemes

The first use of rooted tree based key distribution approach for secure multicast communication was independently proposed in [22] and [23]. A rooted binary tree was used in [22] and key graphs [23] were used in [23]. Both these approaches construct a logical tree [22] or key graph [23] based on the size of the group without making any assumption regarding the relationship among the keys. The group controller key storage requirements of these two schemes grow as $\mathcal{O}(N)$ while the user key storage and the update communication requirements grow as $\mathcal{O}(\log n)^2$.

2.1 Distribution of Keys on the Tree

The Figure 1 presents a rooted binary key distribution tree for a group with eight members. The logical tree is constructed such that each group member is assigned to a unique leaf node of the tree. Every node of the logical tree is assigned a key. The set of keys assigned to the nodes along the path from a leaf node

[2]The recent literature presents approaches that make certain assumptions about the relationship among the keys and significantly reduce the group controller key storage requirements [3, 4, 5] while maintaining the user key storage and the update communication as $\mathcal{O}(\log_d N)$.

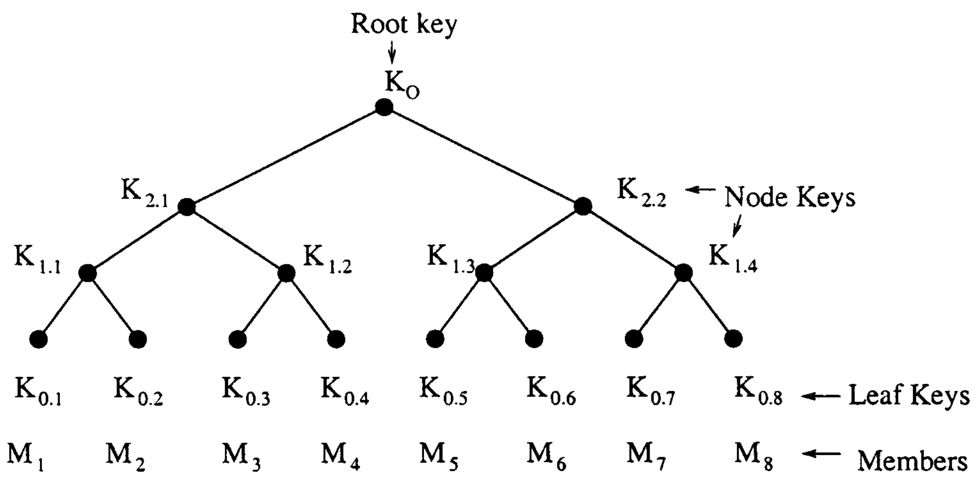
Root key
K_O
K_2.1
K_2.2
Node Keys
K_1.1
K_1.2
K_1.3
K_1.4
K_0.1 K_0.2 K_0.3 K_0.4 K_0.5 K_0.6 K_0.7 K_0.8
Leaf Keys
M_1 M_2 M_3 M_4 M_5 M_6 M_7 M_8
Members

Figure 1: The Logical Key Tree of [3, 4, 5, 22, 23]

to the root are assigned to the member associated with that particular leaf node. For example, member M_1 in Figure 1 is assigned key encrypting keys $\{K_O, K_{2.1}, K_{1.1}, K_{0.1}\}$.

Since the root key K_O is also shared by all the members, if there is no change in group membership, K_O can be used to update the session key (SK) for all the members. Some of these schemes use [5] the root key as the session key.

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

Using the notation $\{m\}_K$ to denote the encryption of message m with key K, and the notation A $\longrightarrow$ B : $\{m\}_K$ to denote the secure exchange of message m from A to B, GC can selectively send a message m to members $M_5, \cdots M_8$ by the following transmission:
GC $\longrightarrow M_5, M_6, M_7, M_8 : \{m\}_{K_{2.2}}$

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

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

2.2 Member Deletion in Rooted Trees

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

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

The 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 each node is assigned a key, and every member shares $\log_d N$ nodes with at least one more member, the total number of key encrypting keys to be updated under a member deletion is $\log_d N$.
- 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 is $\frac{2(N+1)-2}{2-1} = 2N$. 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 [4, 5], binary rooted tree based key distributions which require GC to store a total of $2\log_2 N$ distinct keys were proposed. For a d-ary tree, extension of results in [4, 5] requires $d \log_d N$ keys to be stored at the GC. However, the number of keys that need to be updated under a single member deletion remains at $\log_d N$ as in [22, 23]. Hence, the results in [5] 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

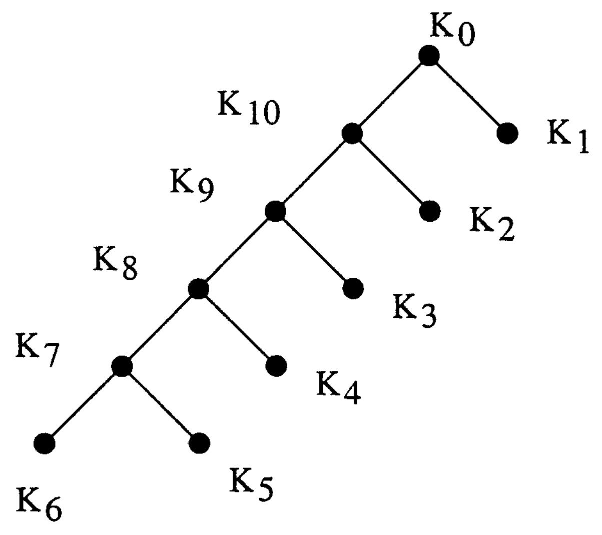

Figure 2: An Unbalanced Key Distribution

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

$$\frac{\sum_{i=1}^{N-1}(i+1)+N}{N} \approx \frac{N}{2}+2. \tag{2}$$

Hence, the average number of keys to be invalidated grows as $\mathcal{O}(N)$ in this model. In [3, 4, 5, 22, 23] the logical tree was built based on the group size. Every member was assigned keys based on the observation that for N members $\log_d N$ keys are sufficient for a d-ary rooted tree.

If we consider the number of keys to be stored by a member as an *efficiency parameter*, a natural question to be posed is if the optimal bound on the efficiency parameter is $\log_d N$ when there is no explicit relationship among keys. We show that the optimal bound on the efficiency parameter can be characterized by studying the member deletion process. We also show that the optimal key assignment problem is abstractly equivalent to optimal codeword length selection problem. In order to do so, we first define the necessary terminology and show that the well-known prefix coding (and hence the Kraft inequality) provides a necessary condition for key distribution under single member deletion.

3.1 Cover Free Key Distribution

In assigning keys to members, the GC needs to ensure that every valid member can be securely reached under member deletion. The GC also needs to make sure that collaboration among two or more members does not enable them to *cover* the keys assigned to a valid member. This cover free property has been used in the context of broadcast encryption and traitor tracing in [8, 11]. We use it in the context of tree based key distribution. Main idea behind the cover free property of the keys in our context is that any set of valid or deleted members should not be able to collaborate and recover all the keys of any other valid member(s). Let the set S_j denote the keys assigned to member M_j and the set $S = \{S_1, S_2, \cdots, S_N\}$ denote the set of all key sets. Let $C = \{S_{C_1}, S_{C_2}, \cdots, S_{C_l}\}$ denote the set of the key sets of the colluding members. Then, we can express the condition for cover free of a set S_i as

$$S_i \not\subseteq \bigcup_{S_j \in C; j \neq i} S_j. \tag{3}$$

In the case of key distribution, the condition that the set of keys assigned to a member be *cover free* of union of any number of other sets of keys is the main feature that allows member deletion as well as collusion resistance feasible. The condition imposed by (3) is satisfied if the set S_i has at least one key K_i^*, such that

$$\begin{aligned} K_i^* &\in S_i \\ K_i^* &\notin S_j \vee j \neq i. \end{aligned} \tag{4}$$

Being cover free without additional relationship among keys also implies that the number of keys to be stored grows at least as $\mathcal{O}(N)$.

3.2 Member Deletion Process

In order to delete a member, the GC has to have a lookup table that contains a unique index and the set of keys assigned to that member. Let $X_{n-1}X_{n-2}\cdots X_1X_0$ be the binary User Index (UID). Since the GC should be able to securely communicate with every valid member after a member deletion, no two members should be assigned identical set of keys. Indeed, every member should be assigned a unique set of keys, some of which are shared with other members. Since a member M_i has unique UID and deletion of member M_i invalidates a unique set of keys assigned to M_i, the UID of a member has a one-to-one correspondence with the set of keys assigned to that member. Hence, if we concatenate the set of keys assigned to a member and form a key index (KID)[3], the UID of a member needs to be in one-to-one correspondence with the KID of that member.

Although the KID and the UID need to be in 1:1 correspondence, the KID needs to satisfy a much more stringent version of the cover free property. We illustrate this by the following example. Consider the alphabets $\{0, 1\}$ used for UID generation and the keys $\{K_1, K_2\}$ used for KID generation. The UIDs 01

[3]In forming the KID, we ignore the root key and the session key that are common to all the members. Unless explicitly mentioned, we also ignore these two keys in all the computations that follow.

and 10 can be generated and assigned uniquely to two different members. The KIDs K_1K_2 and K_2K_1 however can not be assigned to two different members. If we do assign them to two different members, the keys assigned to a member can be completely covered by the keys assigned to the other member. Although this is a special type of covering, this is crucial in defining the KIDs. Any permutation of the keys forming a KID will lead to a KID that is completely *covered* by the original KID. We use this property in defining the KID.

Definition: Key Index (KID) of a member M_i is defined as the string generated by the concatenation of the keys assigned to the member M_i, *taken in any order.* If the number of keys assigned to member M_i is denoted by L_i, then there are $L_i!$ possible different KID sequences that can be generated using these L_i keys. Given a KID, all the KIDs that are generated by permuting and concatenating its keys are equivalent under cover free property. If the set of keys of KID_1 is denoted by S_1, and the set of keys of KID_2 is denoted by S_2, we say $\text{KID}_1 \equiv \text{KID}_2$ if $S_1 = S_2$.

The member M_1 in Figure 1 has four KEKs and is represented by the string $K_OK_{2.1}K_{1.1}K_{0.1}$. Since we ignore K_O, the KID is $K_{2.1}K_{1.1}K_{0.1}$. Since there are six different ways to concatenate these keys, there are five additional KIDs generated by permuting and concatenating the keys forming KIDs of M_1. This equivalence of among the KIDs generated by permuting a set of keys is a feature that separates the conventional UIDs from KIDs.

3.3 Key Indexing and Kraft Inequality

Since the UIDs and the KIDs of a member need to be unique, they do satisfy a variation of the *cover free* property. This is the condition that the KID (UID) of any member should not be a *prefix* of the KID (UID) of any other member. On the rooted-tree, this condition is related 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 forming the KID of member M_i by l_i, the sequence $\{l_1, l_2, \cdots l_N\}$ satisfies the Kraft inequality given by

$$\sum_{i=1}^{N} d^{-l_i} \leq 1. \tag{5}$$

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 [6, 9].

While this prefix free condition is necessary and sufficient for indexing a member using UID, this is only a necessary condition for the KID. We first illustrate this difference. Consider 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 $\{K_1K_2K_3\}$ assigned to members M_1, M_2, M_3 and M_4 respectively. It can be verified that no KID is a prefix of another. Also the KID lengths satisfy the Kraft inequality since $(2^{-3}.4) = 0.5 < 1)$. Since no KID is a prefix of another, if a single member is deleted, 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 does not invalidate *all* keys of other members. It can be verified that the union of any two sets of KIDs will cover the elements of the all other KIDs. Hence, any two members can collaborate and obtain the keys of all other members. Moreover, if the GC deletes two or more members the keys of the rest of the group will be invalidated. *Hence, the prefix free property is only a necessary condition for being able to reach a valid member under member deletion.*

We note that this result can be explained by the fact that the Kraft inequality is a property exhibited by the tree structure and is independent of the nature of the elements of KID. *Hence, the selection of KIDs satisfying prefix condition does not guarantee any safeguarding against failure of the key distribution scheme under member deletion or user collusion*[4].

3.4 Making KIDs Cover Free

We noted the fact that if a KID assignment is cover free, it is prefix free. A natural question to ask is *how to construct a prefix free KID assignment on the tree that is also cover free.* In order to provide a condition that a KID is cover free, we note that even if (N-1) members were to collude or to be deleted, the cover free condition for the valid member M_i with key set S_i is

$$S_i \not\subseteq \bigcup_{S_j \in S; j \neq i} S_j. \tag{6}$$

If we let $S_i^* = S_i \setminus \bigcup_{s_j \in S; j \neq i}(S_i \bigcap S_j)$ denote the set of keys that belong to S_i and don't belong to any other set, the condition $S_i \not\subseteq \bigcup_{S_j \in S; j \neq i} S_j$ is satisfied if $|S_i^*| \geq 1$, i.e., the set S_i contains at least a single element that is not contained in any other set. Since the case of interest is minimum number of keys assigned, it is sufficient to set $|S_i^*| = 1$. In order to be cover free, this condition must be satisfied for every member. Hence, every key set needs to be assigned a unique key that is not shared by any other member.

In order to construct the tree based key assignment scheme that satisfy this condition, we consider the manner in which the members are assigned to the logical tree. Each member is assigned to a unique leaf node. There is a unique path from the leaf to the root of the tree. Every member shares all the keys other than the leaf node key with at least one more member. Hence, choosing the leaf node keys to be distinct will make sure that every key set S_i has at least one element that is not covered by the union of any other key sets.

[4]An example is [5].

Hence, if we *choose all the leaf node keys to be distinct*, the tree based prefix free KID assignment will be *necessary and sufficient* to (a) prevent user collusion from disabling the secure communication, and also (b) reach a valid member under deletion of arbitrary number of members. Since there are N leaf nodes, the number of keys to be stored by the GC grows as $\mathcal{O}(N)$ when there is no additional relationship among keys[5].

The optimal key assignment is very closely tied to the underlying physical process of member deletion as shown in the next section.

4 Towards Optimizing the Keys Assigned to a Member

Using l_i to denote the length of the KID of member M_i, we note that in the rooted tree based key distribution model [22, 23], member M_i shares l_i number of keys with two or more other members. This count includes the session encryption key and excludes the leaf node key of member M_i. In the event the member M_i is deleted, number of keys to be updated is also l_i. Hence, if we can minimize l_i, this will minimize the user key storage.

In the rooted tree based key distribution of [22, 23], each member shares $(1 + \log_d N)$ keys with two or more members. At the time a member is deleted, $(1 + \log_d N)$ keys are to be updated and communicated to other members. Only key that need not be updated is the leaf node key of a deleted member. Ignoring the constants, we note that the user key storage and the keys to be updated under member deletion grows as $\mathcal{O}(\log_d N)$ for the rooted tree based key distribution schemes in [22, 23] that do not make use of the physical process of member deletion in assigning the keys. We now show that the use of the statistics of the member deletion process will enable us to further reduce the user key storage and, hence the key update requirements of the rooted tree based models [22, 23].

4.1 Relating Statistics of Member Deletion Process to the Key Distribution on the Rooted Tree

Let us denote the probability of deletion of member M_i by p_i. We assume that this probability is computable either empirically or is known a priori. This paper does not make attempt to develop methods for computing p_i. Noting that every time a member is deleted, all the keys assigned to that member are deleted, we make the following observations:

- Since every member is assigned to a unique leaf node, and every leaf node is also assigned a unique key, probability p_i of deletion of a member M_i is *identical* to the probability of deletion of the leaf

[5]Reduction of the key storage requirements of the group controller as a sub-linear function of group size was presented in [2] using pseudo-random functions. In our study, we assume that the keys are distinct and have no relationship among them.

node key assigned to M_i.

- Since every member has a unique KID and the KIDs are formed by concatenating the keys assigned to a member, probability of deletion of member M_i is *identical* to the probability of deletion of the KID of member M_i.

Hence, we note that the probability of deletion of a member is identical to the probability of deletion of its node key as well as its KID. Given the knowledge about the probability of member deletion, we define the *entropy of member deletion* by the following formula [6]:

Definition: The $d - ary$ entropy H_d of the member deletion is

$$H_d = -\sum_{i=1}^{N} p_i \log_d p_i \tag{7}$$

where p_i is the probability of deletion of member M_i. A word of caution is in place since this formula of 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 uncertainty about the member deletion event.

Since the member deletion event and the leaf node key deletion event have identical probabilities, the d-ary entropy of the member deletion event is same as the entropy of the leaf key deletion event. This important observation is summarized as a theorem below. Similarly, the entropy of KID deletion is identical to entropy of member deletion. We will use the term entropy of member deletion event instead of entropy of leaf key deletion or entropy of KID deletion since they are equivalent.

A main outcome of these observations is that the *entropy of the KID deletion* is identical to the *entropy of member deletion*, and hence, can be completely characterized once the entropy of member deletion is known.

4.2 Assigning Optimal Number of Keys per Member

When a member M_i is deleted with a probability p_i, the group controller has to generate and update l_i keys that were shared with other members. Hence, on average, the group controller has to generate and update

$$\sum_{i=1}^{n} p_i l_i \tag{8}$$

number of keys. This is also the average number of keys that a member needs to be assigned. As noted earlier, we have chosen not to count the session key and the root key that are updated for every member deletion.

may need clear argument The GC can choose to minimize a variety of cost functions such as the maximum number of keys to be updated or the average number of keys to be updated. In the case of the rooted tree based key distribution, the maximum number of keys of a member is a known fixed quantity $\log_d N$. Hence, the average number of keys to be updated is an alternative that can be minimized. In minimizing this quantity, the l_i's satisfy the Kraft inequality as noted earlier.

The optimization problem arising from the multicast key distribution on the rooted tree models of [22, 23] is to minimize the cost function $\sum_{i=1}^{n} p_i l_i$ subject to the constraint $\sum_{i=1}^{n} d^{-l_i} \leq 1$.

We note that the optimization of the average number of keys to be updated under member deletion with the length of the l_i's satisfying Kraft inequality is identical to the well known optimal codeword length selection problem [6] for prefix coding in the context of information theory. This problem is well studied and the optimal strategy is known to yield the Shannon entropy of the random variable being coded as the optimal average codeword length [6]. Since the abstract mathematical formulations are identical, we can use identical arguments to derive the optimal number of keys to be assigned to a member on the rooted tree. The derivation is standard [6], and leads to the following conclusion: *In the context of key distribution on the rooted trees, the average number of keys to be updated is the entropy of member deletion process.* We summarize this result as Theorem 2 without repeating the obvious proofs [6].

Theorem 3. Let p_i denote the probability of deleting member M_i. Let the group size be N. Let the degree of the rooted tree of key distribution be d. Then, for a key distribution satisfying at least the Kraft inequality, the optimal average number of keys (excluding the root key and the session key) to be assigned to a member is given by the $d-ary$ entropy $H_d = -\sum_{i=1}^{N} p_i \log_d p_i$ of the *member deletion* event. For a member i with probability of deletion p_i, the optimal number of keys to be assigned (excluding the root key and the session key) is given by

$$l_i{}^* = -\log_d p_i. \tag{9}$$

Including the session key and the root key, the optimal average number of keys per member is given by $H_d + 2$, and the number of keys assigned to member i with deletion probability p_i, including the SK and the root key is given by

$$l_i^* + 2 = -\log_d p_i + 2 = \log_d \frac{d^2}{p_i}. \tag{10}$$

The following features of key assignment are direct consequence of the optimization results. They are summarized in the form of lemma.

Lemma 2.

1. A member with higher probability of deletion should be given fewer keys compared to a member with lower probability of being deleted. 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 not exactly the entropy, but is bounded below and above. It is exact if the probabilities are d-adic.

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 $l_i (= -\log_d p_i) < l_j (= -\log_d p_j)$[6].

2. In order to prove this, we note that the KIDs need to be unique with minimum possible number of keys. Hence, if there is only one member 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. Computation of the redundant keys is identical to the computation in [6] (pp. 87-88) and is not repeated.

4.3 Maximum Entropy and the Key Assignment

The results reported in [4, 5, 22, 23] present a rooted tree with all members having the same number of keys. Since the optimal number of key encrypting keys for a member i with probability or deletion p_i is $(1 - \log_d p_i)$,this assignment is equivalent to treating $(1 - \log_d p_i = q)$, where q is a constant. This leads to $p_i = d^{1-q}$ for all values of i. i.e. p_i is a constant. Since $\sum_{i=1}^{N} p_i = 1$, $p_i = N^{-1}$. Hence $l_i = -\log_d p_i = \log_d N$, i.e, all the member are assigned same number of keys $(1 + \log_d N)$. Under this condition, the average number of keys assigned to a member is also $(1 + \log_d N)$. This corresponds to the trees presented in [22, 23]. Hence, the results in [22, 23] assume that all the members of the group have the equal probability of being deleted. Since the uniform distribution *maximizes* the entropy [6], and entropy is the average number of keys per member under a optimal strategy, the schemes in [22, 23] assign maximal set of keys per member on the tree. We summarize these results by the following theorem.

Theorem 4. For a d-ary rooted tree based key distribution scheme, average number of key encrypting keys per member is upper bounded by $(1 - \log_d N)$ and this value is reached when all the members have equal probabilities of being deleted.

[6]Since the members with higher probability of being deleted are assigned lesser keys in this strategy, the GC can adaptively react to any possible coordinated attack effort by members to increase frequency of rekeying by simply forcing deletion by leaving and joining at very high rates. The GC will be able to assign less number of keys to members with higher probabilities of being deleted. In the traditional models [22, 23], there is no explicit mechanism to include this knowledge into the key distribution.

Proof: We showed that the average number of key encrypting keys per member is $(1 + H_d)$ where $H_d = -\sum_{i=1}^{i=N} p_i \log_d p_i$. The entropy H_d of member deletion can be shown to be maximized [6] when the probabilities of the event is uniform.

4.4 Upper Bounds on the Integer Values of keys

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

$$2 - \log_d p_i. \tag{11}$$

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

$$H_d + 2 \leq \hat{l}^* + 2 < H_d + 3. \tag{12}$$

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

$$l_i^* = \lceil -\log_d p_i \rceil. \tag{13}$$

For this choice of l_i^*, it can be shown [6], 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 deletion event.*

4.5 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 deletion probability of member i is p_i and the used probability of deletion for member i is q_i. Hence, the optimal number of keys to be assigned to that member is given by

$$l_i^* = 2 + \lceil -\log_d q_i \rceil \tag{14}$$

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 $l_i = 2 - \log_d q_i$ satisfies the following bounds

$$H_d(p) + D(p||q) + 2 \leq L + 2 < H_d(p) + D(p||q) + 3, \tag{15}$$

where $L = \sum_{i=1}^{i=N} p_i l_i$, and $D(p||q)$ is the information divergence [6].

Proof: Follows with minor modifications of derivations in [6].

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 deletion event is also related to the sustainable key length of the secure multicast group, as shown next.

5 Impact of Member Deletion on the Key Length

For secure communication, the desired key length need to satisfy a minimum threshold depending on the application. In the case of secure multicast key distribution, the admissible key length is tied to the bit generation rate of the group controller and the rate of member deletion as shown below. In doing so, we make the assumption that when a member is deleted, the keys are generated and updated before secure communication continues.

5.1 Bounds on Average Key Length

Since the keys need to be updated in order to continue with the communication without interruptions, if we denote the bit length of a key by λ and the hardware bit generation rate by B, in [22, 23], the key length has to satisfy

$$B \geq \lambda(2 + \log_d N). \tag{16}$$

Hence, the maximum possible key length is given by $\lambda = \frac{B}{2+\log_d N}$. In the probability based modeling, the sustainable key length however can be smaller than $\frac{B}{(2+\log_d N)}$. The following theorem establishes the value of the possible sustainable length of the key for a rooted tree based key distribution scheme.

Theorem 6. For $d - ary$ rooted tree based key distribution scheme [22, 23], if each key is of length λ digits, and the hardware digit generation rate is B, then the maximum sustainable key length λ^* is given by $min\{\frac{B}{(2-\log_d p_i)}, \frac{B}{(2+\log_d N)}\}$.

Proof: We showed above that the bound on the key length when all the members have equal probability of being deleted is $\lambda^* = \frac{B}{(2+log_d N)}$. We now compute the bound for the models using the probabilities. The number of keys to be generated in deleting member M_i with probability of deletion p_i is $(2 - \log_d p_i)$. Hence, the hardware should be able to generate $L(2 - \log_d p_i)$ digits of *suitable quality*[7] in unit of time to let the session continue without interruptions. Hence, the hardware digit generation rate B must satisfy

$$\begin{aligned} B &\geq \lambda(2 - \log_d p_i) \qquad (17)\\ \lambda &\leq \frac{B}{(2 - \log_d p_i)}. \end{aligned}$$

Since B is a fixed quantity, choice of λ is controlled by the choice of p_i. Our objective is to choose the maximum possible value of key length that can be sustained under a member deletion. The lower bound is obtained on the maximum possible λ as $\lambda = \overset{min}{p_i} \frac{B}{(2+\log_d N)}$. We note

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

Therefore, the minimum value of $\frac{B}{(2-\log_d p_i)}$ is attained when $p_i = p_{min}$. Hence, for the probability based models of the rooted tree, the sustainable key length is given by $\lambda \leq \frac{B}{(2-\log_d p_{min})}$. Setting the equality yields the maximum possible length $\lambda^* = \frac{B}{(2-\log_d p_{min})}$.

Hence, the maximum possible sustainable key length under a member deletion is given by $min\{\frac{B}{(2-\log_d p_{min})}, \frac{B}{(2+\log_d N)}\}$ for the rooted tree based key distribution that does not impose any relationship among the keys.

6 A Rooted Key Distribution with User Collusion

The optimal rooted tree derived was shown to be identical to the optimal codeword tree in information theory. In the case of codeword selection problem, Huffman codes are known to be optimal [6]. From the similarities of both the trees, a possible question to ask is where the codeword length selection problem differs from cryptographic key distribution on rooted trees.

As noted earlier, one of the differences is that the keys distribution needs to consider the uniqueness of the assigned keys and the user collusion (or be cover free as noted in sections 3.1 and 3.6) whereas the codeword generation needs to consider the uniqueness of the assigned codes. We also showed that the condition for any KID to be cover free is that at least that the leafs keys are all distinct. Under this condition, the key storage requirement of the group controller is linear in N.

We now describe a rooted key based key distribution scheme [5] that satisfy the maximum entropy bound ($\log_d N$)for the number of keys assigned to a member while attempting to directly map the UID to KIDs.

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

We use this scheme [5] to demonstrate the fact that while a key distribution scheme may attain optimality, it may not be cover free or collusion free.

Let $X_{n-1}X_{n-2}\cdots X_0$ denote a binary UID of the members. Each of the bit X_i is either a zero or a one. There are 2^n possible different UIDs for this sequence. In [5], the following direct mapping between the KIDs and the UIDs was proposed. We illustrate the mapping by the example in [5]. In [4], when $N = 8$, $\log_2 8 = 3$ bits are needed to uniquely index *all* eight members. The authors then proceeded to note that since each bit takes two values, it can be mapped to a pair of distinct keys. The UID string $X_2X_1X_0$ is mapped to the KID string $K_2K_1K_0$. The table below reproduces the mapping between the ID bit # and the key mapping for the case in [4] for $N = 8$ where, the key pair (K_{i0}, K_{i1}) represents the two possible values of the *ith* bit of the member index.

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

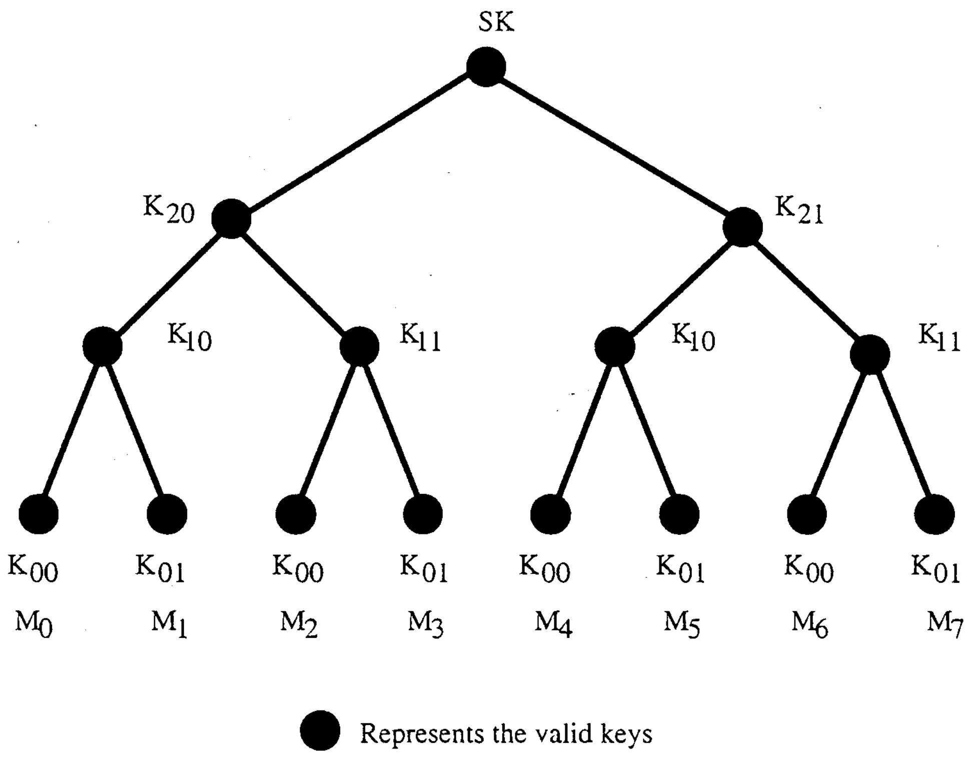

Figure 3: The Key Distribution in [4, 5]

Figure 3 presents the corresponding binary tree for the key assignment. This rooted tree has the special structure that at any given depth from the root, two new keys are used. At depth h from the root, the two new keys $K_{(\log_d N-h)0}$ and $K_{(\log_d N-h)1}$ are duplicated h times. The total number of keys to be stored in this scheme is $2\log_2 N$. For a d-ary rooted tree, the total number of keys to be stored in this scheme is $d\log_d N$.

Although the total number of keys is $d \log_d N$, deletion of more than one member may bring this key distribution scheme to halt. In the case of Figure 3, This happens if the members M_0 and M_7 need to be deleted. The KID of member M_0 is $K_{20}K_{10}K_{00}$ and the KID of member M_7 is $K_{21}K_{11}K_{01}$. The union of the keys forming these two KIDs cover the entire the keys used for key distribution on the tree. The corresponding keys to be deleted are shown in Figure 4. Hence, if these two members need to be simultaneously deleted, the group controller is left with *no key* to securely communicate with the rest of the valid members. The compromise recovery for this case requires that the entire group re-key itself by contacting one member at a time.

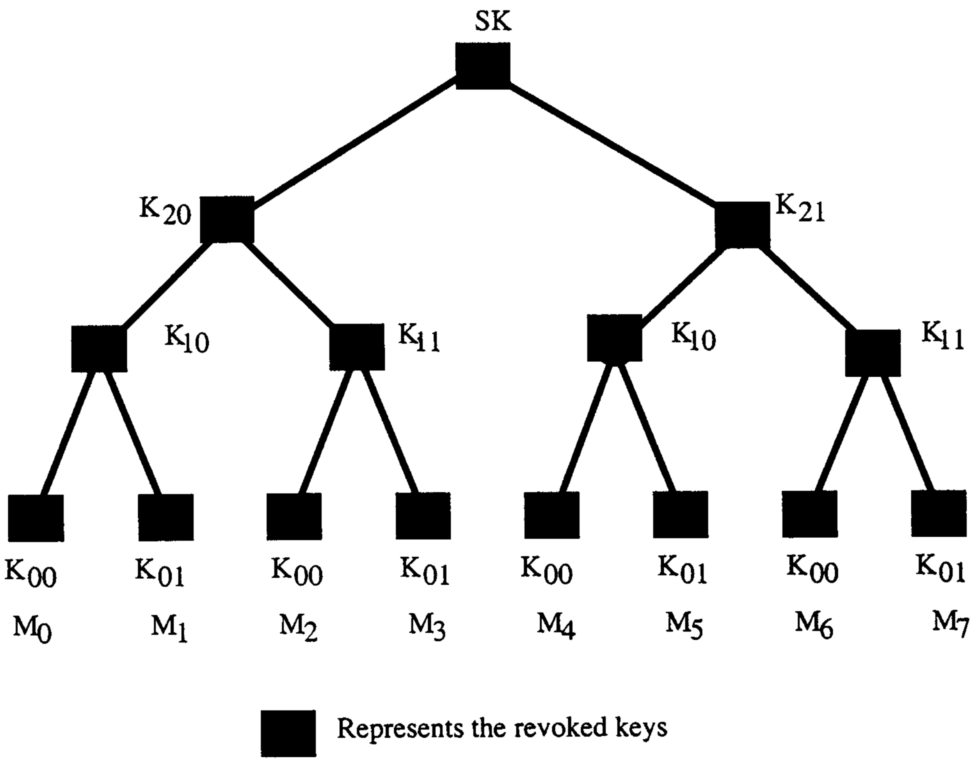

Figure 4: Deletion of Members M_0, M_7 in [4, 5].

Apart from member deletion, the key assignments in [4, 5] and their variations also allow the members to collaborate and break the system. We now interpret the user collusion on the rooted tree in [4, 5].

6.1 Interpretation based on Minimal number of Key Requirements

We noted earlier that the sufficient condition for cover freeness requires that at least the N leaf nodes are assigned distinct keys. Hence, for a binary tree, the model in [5] will be cover free if $2 \log_2 N \geq N$. This condition is satisfied with equality if $N = 2$ or $N = 4$. These two cases are not the target group sizes for the tree based key distribution schemes.

6.2 Interpretation of Collusion Problem Based on Complementary variables

The second interpretation is based on the notion of sets, and is also discussed under the category of complementary variables in [7, 22]. In complementary variable approach, each member of the group is identified by a unique key. This unique key is distributed to everyone in the group excluding the member identified by that key. When a member is deleted, the index of the member is broadcasted. For the next session, all the valid members set the key corresponding to the deleted member as the new session key. Under this model, 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. If we consider any two members, the union of the keys stored by them will cover the keys stored by entire group. Hence, this key assignment does not scale beyond 2 members.

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 $(k_j, \hat{k}_j)$. In a $d-ary$ tree with key distribution in [5], 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. We summarize this by the following condition:

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 $\boxed{b_1 + b_2 \cdots b_k \equiv 0 \bmod \frac{d(d-1)}{2}}$.

7 Conclusions and Discussions

This paper showed that the recently proposed [22, 23] rooted tree based secure multicast key distribution schemes can be systematically studied using basic information theoretic concepts. By using the member deletion event as the basis of our formulation, we showed that the optimal number of average keys assigned to a member is related to the *entropy* of the member deletion event. We derived the necessary and sufficient condition for the key assignment to be collusion free and in general "cover free". In particular, we showed that the cover free condition on the rooted trees require that the leaf node key be all distinct when there is no additional relationship among keys. Under this condition, the storage requirements of the group controller is linear in group membership N. We then proved that the currently available known rooted-tree based strategies [22, 23] and their variations [4, 5] 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 deletion probabilities. We also derived a relationship between the average key length, probability of member deletion event, and the hardware digit generation rate.

Acknowledgments

We are grateful to anonymous reviewers for useful suggestions that improved the presentation. Many useful discussions with Dr. Eric Harder helped during the course of this work. 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. (Also available from *http : //www.wisdom.weizmann.ac.il/home/bennyp/public$_h$tml/index.html*)

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

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

[4] 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.

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

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

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

[8] E. Gafni, J. Staddon, Y. L. Yin, "Efficient Methods for Integrating Traceability and Broadcast Encryption", *Advances in Cryptology- CRYPTO'99*, LNCS. vol 1666, pp. 372- 387, Springer-Verlag, Berlin, Germany, 1999.

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

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

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

[12] M. Luby,

[13] 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.

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

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

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

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

[18] R. Poovendran, and J. S. Baras, "An Information Theoretic Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes", *Advances in Cryptology- CRYPTO'99*, LNCS. vol 1666, pp. 624-638, August 1999, Santa Barbara, USA.

[19] 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.

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

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

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

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

Reviewer 1

Report on the paper
An Information Theoretic Approach for Design and Analysis of
Rooted-Tree Based Multicast Key Management Schemes
by R. Poovendran and J. S. Baras

Recommendation:

The paper contains interesting results on multicast key management schemes based on rooted tree. Using the entropy of member revocation event the authors present methods for design and analysis of rooted tree based schemes. Several recently proposed schemes are analysed using this information theoretic approach. I recommend publication in the *IEEE Transactions on Information Theory* subject to the following revisions.

Suggestions and comments:

- page 4 line 9: replace " show " by " shows "
- page 4 line 11: replace " In We " by " In 7 we "
- page 4 line -11: replace " 16 " by " 8 "
- page 8 line 6: replace " L_i " by " $L_i!$ "
- page 8 line 7: replace " a equivalence " by " an equivalence "
- page 9 line -7: remove the comma (,) after *event*
- page 10 line -10: replace " indepent " by " independent "
- page 11 line 2 and 11: replace " occurance " by " occurence "
- page 11 line -6: insert " *to the entropy of member revocation event* " right after " *is identical* "
- page 12 line 5 and 11: change " per members " to " per member "
- page 12 line 10: change " satify " to " satisfy "
- page 12 line 20: Is it SK instead of TEK?

- page 13 line 3: change " $l_i(= -\log_d p_i > l_j(= -\log_d p_j)$ " to " $l_i(= -\log_d p_i < l_j(= -\log_d p_j)$ "
- page 13 line -5: change " $\log_d pi$ " to " $\log_d p_i$ " in equation (9)
- page 14 line 5: change " indicates " to " indicate "
- page 14 line 12: replace " or " by " of "
- page 15 line 15: replace " yiled " by " yield "
- page 16 line 16: replace " $< H_d(q)$" by "$< H_d(p)$ " in equation (17)
- page 18 line -3: change " $\{K_2, K_4, K_5, K_6\}$ " to " $\{K_1, K_2, K_3\}$ "
- page 19 line 1: replace " i and j " by " i and k "
- page 19 line 4 and 5: change " $\in$ " to " $\subseteq$ "
- page 19 line 10: change " larger group). " to " larger group.) "
- page 22 line 9: change " the the number " to " the number "
- Only the following 12 items: 4, 5, 6, 9, 10, 20, 21, 22, 23, 24, 26, 31 are quoted in the paper. The other 19 items appear nowhere, so they should be removed from the list of references.

Reviewer ②

This paper studies rooted-tree based multicast schemes. The goal of the paper is to show how information theory can be used to clarify recent work in this area, and to obtain optimality conditions. To this end, the authors consider multicast schemes with an arbitrary probability distribution over the member revocations (i.e. distributions $p = (p_1,\ldots,p_n)$, where p_i denotes the probability that member i is revoked). Previous work assumes that each user is revoked with equal probability ($p_i = 1/n$ for every i). The authors show that the entropy determined by the probability distribution is equal to the optimal average number of keys per user. They show that while a scheme may be optimal with respect to the uniform probability distribution it may not be secure against collusions (i.e. the scheme is only 1-resilient). The authors also make observations about the effect of the hardware digit generation rate and incorrect entropy values on key length.

This paper makes some useful points about tree-based multicast schemes. The theorems follow directly from well-known results in information theory. In addition, allowing for arbitrary revocation probability distributions is a natural extension of previous work.

The scope of this paper is a bit narrow in two respects. First, the authors make the point that unless special care is taken, a tree-based scheme may be vulnerable to coalitions of as few as two users. The concept that seems to be at the heart of the problem is "cover-freeness". Informally, if two users can cover another user's key set then the scheme is not 2-cover-free, and it will be impossible to reach the third member securely if the two members are revoked. The concept of cover-freeness has been used to construct broadcast encryption schemes quite a bit recently (e.g. the Crypto `99 papers of Kumar-Rajagopalan-Sahai and Gafni-Staddon-Yin). This concept should be discussed in this paper.

fine here.
need to spend

Secondly, the main goal of the paper appears to be the demonstration that bounds on the average number of keys per user follow naturally when the problem is looked at in an information theoretic context. In a multicast scheme that is tight with the bounds proven in this paper, the length of the transmission necessary to reach the set of targeted users may grow to be quite large. This trade-off between transmission length and the number of keys per user has been extensively studied (e.g. Canetti-Malkin-Nissim, Eurocrypt `99) and it is fairly well understood. I think it's important that this paper make it clear that they are not striving to bound the transmission length, when calculating optimal values for the number of keys per user (e.g. through expanding the discussion in section 4.5).

with
exponentia
low probabili

In conclusion, I recommend accepting the paper largely because of the important and natural connections the authors draw between multicast schemes and information theory. I also suggest that the authors revise the papers in light of the above comments and the editorial feedback below.

Editorial comments follow:

1. I think there's an error in equation (2) on page 7. (Still $O(N)$, though.) ✓
2. Assertions are sometimes made without supporting references. For example, see the first paragraph of section 5.
3. It's a bit hard to find things in the bibliography because the entries are not alphabetized by author name.
4. The paper could use some rereading to catch typos and awkward phrasing.

Reviewer # 3

Report on "An information Theoretic Approach..."

by Poovendran and Baras

The paper presents a different exposition and some improvements to the "virtual tree" scheme of Wong et.al. (Co-discovered by Wallner et.al) for efficient membeship revocation in secure multicast groups.
In particular the paper adresses the following issues:

1. The need to keep the tree balanced.
2. The requirement that the numbers of keys held by the users should satisfy the Kraft Inequality.
3. Assigning probabilities to the events that users leave the group, and constructing the tree according to those probabilities.
4. A relation between the size of the keys and the size of the group.
5. Drawbacks in other membership revocation schemes.

I must say that I found the paper quite un-interesting, and cannot recomment acceptance. In particular:

-Points 1 and 2 are quite obvious. Also, adding users in a balanced way is quite straighforward.

-It is usually not the case that
users choose whether to leave at each step using some a-priorily fixed probability. Also, users do not make a new, independent choice whether to leave at each step. Rather, users usually have some profile of membership length (e.g., fixed or expected duration, or some specified real time event).
Consequently, assigning probabilities (that are fixed a-priori) to the event of user removal does not seem to model reality.

-I could not follow the claimed bound on key length as a function of the group size, nor could I understand what was the claim.

-The described drawback in the scheme of [22,24] was known. It is even mentioned in 24. Also the extension in section 6.3.3, and its breaking, are not surprising.

University of
Waterloo

519 888 - 4567
5590

Department of Combinatorics and Optimization

Faculty of Mathematics

University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada
N2L 3G1

519·888·4567, ext. 3482
Fax 519·725·5441
combopt@math.uwaterloo.ca
http://math.uwaterloo.ca/CandO_Dept/homepage.html

February 21, 2000

Prof. Radha Poovendran
Room #2243, AVW Building
Institute for Systems Research
University of Maryland
College Park, MD
20742 USA

Dear Prof. Poovendran:

I have now received three reports on your paper

"An Information Theorectic Approach for Design and Anaysis ..."
(file number CLN 99-211).

These reports are enclosed. Please send me three copies of a revised version of the paper that addresses the referees' comments. Then I will send the paper out for re-review.

Yours sincerely,

[illegible]

Douglas Stinson
Associate Editor for Complexity and Cryptography
IEEE Transactions on Information Theory

Report on the paper
An Information Theoretic Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes
by R. Poovendran and J. S. Baras

Recommendation:

The paper contains interesting results on multicast key management schemes based on rooted tree. Using the entropy of member revocation event the authors present methods for design and analysis of rooted tree based schemes. Several recently proposed schemes are analysed using this information theoretic approach. I recommend publication in the *IEEE Transactions on Information Theory* subject to the following revisions.

Suggestions and comments:

- page 4 line 9: replace " show " by " shows "
- page 4 line 11: replace " In We " by " In 7 we "
- page 4 line -11: replace " 16 " by " 8 "
- page 8 line 6: replace " L_i " by " $L_i!$ "
- page 8 line 7: replace " a equivalence " by " an equivalence "
- page 9 line -7: remove the comma (,) after *event*
- page 10 line -10: replace " indepent " by " independent "
- page 11 line 2 and 11: replace " occurance " by " occurence "
- page 11 line -6: insert " *to the entropy of member revocation event* " right after " *is identical* "
- page 12 line 5 and 11: change " per members " to " per member "
- page 12 line 10: change " satify " to " satisfy "
- page 12 line 20: Is it SK instead of TEK?

- page 13 line 3: change " $l_i(=-\log_d p_i > l_j(=-\log_d p_j)$ " to " $l_i(=-\log_d p_i < l_j(=-\log_d p_j)$ "
- page 13 line -5: change " $\log_d pi$ " to " $\log_d p_i$ " in equation (9)
- page 14 line 5: change " indicates " to " indicate "
- page 14 line 12: replace " or " by " of "
- page 15 line 15: replace " yiled " by " yield "
- page 16 line 16: replace " $< H_d(q)$" by "$< H_d(p)$ " in equation (17)
- page 18 line -3: change " $\{K_2, K_4, K_5, K_6\}$ " to " $\{K_1, K_2, K_3\}$ "
- page 19 line 1: replace " i and j " by " i and k "
- page 19 line 4 and 5: change " $\in$ " to " $\subseteq$ "
- page 19 line 10: change " larger group). " to " larger group.) "
- page 22 line 9: change " the the number " to " the number "
- Only the following 12 items: 4, 5, 6, 9, 10, 20, 21, 22, 23, 24, 26, 31 are quoted in the paper. The other 19 items appear nowhere, so they should be removed from the list of references.

This paper studies rooted-tree based multicast schemes. The goal of the paper is to show how information theory can be used to clarify recent work in this area, and to obtain optimality conditions. To this end, the authors consider multicast schemes with an arbitrary probability distribution over the member revocations (i.e. distributions $p = (p_1,\ldots,p_n)$, where p_i denotes the probability that member i is revoked). Previous work assumes that each user is revoked with equal probability ($p_i = 1/n$ for every i). The authors show that the entropy determined by the probability distribution is equal to the optimal average number of keys per user. They show that while a scheme may be optimal with respect to the uniform probability distribution it may not be secure against collusions (i.e. the scheme is only 1-resilient). The authors also make observations about the effect of the hardware digit generation rate and incorrect entropy values on key length.

This paper makes some useful points about tree-based multicast schemes. The theorems follow directly from well-known results in information theory. In addition, allowing for arbitrary revocation probability distributions is a natural extension of previous work.

The scope of this paper is a bit narrow in two respects. First, the authors make the point that unless special care is taken, a tree-based scheme may be vulnerable to coalitions of as few as two users. The concept that seems to be at the heart of the problem is "cover-freeness". Informally, if two users can cover another user's key set then the scheme is not 2-cover-free, and it will be impossible to reach the third member securely if the two members are revoked. The concept of cover-freeness has been used to construct broadcast encryption schemes quite a bit recently (e.g. the Crypto `99 papers of Kumar-Rajagopalan-Sahai and Gafni-Staddon-Yin). This concept should be discussed in this paper.

Secondly, the main goal of the paper appears to be the demonstration that bounds on the average number of keys per user follow naturally when the problem is looked at in an information theoretic context. In a multicast scheme that is tight with the bounds proven in this paper, the length of the transmission necessary to reach the set of targeted users may grow to be quite large. This trade-off between transmission length and the number of keys per user has been extensively studied (e.g. Canetti-Malkin-Nissim, Eurocrypt `99) and it is fairly well understood. I think it's important that this paper make it clear that they are not striving to bound the transmission length, when calculating optimal values for the number of keys per user (e.g. through expanding the discussion in section 4.5).

In conclusion, I recommend accepting the paper largely because of the important and natural connections the authors draw between multicast schemes and information theory. I also suggest that the authors revise the papers in light of the above comments and the editorial feedback below.

Editorial comments follow:

1. I think there's an error in equation (2) on page 7. (Still $O(N)$, though.)
2. Assertions are sometimes made without supporting references. For example, see the first paragraph of section 5.
3. It's a bit hard to find things in the bibliography because the entries are not alphabetized by author name.
4. The paper could use some rereading to catch typos and awkward phrasing.

Report on "An information Theoretic Approach..."

by Poovendran and Baras

The paper presents a different exposition and some improvements to the "virtual tree" scheme of Wong et.al. (Co-discovered by Wallner et.al) for efficient membeship revocation in secure multicast groups.
In particular the paper adresses the following issues:

1. The need to keep the tree balanced.
2. The requirement that the numbers of keys held by the users should satisfy the Kraft Inequality.
3. Assigning probabilities to the events that users leave the group, and constructing the tree according to those probabilities.
4. A relation between the size of the keys and the size of the group.
5. Drawbacks in other membership revocation schemes.

I must say that I found the paper quite un-interesting, and cannot recomment acceptance. In particular:

-Points 1 and 2 are quite obvious. Also, adding users in a balanced way is quite straighforward.

-It is usually not the case that
users choose whether to leave at each step using some a-priorily fixed probability. Also, users do not make a new, independent choice whether to leave at each step. Rather, users usually have some profile of membership length (e.g., fixed or expected duration, or some specified real time event).
Consequently, assigning probabilities (that are fixed a-priori) to the event of user removal does not seem to model reality.

-I could not follow the claimed bound on key length as a function of the group size, nor could I understand what was the claim.

-The described drawback in the scheme of [22,24] was known. It is even mentioned in 24. Also the extension in section 6.3.3, and its breaking, are not surprising.