Combinatorial Optimization of Group Key Management*

M. Eltoweissy
Dept. of Computer Sci
Virginia Tech
Falls Church, VA 22043
toweissy@nvc.cs.vt.edu

M.H. Heydari
Dept. of Computer Sci.
James Madison Univ.
Harrisonburg, VA 22807
heydarmh@jmu.edu

L. Morales
Dept. of Computer Sci.
Texas A&M Univ.
Commerce, TX 75429
Linda_Morales@tamu-commerce.edu

H. Sudboorough
Dept of Computer Sci.
Univ. of Texas - Dallas
Dallas, TX 75083
hal@utdallas.edu

Abstract: *Given the growing number of group applications in many existing and evolving domains much recent attention has been focused on secure multicasting over the Internet. When such systems are required to manage large groups that undergo frequent fluctuations in group membership, the need for efficient encryption key management becomes critical. This paper presents a new key management framework based on a combinatorial formulation of the group multicast key management problem that is applicable to the general problem of managing keys for any type of trusted group communication, regardless of the underlying transmission method between group participants. Specifically, we describe Exclusion Basis Systems and show exactly when they exist. In addition, the framework separates key management from encrypted message transmission resulting in a more efficient implementation of key management.*

Keywords: Group Communications, Combinatorial Optimization, Key Management, Secure Communications

I. Introduction

Group communications is a topic of considerable interest today due to the growth of the Internet and the widespread availability of high bandwidth connections, the emergence of Grid

* This research is supported in part by grant number SE 2001-01 from the Commonwealth Technology Research Fund

Computing, and wireless networks. The need for secure group communication techniques has also been recognized. Many applications, such as pay-per-view services and secure teleconferencing, require group access control and message privacy to be viable. Secure group communication techniques must gracefully tolerate rapid fluctuations in group membership. Members are generally allowed to join and leave groups at will, and access to underlying transmission infrastructure, such as multicast transmission must be granted and revoked with minimal system overhead.

One way to implement secure multicast transmissions is through the use of message encryption. This requires that each authorized member of a secure multicast group has knowledge of a session encryption key shared by the entire group. The message source uses this session key for encrypting data packets before sending them to the group. When a member is evicted from the group, the session key must be changed in order to maintain message privacy. All remaining group members receive the new session key by secure transmission, which is typically accomplished by multicasting an encrypted message containing the new key to the group. The message must be indecipherable to the evicted member, which means that each remaining group member must have one or more administrative encryption keys known also to the key server. (We assume the existence of a key server function whose purpose is to manage all session and administrative keys used by the multicast group. The key server may or may not be implemented within a message source.) The administrative keys are used only for re-keying operations that take place when group membership changes. In this paper, we will use the word “key” to refer collectively to session keys and administrative keys. Each group member is also assumed to possess a personal key, known only to the member and to the key server.

For large groups, the number of keys may become quite large, making efficient key management a non-trivial problem. Some end-user devices, such as mobile phones and PDAs, are

memory constrained, so it is desirable to keep the number of keys stored by each member to a minimum. The key server must store the keys for the entire group, so the total number of keys must also be minimized. Furthermore, group membership may change frequently, so the number of re-key messages needed to re-establish security when a member is evicted must be minimized as well. These concerns give rise to the need for efficient key management techniques that minimize both the number of keys and the number of re-key messages.

Several key management techniques have been proposed. See for example [1-9]. These techniques are generally based on the use of specific logical data structures, such as stars and trees, for storing keys. In general, for a tree of degree k, each member in a group with n participants must remember $\log_k n$ administrative keys, and the server must send $(k-1)*\log_k n$ re-key messages. It can be shown that among all k-ary trees, binary trees require the smallest number of re-key messages.

There are numerous situations that require the restricted sharing of information in groups whose membership fluctuates. Examples include various military applications, diplomatic communications, e-learning, air traffic control, etc. Further, end-users need not be human. Secure network management of computer or telecommunication networks could be facilitated by the use of secure group communications. Encrypted network management signaling messages could be multicast to trusted network entities, and key management techniques would be needed if the trusted network is large and experiences fluctuations in membership. Regardless of the underlying communication network (e.g., human courier or Internet) the same concerns arise when the network gets big. The number of keys per user and the number of re-key messages needed to re-establish security, should be as close to optimal as possible.

In this paper, we present *Exclusion Basis Systems* (EBS), a combinatorial formulation of the group key management problem that produces optimal results with respect to the parameters n, k and

m, where n is the size of the group, k is the number of keys stored by each member, and m is the number of re-key messages. We develop a general technique for determining optimal values of k and m as a function n, and describe the trade-off between k and m. Our formulation contains stars, k-ary trees and all such structures as special cases. In addition, we describe algorithms for admitting and evicting group members, and demonstrate the scalability of EBSs.

The study of EBSs applies combinatorial optimization techniques to the key management problem. That is, the purpose is to identify systems that minimize the number, m, of multicast messages needed for re-keying operations, while simultaneously minimizing the number, k, of keys per user. The solutions to our combinatorial problem offer a broad spectrum of optimal solutions, indicating a tradeoff between the numbers m and k. Our main result also offers a general lower bound, indicating for which values of m and k a solution for n users exists.

An Exclusion Basis System is an abstraction of the problem of constructing a logical structure for key management. For example, in the standard binary tree structure, each node of the tree represents a key. The root of the tree represents the session or group key that can be used to encrypt messages to be multicast to all members. The leaves (one for each member) represent the personal keys of the members. The internal nodes (other than the root) are administrative keys used by the system to facilitate economical re-key operations when a user is evicted. In the binary tree, a user knows all keys corresponding to nodes from his personal key position (at one of the leaves) to the root of the tree. Hence, if the binary tree is a complete or balanced tree, each user knows $O(\log_2 n)$ keys, equal to the height of the tree [1,2,7,9-13].

Viewing the logical structure as a collection of subsets of the group members allows one to consider more general scenarios that are not always easily identified as a common data structure or graph. For example, to view the binary tree structure as a collection of subsets one can literally view

each key of the tree as a set, namely the set of users who know that key. Consider the complete binary tree of height three shown in Figure 1.

The key corresponding to node 00 is known by both users 0 and 1 and can be represented by the set {0,1}. The key corresponding to node 0 is known by users 0, 1, 2, and 3 and can be represented by the set {0,1,2,3}. So, the entire set of keys represented by the tree can be viewed as a collection of subsets of {0,1,2,...,7}. If one excludes for this consideration the group key at the root of the tree, then the number of keys known by each user is $\log_2 n$, the height of the tree, which is 3 in the example. Also, if user 0 is evicted, then three messages are sufficient to communicate the new keys corresponding to nodes λ, 0, and 00, i.e., the keys known by the evicted user and which must be changed. The three messages are: (1) a message containing the new keys for λ, 0, and 00 encrypted by user 1's personal key, (2) a message containing the new keys for λ and 0 encrypted by the key at position 01 (known to users 2 and 3), and (3) a message containing the new group key for λ encrypted by the key at position 1 (known to users 4, 5, 6, and 7). Again, viewing this logical structure as a collection of subsets, an equivalent and more succinct statement can be made: the union of the three sets {1}, {2,3}, {4,5,6,7} yields [0,7] - {0}. That is, more generally, any user can be excluded by the union of three subsets in the collection.

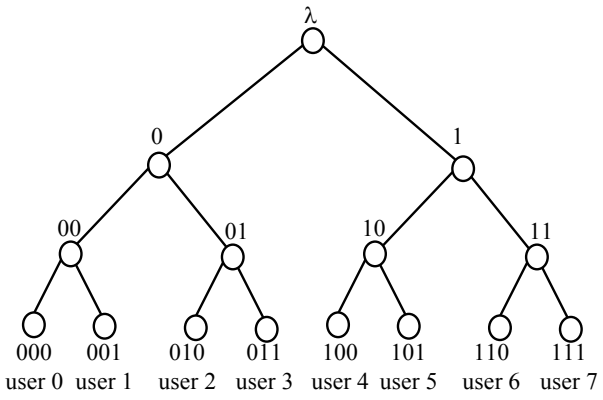

Figure 1. Binary tree-based key management structure

The remainder of this paper is organized as follows. Section 2 describes Exclusion Basis Systems with the goal of understanding how to construct the best possible system. Section 3 discusses scalability issues in EBSs. Section 4 presents solutions for preventing collusions in EBSs. The paper concludes with an outline of future work in Section 5.

II. Exclusion Basis Systems

In the following formulation, an Exclusion Basis System (EBS) is defined as a collection Γ of subsets of the set of users. Each subset corresponds to a key and the elements of a subset $A \in \Gamma$ are the users that have that key. We assume that, in addition to the administrative keys corresponding to subsets in Γ, a key server also has a session key known to all users, and for each user, an personal key known only to that user (and the key server). The session key is clearly needed for multicasting encrypted data messages to all users. Personal keys are used for user authentication and for unicasting initialization information when an individual user joins a multicast group.

Definition 1. Let n, k and m be positive integers, such that $1<k,m<n$. An *Exclusion Basis System of dimension (n,k,m),* denoted by *EBS(n,k,m),* is a collection Γ of subsets of $[1,n] = \{1,2,...,n\}$ such that for every integer $t \in [1,n]$ the following two properties hold:

(a) t is in at most k subsets of Γ, and

(b) there are exactly m subsets, say $A_1, A_2, ..., A_m$, in Γ such that $\bigcup_{i=1}^{m} A_i$ is $[1,n] - \{t\}$. (That is, each element t is excluded by a union of exactly m subsets in Γ.)

An example of an EBS(8,3,2) is the collection of subsets $\Gamma=\{A_1=\{5,6,7,8\}, A_2=\{2,3,4,8\}, A_3=\{1,3,4,6,7\}, A_4=\{1,2,4,5,7\}, A_5=\{1,2,3,5,6,8\}\}$. One can easily verify that each integer $t\in[1,8]$ is in exactly 3 subsets of Γ, and each integer t is excluded by a union of exactly 2 subsets in Γ, as illustrated below:

$[1,8] - \{1\} = A_1 \cup A_2$,
$[1,8] - \{2\} = A_1 \cup A_3$,
$[1,8] - \{3\} = A_1 \cup A_4$,
$[1,8] - \{4\} = A_1 \cup A_5$,
$[1,8] - \{5\} = A_2 \cup A_3$,
$[1,8] - \{6\} = A_2 \cup A_4$,
$[1,8] - \{7\} = A_2 \cup A_5$, and
$[1,8] - \{8\} = A_3 \cup A_4$.

An Exclusion Basis System Γ of dimension (n,k,m) represents a situation in a secure group where there are n users numbered 1 through n, and where a key server holds a distinct key for each set in Γ. In this paper, we will use the terms "key" and "subset" interchangeably. If the subset A_i is in Γ, then the key A_i is known by each of the users whose number appears in the subset A_i. (For example, in the EBS(8,3,2) instance above, key A_1 is known by users 5, 6, 7, and 8 and by no others.) Furthermore, for each $t\in[1,n]$ there are m sets in Γ whose union is $[1,n] - \{t\}$. From this it follows, as we shall see, that the key server can eject any user t, re-key, and let all remaining users know the replacement keys for the k keys they are entitled to know including the session key, by multicasting m messages encrypted by the keys corresponding to the m sets whose union is $[1,n] - \{t\}$.

To illustrate, consider the case when user 1 is ejected in the example EBS(8,3,2) above. (This system is fully symmetric, so evicting other users is handled in an analogous manner.) User 1 knows keys A_3, A_4, and A_5, so these keys need to be changed and the new values sent out to authorized users. Observe that $[1,8]-\{1\} = A_1 \cup A_2$, so we show that two messages, encrypted by keys A_1 and

A_2, respectively, are sufficient to distribute the new keys to authorized users. Let the first message be one encrypted with key A_1, and which contains four subparts. The four parts of the message are:

(1) a new session key, S',

(2) replacement key for A_3 encrypted by the former A_3 key,

(3) replacement key for A_4 encrypted by the former A_4 key,

(4) replacement key for A_5 encrypted by the former A_5 key,

In other words, the first message is represented as:

$$A_1(S',A_3(A'_3,), A_4(A'_4,) A_5(A'_5,)),$$

where $A_i(x)$ denotes encryption of x by key A_i, and A'_i represents the replacement key for the old key A_i. In this notation, the second message would be:

$$A_2(S',A_3(A'_3,), A_4(A'_4,) A_5(A'_5,)),$$

It is easily verified that these two messages allow every remaining user, after user 1's departure, to learn exactly the set of new keys to which he is entitled. Furthermore, user 1 cannot decipher the re-key messages since user 1 does not possess keys A_1 and A_2. At the conclusion of the re-key operation, user 1 has been effectively excluded from the secure group. The general case for an arbitrary EBS(n,k,m) is done in an analogous fashion. That is, for A_1, A_2, ..., A_m, in Γ such that $\underset{i=1}{\overset{m}{Y}} A_i$ is [1,n] - {t}, the key server can evict user t by re-keying, and sending out m messages, such that the i^{th} message is encrypted with key A_i and contains the new session key and new administrative keys encrypted by their predecessors to limit their decipherability to appropriate users only.

If desired, the use of double encryption when rekeying, as described above, can be avoided. For example, as indicated in [8], a new administrative key can be made to be computable by a one-way

trapdoor function from the new session key and the old administrative key. That is, $A'_i = f(S', A_i)$, for some one-way function f. Then, the new administrative key A'_i can be computed by any user that knows the new session key S' and the previous administrative key A_i. Since each user would generate his own new administrative keys, the server would not need to include them in the message, hence no double encryption would be needed.

Section 2.1. Constructing Exclusion Basis Systems.

In order to describe the construction of EBS(n,k,m) for feasible n, k and m, we first present a canonical enumeration of all possible ways of forming subsets of k objects from a set of k+m objects. We do this because our construction is based on such an enumeration. There are several algorithms for producing such a sequential enumeration. We choose to describe an enumeration where each element of the sequence is a bit string of length k+m, where a 1 in the i^{th} position of a string means that object i is included in that subset, for all i ($1 \le i \le k+m$). Note that every bit string in this enumeration will have k 1's. We describe our so-called canonical enumeration for the $\binom{k+m}{k}$ subsets by induction on k+m.

For the basis step, when k+m=2, there are three sequences made up of bit strings of length 2. The sequence for k=0 (*i.e.*, $\binom{2}{0}$ has one bit string, namely 00. The sequence for k=1 (*i.e.,* $\binom{2}{1}$ has two bit strings, namely 01 and 10. Finally, the sequence for k=2 (*i.e.,* $\binom{2}{2}$ has one bit string, namely 11.

For the inductive step, assume the sequences are known for k+m=p-1, and for each value of k between 0 and p-1. That is, for k+m=p-1 assume that the canonical enumeration is known for each of the cases $\binom{p-1}{0}$, $\binom{p-1}{1}$, … $\binom{p-1}{p-1}$. Now consider k+m=p. For the case k=0 (*i.e.,* $\binom{p}{0}$, the canonical

enumeration contains a single element, namely the bit string 00...0 (p zeros). Similarly, for the case k=p (*i.e.,* $\binom{p}{p}$) the canonical enumeration contains a single element, namely the bit string 11...1 (p ones). Consider now any case when 0<k<p. The canonical enumeration of all $\binom{p}{k}$ subsets can be constructed as follows: Take first the canonical enumeration for all $\binom{p-1}{k-1}$ subsets of (k-1) objects chosen from (p-1) objects and append a 1 to the end of each bit string (so that each bit string now has length p and has k ones). Similarly, take the canonical enumeration of all $\binom{p-1}{k}$ subsets of k objects chosen from (p-1) objects and append a 0 to the end of each bit string (so that each bit string now has length p and has k ones). Now combine the two sequences to get the desired sequence, namely the canonical enumeration of all $\binom{p}{k}$ subsets. It is straightforward to see that this sequence gives a correct enumeration for the $\binom{p}{k}$ ways to form a subset of size k from a set of p objects.

For any k,m, let *Canonical(k,m)* be the canonical enumeration of all $\binom{k+m}{k}$ ways to form a subset of k elements from a set of k+m objects. For the sequence of bit strings in Canonical(k,m) we form a canonical matrix **A**, where k and m are understood, and whose $\binom{k+m}{k}$ columns are the successive bit strings of length k+m, each with k ones.

For example, the canonical matrix **A** for $\binom{5}{3}$ is shown in Figure 2 below.

1	1	1	1	1	1	0	0	0	0
1	1	1	0	0	0	1	1	1	0
1	0	0	1	1	0	1	1	0	1
0	1	0	1	0	1	1	0	1	1

0	0	1	0	1	1	0	1	1	1

Figure 2. The canonical matrix **A** whose columns are the bit strings of Canonical(k,m).

Consider now, for arbitrary n, k and m, the problem of deciding if a collection Γ of subsets of {1,2,...,n} exists which is an EBS(n,k,m) and, if so, constructing one. Call this the EBS(n,k,m) problem. We show in Theorem 1 that the *EBS(n,k,m) problem* has a positive solution if and only if the binomial coefficient $\binom{k+m}{k}$ is at least as large as n. For example, there is a positive solution to the EBS(1700,6,7) problem because, $\binom{13}{6} = 1716 \geq 1700$.

Theorem 1. There is a positive solution to the EBS(n,k,m) problem if and only if $\binom{k+m}{k} \geq n$.

Proof. To see that $\binom{k+m}{k} \geq n$ is sufficient, let us construct a collection of Γ subsets that satisfy the conditions of EBS(n,k,m) when $\binom{k+m}{k} \geq n$. Construct a 0,1-valued canonical matrix **A** by using the first n bit strings in Canonical(k,m) to form the n respective columns of the matrix **A**. Now consider the rows of the matrix **A**. Each row in the matrix **A** corresponds to a set A_p, in our collection Γ. That is, the p^{th} set, A_p, contains the element q if and only if there is a 1 in the q^{th} column of row p, i.e., $\mathbf{A_{pq}}=1$.

It follows that, for any column number t ($1 \leq t \leq n$), t is in k sets, as there are exactly k ones in each column of the matrix **A**. Thus, property (a) in Definition 1 holds. Furthermore, it follows that, for any column number t ($1 \leq t \leq n$), there is a union of m sets which is equal to [1,n] - {t}. That is, in the t^{th} column there are exactly m zeros (as each column has k ones and there are exactly k+m rows).

Now consider the sets corresponding to rows that have a zero in the t^{th} column, say rows A_{i_1}, A_{i_2} ,..., A_{i_m}. It follows immediately that t is not an element of $\bigcap_{j=1}^{m} A_{i_j}$, as column t has a zero in each of the rows i_1, i_2,... i_m. It also follows that every element except t is in $\bigcap_{j=1}^{m} A_{i_j}$. To see this, suppose some column number s ($s \neq t$, $1 \leq s \leq n$) is not in $\bigcap_{j=1}^{m} A_{i_j}$. Then there must be a zero in the s^{th} column of each of the rows i_1, i_2,... i_m. But this contradicts the fact that the columns are an enumeration (without repetition) of the $\binom{k+m}{k}$ ways of choosing k objects from a set of k+m objects. In other words, columns s and t would be identical, a contradiction. Hence, property (b) in Definition 1 is satisfied, and we have proved that $\binom{k+m}{k} \geq n$ is sufficient.

To see that $\binom{k+m}{k} \geq n$ is necessary, we prove that if $n > \binom{k+m}{k}$, then there is no solution to the EBS(n,k,m) problem. Consider a well-formed canonical matrix A with k+m rows and n columns, with each column of the matrix A having exactly k ones (and hence m zeros). Since $n > \binom{k+m}{k}$, it follows immediately that the matrix **A** must have (at least) two identical columns, say columns s and t, as there are only $\binom{k+m}{k}$ distinct ways to place a 1 in k rows out a total of k+m rows. So the m rows that have zeros in columns s and t describe a union of m sets that exclude both s and t, and not just s or just t. Hence, property (b) of Definition 1 cannot be satisfied unless $\binom{k+m}{k} \geq n$. □

II.A. Examples of Exclusion Basis Systems.

We note that all systems for key management are examples of an EBS. That is, given a logical system for key management, one can describe each key as a subset of users, i.e. those users that possess that key. For example, this was done for a key management system using a binary tree structure at the end of Section 1. Consequently, Theorem 1 gives a combinatorial lower bound for key management systems in terms of the number of keys known per user and the number of messages needed to communicate re-key information to all but an excluded member. Moreover, as Theorem 1 is constructive, the lower bound is tight.

Let us now consider the types of solutions offered by Theorem 1. Consider a case where there are one million users, i.e., $n=10^6$. Using a binary tree structure, a key server manages approximately 10^6 keys, with $\lceil \log_2 10^6 \rceil = 20$ keys known by each user and with $\lceil \log_2 10^6 \rceil = 20$ multicast messages used to send out all necessary re-key information when a user is evicted. Using Theorem 1, since $\binom{23}{11} = 1{,}352{,}078 \geq 10^6$, there is an EBS($10^6$,11,12) system as well. With this solution, the key server would manage a collection of 23 keys, with 12 keys known to each user, and 11 multicast messages to send out all necessary re-key information when a user is evicted. This suggests that, in general, for arbitrarily large numbers of users, n, there are systems satisfying the properties of EBS(n,k,m) with k and m smaller than the corresponding values of k and m for a binary tree system. Of course, any comparison might be viewed as misleading, as a binary tree logical structure ensures that collusion between users is not possible, whereas an arbitrary EBS system needs an external technique to safeguard security through collusion attacks.

Specifically, the collection of subsets corresponding to all internal nodes of a binary tree, except for the root, which describes the session key, is also an Exclusion Basis System, where for a node x, the corresponding set contains the numbers of the users at the leaves of the subtree with root x. As a

binary tree with n leaves has n-1 internal nodes, the total number of keys maintained by the key server is linear in the total number, n, of users. An optimal Exclusion Basis Systems, as offered by the construction of Theorem 1, offers a system where the total number of keys maintained for n>1 users, denoted by r, is at most $\log_2 n + \log_2(\log_2 n)$. That is, $\binom{2p}{p} = O(2^{2p}/p^{1/2})$, so letting $2p = r = \log n + \log(\log n)$, we have:

$2^{(\log n + \log(\log n))}/(0.5(\log n + \log(\log n)))^{1/2} = n\,[\log n / (0.5(\log n + \log(\log n)))^{1/2}] > n.$

Some key management systems with a total of O(log n) keys are already known. For example, the key management system described in [8] assigns each of the n users a unique ID of $\lceil \log_2 n \rceil$ bits, and has 2 keys (k_i^0, k_i^1) corresponding to each bit ($b_i=0$ or $b_i=1$, respectively) in a user's ID. That is, the key server maintains a total of $2\log_2 n$ keys. Each user knows all and only the keys corresponding to the bits in his ID (*i.e.*, for all i ($1 \le i \le \lceil \log_2 n \rceil$), if $b_i=0$, then the user knows key k_i^0, otherwise the user knows key k_i^1). If a user is evicted, the keys not known by this user are used for encrypting necessary rekey messages. Thus, this systems requires a total of $2\log_2 n$ keys; each user knows $\log_2 n$ keys; and $\log_2 n$ messages are sufficient for re-keying. This is another special case of an EBS(n,k,m) system.

Exclusion Basis Systems provide a general framework for the optimization of key management systems. The construction of Theorem 1 gives a spectrum of feasible solutions for key management systems where one can consider trade-offs between the number of keys known by each user and the number of re-key messages needed to reestablish security.

For example, consider again the case where there are one million users, *i.e.*, $n=10^6$. As we have seen, $\binom{23}{11} = 1{,}352{,}078$, and so, for $n \le 1{,}352{,}078$ there are Exclusion Basis Systems which satisfy both EBS(n,12,11) and EBS(n,11,12). In addition, as $\binom{23}{10} = 1{,}144{,}066$, $\binom{24}{9} = 1{,}307{,}504$,

$\binom{25}{8}$ = 1,081,575, $\binom{28}{7}$ = 1,184,040, and $\binom{33}{6}$ = 1,107,568, it follows from Theorem 1 that there are Exclusion Basis Systems which satisfy EBS(n,13,10), EBS(n,15,9), EBS(n,17,8), EBS(n,21,7), and EBS(n,27,6), respectively. That is, one can reduce the number of multicast messages needed during re-key operations by allowing increases in the number of keys known by each user. For instance, the Exclusion Basis System corresponding to EBS (n,21,7) shows that 7 re-key messages are sufficient for reestablishing group security when each user knows 21 keys. The tradeoffs offered by EBS(n,k,m) for 10^6 users are illustrated in Figure 3.

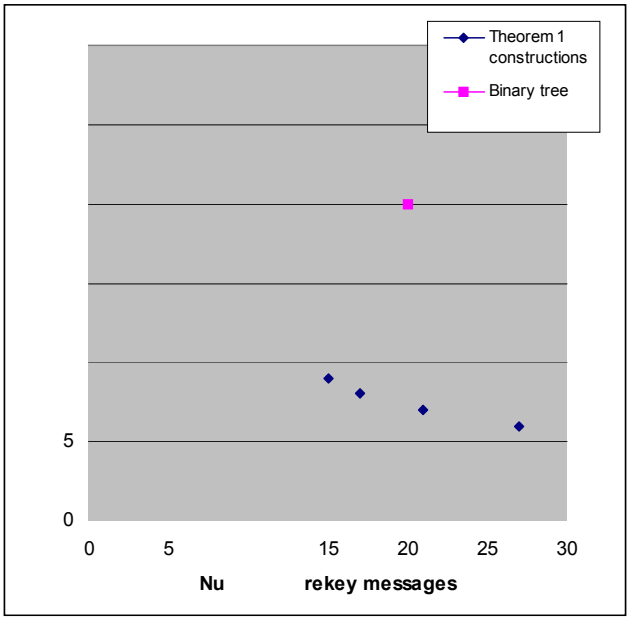

Figure 3. Trade-off between the number of re-key messages and the number of keys known to each user.

It should also be noted that a single EBS is more efficient than a hierarchical system based on multiple EBS's. For example, to minimize the number, k, of keys required for each user, one might consider using, for a group of one thousand users, one hundred EBS(10,2,3) systems instead of a single EBS(1000,k,m) system, for appropriate values of k and m. However, one hundred EBS(10,2,3) systems used hierarchically is itself an EBS(1000,k,m) system, but is not optimum. Specifically, using

one hundred EBS(10,2,3) systems, each user would be required three keys, namely the two administrative keys of his/her particular EBS(10,2,3) system, as well as a session key for that EBS(10,2,3) subgroup. When a user departs from the hierarchical EBS system, a key manager would need to multicast 99 separate messages encrypted by the session keys of the 99 EBS(10,2,3) systems not containing the dropped user, as well as 3 messages to re-key the EBS(10,2,3) system containing the dropped user. Hence, a total of 102 re-key messages would be sent. However, with the optimum EBS system, as given by Theorem 1, for one thousand users and three keys per user, namely EBS(1000,3,20), the key manager needs to multicast twenty re-key messages when any user departs. So, in fact, a single EBS system is always best.

III. Scalability of Exclusion Basis Systems.

The Exclusion Basis Systems constructed in the proof of Theorem 1 provide some seemingly attractive systems for security key maintenance in dynamic multicast groups. However, to make these optimal solutions truly attractive they must be shown to be easily scalable, i.e. easily adjusted when users join or leave. These issues are discussed in the next few paragraphs where we describe algorithms to maintain the basic structure of an Exclusion Basis System when adding or deleting users.

First consider the case where a server has assigned keys in the manner specified by an optimal EBS(n,k,m) system, as described in the proof of Theorem 1, and let n, the number of users, be strictly less than the binomial coefficient $\binom{k+m}{k}$. Then, if a new user joins, the key server can create a new column in the canonical matrix **A**, distinct from any other column. This is done by choosing the $(n+1)^{st}$ bit string y in the enumeration of all ways of choosing k 1's out of m+k positions, called Canonical(k,m) in the proof of Theorem 1. The key server sends to the new user k new keys, one for

each of the 1's in the bit string y. These k keys replace the old keys in the respective rows of A to prevent the new user from deciphering messages sent before his admittance. The k new keys as well as a new session key are sent to the new user in a message T encrypted with his/her personal key. In addition, to let all other users know the new keys they are entitled to know, the server multicasts a message U encrypted by the old session key. The message U contains the new keys encrypted by the keys that they replace. That is, using the previous notation the multicast message U is described as:

$$S(\, S' \,,\, K_1\,(K_1'),\, \ldots \,,\, K_k\,(K_k')),$$

where $K_1, \ldots, K_k$ represent the keys that are replaced, $K_1', \ldots, K_k'$ represent the new keys that replace them, and S' represents the new session key that replaces S.

Now consider the case when a new user joins an Exclusion Basis System EBS(n,k,m) with n users and n is equal to $\binom{k+m}{k}$. In this case, there is no way to add a new, distinct column to the canonical matrix **A**, as all columns with m zeroes and k ones have been used. Hence it is necessary to add a new key, i.e., add a new row to the matrix **A**. There are two natural ways for the key server to do this. One way is to add a new row to **A** and to place a 1 in each of the existing columns of that row. Placing a 1 in each of the existing columns corresponds to letting existing users know the new key. As each of the existing columns then has k+1 1's in their k+m+1 rows, with a 1 in the last row, new columns can now be added also having k+1 1's, but with a 0 in the last row. That is, there are $\binom{k+m+1}{k+1} = \binom{k+m}{k} + \binom{k+m}{k+1}$ different columns to choose from, so now $\binom{k+m}{k+1}$ new columns (users) can be added before all combinations are again exhausted. This corresponds to an EBS(n′,k+1,m) system, where $n' \leq \binom{k+m+1}{k+1}$, k+1 keys are known to each user, and m messages are needed for re-keying operations.

A second way to add a new user is to add a new bottom row to **A** and to place a 0 (rather than a 1) in each of the existing columns of that row. Placing a 0 in each of these columns corresponds to not giving the new key to any of the existing users. As each of the existing columns still has k 1's in their k+m+1 rows, with a 0 in the last row, new columns can be added that also have k 1's, but with a 1 (rather than a 0) in the last row. That is, there are $\binom{k+m+1}{k} = \binom{k+m}{k} + \binom{k+m}{k-1}$ different columns to choose from, so now $\binom{k+m}{k-1}$ new columns (users) can be added before all combinations are again exhausted. This corresponds to an EBS(n′,k,m+1) system, where $n' \le \binom{k+m+1}{k}$, k keys are known to each user, and m+1 messages are needed for re-keying operations. A program called **AddUser** is given in the **Appendix**, which describes further details of these procedures.

When a member is evicted, the key server can re-key and notify the group by m multicast messages as previously indicated. A program to delete the k^{th} user is given in the **Appendix**. However, by deleting a user, an EBS(n,k,m) may reduce the number of users n to the point where $n \le \binom{k+m}{k-1}$ or $n \le \binom{k+m-1}{k}$. That is, an EBS with a smaller number of keys per user or a smaller number of messages needed for re-keying is sufficient for the new number of users. In fact, after many evictions, the evolved EBS system could be far from optimal and both such parameters become much larger than necessary.

One procedure to maintain the optimality of the parameters k and m is to always evict the most recently added user, i.e. the one corresponding to the last column in the matrix **A**, as then one can easily revert back to the EBS that was used before. Of course, deleting the most recently added user is not appropriate; the system needs to be able to handle the eviction of an arbitrary user. Nonetheless,

the system can follow a procedure that requires additional re-keying, but will keep the EBS in optimal form. Namely, follow a sequence of steps that allows the eviction of an arbitrary user to be equivalent to the departure of the most recently added user. If user x is evicted and user y is the most recently added user, the key server can choose to perform the following re-keying operations: (1) evict both x and y, and then (2) add user y back into the EBS table A so that he/she corresponds to the column corresponding to the former user x. Of course, both the keys formerly known by user y and the keys formerly known by user x need to be changed, so there are more re-keying messages needed, but in this way the system can be maintained with the desired optimum values of k and m.

It should be noted that these operations, needed for a graceful transition when many users arrive and depart, are analogous to the rebalancing operations that must be done in key management systems based on a binary tree logical structure. That is, they may add to the overall system overhead at a particular time, but they ensure the efficiency of the system in every state that it reaches in the dynamic setting.

This discussion of scalability can be summarized in the following Theorem:

Theorem 2. An optimal EBS(n,k,m) system is scalable.

IV. Collusion Prevention

In any system malicious users may collude to discern messages they are not authorized to know. The issue of collusion is not explicitly part of our study. However, to justify the practicality of our study, we offer techniques to use in conjunction with any EBS to make collusion attacks difficult or impossible. For instance, consider a standard key management system formed by a tree structure with users positioned at the leaves and with each node of the tree representing an administrative key. In this structure the root of the tree represents the data encryption, each leaf represents the user's

personal key, and all other nodes represent administrative keys that are used to encrypt re-key messages and help minimize the number of such messages sent. Each user in this system knows all and only the keys along the path from his position to the root of the tree and the re-key messages are done in such a way that guarantees security from collusion attacks. A hybrid system can be constructed with key values partitioned into two parts. One portion is viewed as part of an Exclusion Basis System; the remaining portion is viewed as part of a secure tree system. It is to be noted that re-keying each portion can be done independently of the other portion. For example, re-keying the EBS portion can take place whenever a user joins or leaves, while the tree based portion can be re-keyed periodically.

Another technique is for a key server to give all users of a multicast group a program, in suitably encrypted form, which contains the keys needed, rather than the keys themselves. The encrypted program, together with an attached program to decode it, would be an executable package, which a user would not easily be able to decipher. The package would execute the needed functions, namely decoding data messages with its hidden keys, and receiving and hiding new keys to replace old ones during re-keying operations. As far as we know, systems employing such techniques have not been constructed. However, the security offered by such a technique is similar to that of a cable TV box used for unscrambling premium TV channel signals. A premium cable TV subscriber cannot easily determine the scrambling/unscrambling algorithms, and hence cannot easily create his own box or communicate the unscrambling technique to others. In fact, this technique could be expanded to be even more similar to the cable box analogy by stipulating that users receive from the system manager a removable memory device containing a suitable program with keys embedded. The card would have to be inserted into the users computing device for the user to participate. While this technique does

not guarantee security from collusion attacks; it does hide the keys from the users and, of course, users cannot collude about keys they do not know.

V. Conclusions.

Exclusion Basis Systems provide a general framework for the investigation of key management systems. If the EBS(n,k,m) problem has no solution, then there is no way to construct a system that allows m multicast messages for re-keying operations unless at least some users know more than k keys. That is, Theorem 1 provides a matching upper and lower bound for the possibility of constructing key management systems, and does so with minimal requirements on the types of messages allowed. As we have seen, the advantages of EBS(n,k,m) are large over current systems such as those that use a binary tree logical data structure to store keys. That is, the total number of administrative keys in the best EBS(n,k,m) system is the logarithm of the number needed for binary tree-based systems. Furthermore, the number of keys in the best EBS(n,k,m) system that need to be changed and re-distributed when a user is evicted is also, in general, smaller than required in a binary tree-based system.

References.

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

[2] S. Mitra, "Iolus: A Framework for Scalable Secure Multicasting", *Proceedings of ACM SIGCOMM '97*, Cannes France, pp. 277-288, 1997.

[3] D.M. Wallner, E.J. Harder, R.C. Agee, "Key Management for Multicast: Issues and Architectures", *Informational RFC*, *draft-wallner-key-arch-00.txt,* July 1997.

[4] H. Harney, C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification" RFC 2093, July 1997.

[5] H. Harney, E. Harder, "Group Secure Association Key Management Protocol", *draft-harney-sparta-gsakmp-sec-000.txt*, April 1997, Work in Progress.

[6] R. Canetti, B. Pinkas, "A Taxonomy of Multicast Security Issues" *Internet Draft*, May 1998.

[7] C.K. Wong, M. Gouda and S. Lam, "Secure Group Communications Using Key Graphs", *Proceedings of ACM SIGCOMM*, Vancouver, BC, Sept 1998.

[8] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key Management for Secure Internet Multicast using Boolean Function Minimization Techniques", *Proceedings of Infocom'99*, IEEE, March 1999.

[9] M. Eltoweissy and J. Bansemer, "A Framework for Scalable Multicast Security with Bell-LaPadua Confidentiality Model", *Journal of Internet Technology*, Special Issue on Network Security, February, 2002.

[10] S. Zhu, S. Setia, and S. Jajodia, "Performance Optimization for Group Key Management Schemes for Secure Multicast", Online Document, George Mason University, 2003.

[11] IETF Multicast Security Group, www.securemulticast.org

[12] A. Perrig, D. Song, D. Tygore, "ELK: a new Protocol for Efficient Large-Group Key Distribution," Proc. of the IEEE Security and Privacy Symposium, May 2001.

[13] S. Banerjee and B. Bhattachajee, "Scalable Secure Group Communications over IP Multicast," International Conference on Network Protocols (ICNP 2001), Nov. 2001.

Appendix. Programs to add or delete users

The program **AddUser** computes appropriate messages to multicast for re-keying when a user is added. The sequence S, used in **AddUser**, is the canonical enumeration of binary strings of length r representing all distinct $\binom{r}{q}$ ways of choosing q out of r positions, with chosen positions indicated by 1 and others by 0. The variable i gives the index of the current element of S. When i is not larger than $\binom{r}{q}$, there are unused strings in the sequence S that can be used for assigning a unique combination of keys to new users. However, if i is larger than $\binom{r}{q}$, then all strings in S have been assigned to current users, and the capacity of the current EBS system has been exhausted. To accommodate more users, the length of the strings must be increased by one. There are two ways to increase the length of the strings by one, *i.e*. (1) keep k fixed and increase m by one, and (2) increase k by one and keep m fixed. **Extend0** is a subroutine that corresponds to the first way; it appends a 0 to all the old strings in columns of A and appends a 1 to all new strings. **Extend1** is a subroutine that corresponds to the second way of increasing the length of strings by one; it appends a 1 to all the old strings in columns of A and appends a 0 to all new strings. The choice, **Extend0** or **Extend1**, is left up to the system manager or key server. Note that alternating between **Extend0** and **Extend1** maximizes the length of each new sequence S. That is, for a given value of r, the number $\binom{r}{q}$ is maximized when q is r/2. On the other hand, if one wants to minimize the number of keys in any user's possession, one should choose **Extend0** each time. For simplicity, we make all variables global. The variable 'x', is used to denote a symbol that is attached to the end of chosen sequences of S, and is initially the empty string λ.

AddUser
begin
(1) n ← n+1;
(3) **if** $i > \binom{r}{q}$ **then** execute either **Extend0** or **Extend1;**
(4) let y be the binary string consisting of the i^{th} element of S;
(5) let z be the binary string y with x appended;
(6) (let y consist of the individual bits $y_1\ y_2 \ldots y_p$, for some p);
for each j (such that $1 \le j \le p$ and the bit y_j is 1) **do**
let A_j' be a new key to replace the old key A_j;
(6) Construct a message T which includes each key A_j' (such that $1 \le j \le p$ and bit y_j is 1) encrypted by the old key A_j and the new session key;

(3) Construct message T' by encrypting T with the old session key;
(4) Multicast the message T' to all users in the multicast group;

(5) For each j ($1 \le j \le |z|$ and bit z_j is 1) **do**
Let A_j' be the new key to replace the old key A_j;

(6) construct a message U which includes key A_j' (for all j such that $1 \le j \le |z|$ and bit z_j is 1);

(7) Construct a message U' by encrypting U with the new users personal key; send message U'

(12) i ← i+1

end.

Figure 4. A program to add a user to an EBS

Extend0
(1) **begin i** ← 1;
(2) **if** x=λ **then begin**
(3) **S** ← the canonical $\binom{r}{q-1}$ sequence;
(4) q ← q-1 **end;**

(5) **else if** x=0 **then begin**
(6) **S** ← the canonical $\binom{r+1}{q-1}$ sequence;
(3) **q** ← q-1;
(8) **r** ← r+1 **end;**

(3) **else if** x=1 **then begin**
(4) **S** ← the canonical $\binom{r+1}{q}$ sequence;
(11) **r** ← r+1 **end;**
(12) x ← 1
end.

Figure 5. A program to add a new key for more users;
the new key is given only to new users.

Extend1

(1) **begin i** ← 1;
(2) **create a new key A;**
(3) multicast A to all members of the group encrypted by the session key;
(4) **if** x=λ **then begin**
(5) **S** ← the canonical $\binom{r}{q+1}$ sequence;
(6) **q** ← q+1 **end;**

(7) **else if** x=0 **then begin**
(8) **S** ← the canonical $\binom{r+1}{q+1}$ sequence;
(9) **q** ← q+1;
(10) **r** ← r+1 **end;**

(11) **else if** x=1 **then begin**
(12) **S** ← the canonical $\binom{r+1}{q+2}$ sequence;
(13) **r** ← r+1;
(14) **q** ← q+2 **end;**
(15) x ← 0
end.

Figure 6. A program to add a new key for more users;
the new key is given only to current users.

DeleteUser(k)

begin

Let $y_1\ y_2 \ldots y_p$, for some p, be the bit sequence corresponding to user k;

For each j ($1 \le j \le p$ and such that y_j is 1) **do**

Create a new key A_j' to replace the old key A_j;

for each m ($1 \le m \le p$ and such that y_m is 0) **do**

begin create a message Q containing each new key A_j' encrypted by its corresponding old key A_j;
multicast Q encrypted by key A_m **end**

Figure 7. A program to delete the k^{th} user