http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (8): 1106-1115, 2008
ISSN 1812-5638

Combinatorial Based Optimization Technique for Secure Multicast Key Management

P.M. Joe Prathap and V. Vasudevan
Department of Information Technology, Arulmigu Kalasalingam College of Engineering,
KrishnanKoil, 626 190, India

Abstract: This study presents a combinatorial formulation of the multicast key management problem that is applicable not only to the specific problem of multicast key management, but also to the general problem of managing keys for any type of trusted group communication. Specifically, in this study Exclusion Basis Systems (EBS) shows exactly when they exist and demonstrate that such systems represent improvements over the current binary tree-based key management systems and other related systems. This study also simulate the process of subgroup eviction for exclusion basis system and compare the work that needs to be done in a group join and a group-eviction both in EBS and in batch re-keying process. The comparisons of EBS with algorithms for batch Re-keying process show that for group eviction Batch process is more efficient while EBS is for group join.

Key words: Multicasting, key tree approach, group communication, rekeying, exclusion basis systems

INTRODUCTION

Multicasting is a type of communication between computers in a network that enables a computer to send one stream of data to many interested receivers without interrupting computers that are not interested. For these reasons, multicasting has become the favored transmission method for most multimedia and triple play applications, which are typically large and use up a lot of bandwidth. Multicasting not only optimizes the performance of your network, but also provides enhanced efficiency by controlling the traffic on your network and reducing the loads on network devices. This technology benefits many group communication applications such as pay-per-view, online teaching, share quotes, secure teleconferencing, ecommerce, e-banking, command and control, video-on demand and other internet-based services (Varshney, 2002).

Before these group oriented multicast applications can be successfully deployed, access control mechanisms (Canetti *et al*., 1999; Wallner *et al*., 2007; Howarth *et al*., 2004) must be developed such that only authorized members can access the group communication. The only way to ensure controlled access to data is to use a shared group key, known only to the authorized members, to encrypt the multicast data. As group membership might be dynamic, this group key has to be updated and redistributed securely to all authorized members whenever there is a change in the membership in order to provide forward and backward secrecy (Varshney, 2002). Forward secrecy means that a departing member cannot obtain information about future group communication and backward secrecy means that a joining member cannot obtain information about past group communication. This study assumes the existence of a trusted entity, known as the Group Controller (GC), which is responsible for updating the group key. This allows the group membership to scale to large groups. A number of scalable approaches have been proposed and one in particular, the key tree approach (Wong *et al*., 2006; Stinson and Trung, 1998; Balenson *et al*., 2000; Pegueroles and Riceo-Nivella, 2003), is analyzed in detail and extended in this study. In short, the key tree approach employs a hierarchy of keys in which each member is assigned a set of keys based on its location in the key tree. The rekeying cost of the key tree approach increases with the logarithm of the group size for a join or depart request (Li *et al*., 2001; Setia *et al*., 2000; Zhang *et al*., 2003). The operation for updating the group key is known as rekeying and the rekeying cost denotes the number of messages that need to be disseminated to the members in order for them to obtain the new group key.

Individual rekeying, that is, rekeying after each join or depart request, has two drawbacks (Howarth *et al*., 2004; DeCleene *et al*., 2001). First, it is inefficient since each rekey message has to be signed for authentication purposes and a high rate of join/depart requests may result in performance degradation because the signing operation is computationally expensive. Second, if the delay in a rekey message delivery is high or the rate of join/depart requests is high, a member may need a large amount of memory to temporarily store the rekey and data

Corresponding Author: P.M. Joe Prathap, Department of Information Technology,
Arulmigu Kalasalingam College of Engineering, KrishnanKoil, 626 190, India

messages before they are decrypted. Batch rekeying techniques have been recently presented as a solution to overcome this problem. In such methods, a departed user will remain in the group longer and a new user has to wait longer to be accepted. All join and leave requests received within a batch period are processed together at the same time. A short rekey interval does not provide much batch rekeying benefit, whereas a long rekey interval causes a delay to joining members and increases vulnerability from departing members who can still receive the data.

The efficiency of the key tree approach critically depends on whether the key tree is balanced (Lee *et al*., 2002; Moyer *et al*., 2007). A key tree is considered balanced if the distance from the root to any two leaf nodes differs by not more than 1. For a balanced key tree with N members, the height from the root to any leaf node is $\log_k N$, where k is the out degree of the key tree, but, if the key tree becomes unbalanced, then the distance from the root to a leaf node can become as high as N. In other words, this means that a member might need to perform N - 1 decryptions in order to get the group key.

Secure multicast techniques must gracefully tolerate frequent fluctuations in group membership. Membership fluctuations need to be dealt with in an efficient way. In the eviction process, this study uses a technique to change the keys that are known by the evicted user and send the new keys to the ones who are using those keys and are still group members. Similarly, In the join process, add users to the multicast group to be able to send them messages that are supposed to be sent to the group members. In the Internet, multicast has been used successfully to provide an efficient, best effort delivery service to large groups (Canetti *et al*., 1999). This article envisions that deployment of network applications requiring group communications will accelerate in coming years. As a result, securing group communications, i.e., providing confidentiality, authenticity and integrity of messages delivered between group members, will become a critical networking issue in the near future.

An important design goal is to minimize storage requirements and system overhead associated with re-establishing system security when users depart or join a secure multicast group. That is, new cryptographic keys need to be generated and sent out to the remaining users to continue a secure multicast session. These keys clearly should be sent out in the most efficient manner possible. In addition, all session and administrative keys must be stored by the system server and by group members. Keys must be changed when users depart or join in order to provide forward and backward security of multicast data. The literature describes a number of methods for rekeying when single users depart and join (Ng *et al*., 2006; Stinson and Trung, 1998; Wallner *et al*., 2007; Mingyan *et al*., 2002). A common approach is to use a logical tree structure formed by adding administrative keys to the user's personal keys. For example, in a binary tree structure for a group of n individuals, there are n leaves, where each leaf represents a distinct user (and his/her personal key), each internal node represents an additional administrative key and the root represents a session key.

KEY TREE APPROACH

In a typical key tree approach (Pegueroles and Rico-Novella, 2003; Ng and Sun, 2006) as shown in Fig. 1a, there are three different types of keys: Traffic Encryption Key (TEK), Key Encryption Key (KEK) and individual key. The TEK is also known as the group key and is used to encrypt multicast data. To provide a scalable rekeying, the key tree approach makes use of KEKs so that the rekeying cost increases logarithmically with the group size for a join or depart request. An individual key serves the same function as KEK, except that it is shared only by the GC and an individual member.

In the example in Fig. 1a, K0 is the TEK, K1 to K3 are the KEKs and K4 to K12 are the individual keys. The keys that a group member needs to store are based on its location in the key tree; in other words, each member needs to store $1+\log_k N$ keys when the key tree is balanced. For example, in Fig. 1a member U1 knows K0, K1 and K4 and member U7 knows K0, K3 and K10. The GC needs to store all of the keys in the key tree.

To uniquely identify each key, the GC assigns an ID to each node in the key tree. The assignment of the ID is based on a top-down and left-right order. The root has the lowest ID, which is 0. For a node with an ID of m, its parent node has an ID of (m-1)/k, with its children's IDs ranging from km+1 to km+k, as shown in Fig. 1b.

When a member is removed from the group, the GC must change all the keys in the path from this member's leaf node to the root to achieve forward secrecy. All the members that remain in the group must update their keys accordingly. If the key tree is balanced, the rekeying cost

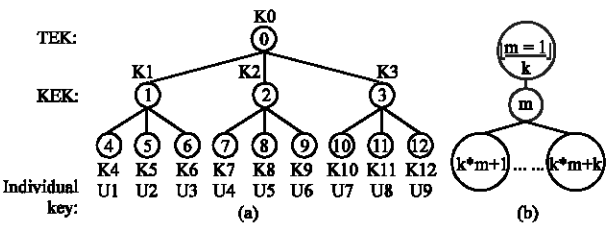

Fig. 1: (a) key tree structure (b) ID assignment

for a single departing member is $k\log_k(N)$-1 messages. For example, suppose member U9 is departing in Fig. 1a. Then, all the keys that it stores (K0 and K3) must be changed, except for its individual key.

If backward secrecy is required, then a join operation is similar to a depart operation in that the keys that the joining member receives must be different from the keys previously used in the group. The rekeying cost for a single joining member is $2\log_k N$ messages when the key tree is balanced.

The efficiency of the key tree approach critically depends on whether the key tree remains balanced. For a balanced key tree with N leaf nodes, the height from the root to the any leaf node is $\log_k N$. However, if the key tree becomes unbalanced, the distance from the root to a leaf node can become as high as N. and if it is unbalanced we can't predict the number of rekeying messages also.

An Exclusion Basis System (EBS) is a generalization of the problem of constructing a logical structure for key management (Morales *et al*., 2003). 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 represent the personal keys of the members. The internal nodes are administrative keys used by the system to facilitate rekey operations when a user is evicted. In the binary tree, a user knows all keys corresponding to nodes on the path from his personal key position 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.

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. To illustrate, consider the complete binary tree of the height of tree shown in Fig. 2.

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 log2n, 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 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.

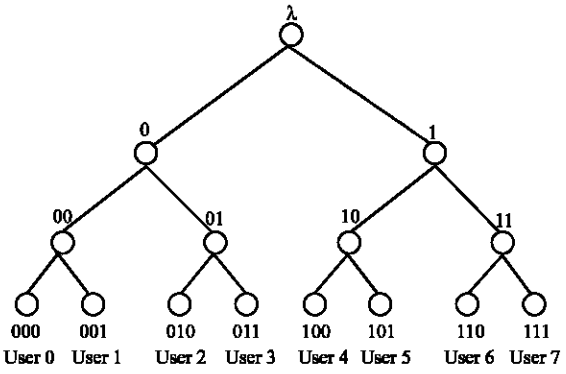

Fig. 2: Binary tree-based key management structure

EXCLUSION BASIS SYSTEMS

A more general scenario of the binary tree structure discussed above is the Exclusion Basis System (EBS) (Morales *et al*., 2003) in which the total number of keys that should be maintained by each user is smaller than the binary tree structure and the number of the rekeying messages can be half of the one in the binary tree. According to Morales *et al*. (2003) EBS stated as follows: 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 in Γ and (b) there are exactly m subsets, say A1, A2, ..., Am , in Γ such that $U_{i=1}^{m} Ai$ is [1,n] - {t}. (That is, each element t is excluded by a union of exactly m subsets in Γ).

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 ith position of a string means that object I is included in that subset, for all i (1 = i = k+m). Note that every bit string in this enumeration will have k 1's. We describe our so-called canonical enumeration of the $\binom{k+m}{k}$ subsets by induction on k+m. the subsets by induction on k+m. For the basis step, when k+m = 2, there are three canonical enumerations 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 $\binom{p-1}{0}$ 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..., $\binom{p}{k}$) 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 for 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 and 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 $\binom{4}{2}$ is shown in Fig. 3.

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≥1700.

0	**0**	**1**	**1**
0	1	**0**	**1**
0	1	1	**1**
1	**0**	**0**	**1**
1	**0**	1	**0**

Fig. 3: The canonical matrix A

It is mathematically proved that, there is a positive solution to the EBS(n,k,m) problem (Stinson and Trung, 1998) if and only if $\binom{k+m}{k} \geq n$.

To illustrate this, we describe EBS (6,2,2). The collection of subsets Γ = {A1 = {1,2,4}, A2 = {1,3,5}, A3 = {2,3,6}, A4 = {4,5,6} }. In this simple example, we can easily verify that each integer t in the interval [1,6] is in exactly two subsets of Γ and each integer t is excluded by a union of exactly 2 subsets in Γ. We borrow the second example and its illustration from (Heydari *et al*., 2006).

An example of an EBS (8,3,2) is the collection of subsets Γ = {A1 = {5,6,7,8}, A2 = {2,3,4,8}, A3 = {1,3,4,6,7}, A4 = {1,2,4,5,7}, A5 = {1,2,3,5,6,8}. One can easily verify that each integer t∈[1,8] is in exactly 3 subsets in Γ and each integer t is excluded by a union of exactly 2 subsets in Γ, as shown below:

[1,8] - {1} = A1 U A2 , [1,8] - {2} = A1 U A3 , [1,8] - {3} = A1 U A4 , [1,8] - {4} = A1 U A5 ,[1,8] - {5} = A2 U A3 , [1,8] - {6} = A2 U A4 , [1,8] - {7} = A2 U A5 and [1,8] - {8} = A3 U A4 .

An Exclusion Basis System (EBS) 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 subset in Γ. In this study, we will use the terms key and subset interchangeably. If the subset Ai is in Γ, then the key Ai is known by each of the users whose number appears in the subset Ai. (For example, in the EBS (8,3,2) instance above, key A1 is known by users 5, 6, 7 and 8 and by no others.) Furthermore, for each t∈[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 evict any user t, rekey 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 the eviction of other users is handled in an analogous manner). User 1 knows keys A3, A4 and A5, so these keys need to be changed and the new values sent out to authorized users. Observe that [1,8]-{1} = A1UA2, so we show that two messages, encrypted by keys A1 and A2, respectively, are sufficient to distribute the new keys to authorized users. Let the first message be one encrypted with key A1 and which contains four subparts.

The four parts of the message are: (1) a new session key, S', (2) replacement key for A3 encrypted by the former A3 key, (3) replacement key for A4 encrypted by the former A4 key and (4) replacement key for A5 encrypted by the former A5 key,

In other words, the first message is represented as: A1(S',A3(A'3,), A4(A'4,) A5(A'5,)), where Ai(x) denotes encryption of x by key Ai and A'i represents the replacement key for the old key Ai. In this notation, the second message would be: A2(S',A3(A'3,), A4(A'4,) A5(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 rekey messages since user 1 does not possess keys A1 and A2. At the conclusion of the rekey 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 A1, A2,..., Am in Γ such that $U_{i=1}^{m}Ai$ is [1,n] - {t}, the key server can evict user t by rekeying and sending out m messages, such that the ith message is encrypted with key Ai 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', Ai), 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 Ai. 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.

EBS examples: Consider the case when there are one million users, i.e., n = 10^6. Using the logical structure corresponding to a binary tree, the key server must manage approximately 10^6 keys, with $[\log_2 10^6] = 20$ keys known by each user and with $[\log_2 10^6] = 20$ multicast messages used to send out all necessary rekey information when a user is evicted. Using Theorem 1, since $\binom{23}{11} = 1{,}352{,}078 \geq 10^6$ there is an EBS (23,11,12) system which provides a more manageable solution. With this solution, the key server would manage a collection of 23 keys, with 11 keys known to each user, so that 12 multicast messages would be used to send out all necessary rekey information when a user is evicted. This suggests that, in general, for arbitrarily large numbers of users, there are systems satisfying the properties of EBS (n,k,m) that are superior to the binary tree logical structure. The collection of subsets corresponding to the internal nodes of a binary tree is also an Exclusion Basis System (EBS), where for a node x, the corresponding set contains the user 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. On the other hand, for the Exclusion Basis Systems (EBS) offered by the construction of Theorem 1, the total number of keys maintained by the key server is at most $2\log_2 n$, as the binomial coefficient $\binom{2r}{r} \geq 2^r$, where $2^r = n$ (i.e., $r = \log_2 n$. Some 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 $\log_2 n$ 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 ($1 \leq i \leq \log_2 n$), if bi = 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 know by this user are used for encrypting necessary rekey messages. Thus, this system requires a total of $2\log_2 n$ keys; each user knows $\log_2 n$ keys and $\log_2 n$ messages are sufficient for rekeying. This system is a special case of an EBS (n,k,m) and is not optimal. Exclusion Basis Systems (EBS) 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 rekey 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 $n \leq 1{,}352{,}078$ there are Exclusion Basis Systems (EBS) which satisfy both EBS (n,12,11) and EBS (n,11,12). In addition as $\binom{23}{13} = 1{,}144{,}066$, $\binom{24}{15} = 1{,}307{,}504$, $\binom{25}{17} = 1{,}081{,}575$, $\binom{28}{21} = 1{,}184{,}040$ and $\binom{33}{27} = 1{,}107{,}568$. 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 rekey 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 rekey 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 Fig. 3. That is there is always a tradeoff between the number of rekeying messages needed to multicast and the number of the keys that should be remembered by each user. In comparison, a group key management scheme for 10^6 users based on a binary tree would require each user to know 20 keys and 20 rekey messages would be needed to reestablish group security. Similarly, the method described by Varshney (2002) would also require each user to know 20 keys and 20 rekey messages would be needed to reestablish group security. The solution for 10^6 users provided by systems based on these two methods is also shown in Fig. 4.

Trade-off between the number of rekey messages and the number of keys known to each user in key management systems based on EBS (n,k,m) for 10^6 users. The square marker indicates number of rekey messages and the number of keys known to each user in binary tree-based and other similar key management systems.

Ng and Sun (2006), showed that there is a positive solution to the EBS (n,k,m) problem if and only $\binom{k+m}{k} \geq n$.

EBS for individual rekeying: The Exclusion Basis Systems constructed above provide some seemingly attractive systems for security key maintenance in dynamic multicast groups. However, to make them truly attractive they must be easily scalable for adding and deleting an arbitrary number of users. These issues are discussed in this section where we describe simple algorithms to maintain the basic structure of an Exclusion Basis System based on the combinatories of binomial coefficients.

Adding users: First consider the case where a system satisfying EBS (n,k,m) has n users and n is strictly less than the binomial coefficient $\binom{k+m}{k}$. Suppose a new user joins the system. In this case, the key server can easily create a new column in the matrix A, distinct from any other column, by choosing the $(n+1)^{st}$ bit string in Canonical (k,m). Thus, a user can be added to the system without changing any administrative keys in the system or altering the keys known by any previous user. Only the session key is changed, which can be accomplished by a single multicast message just before the new user is added. Otherwise, the only action required is for the key server to send a message, using the new user's personal key that describes the keys the new user is authorized to

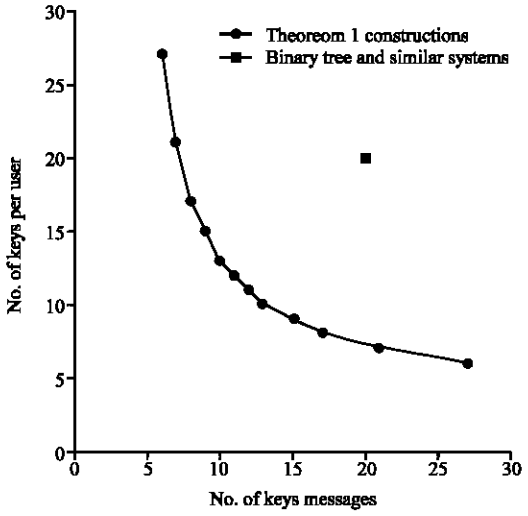

Fig. 4: Trade off between k and m

known. Now consider the case when a new user joins an Exclusion Basis System EBS(n,k,m) where n is exactly equal to $\binom{k+m}{k}$. In this case, there is no way to add a new column in the matrix A that is different from existing columns, as all columns with m zeros (or equivalently, k ones) have been used. The answer, of course, is to add a new key, i.e., add a new row to the matrix A. This can be done by either (1) letting all current users know the new key, which corresponds to adding a 1 in the new row of each of the existing columns of A, or (2) not letting any of the current users know the new key, which corresponds to adding a zero in the new row of each of the existing columns. The first choice corresponds to an increase in the parameter k and the second to an increase in the parameter m. Both choices have been described in the inductive definition of Canonical (k,m). In both cases, the server must multicast the new session key. Aside from this, choice (2) does not require that any additional key be communicated by the key server to existing users; choice (1) requires that the key server communicates the new administrative key to the entire existing group. In the latter case, the communication from the key server can be multicast in a secure fashion. And, in both cases, the new user can receive all authorized key information via a communication from the key server encrypted with the new user's personal key. For choice (1), the keys for the new user would correspond to the 1's in a new column for the matrix A as given by Canonical (k+1,m). That is, this new column has some choice of k+1 ones in the first k+m rows of the matrix A and a zero in the new $(k+m+1)^{th}$ row. For choice (2), the new user's keys would correspond to the 1's in a new column for the matrix A as given by Canonical (k,m+1), but now this new column has some choice of m+1 zeros in the first k+m rows of the

matrix A and a 1 in the new $(k+m+1)^{th}$ row. In either case, as $\binom{k+m+1}{m+1}$ and $\binom{k+m+1}{k+1}$ are larger than $\binom{k+m}{m} = \binom{k+m}{k}$, many new users can then be added before this case is needed again.

Evicting users: When a member is evicted, the key server can rekey and notify the group by m multicast messages as previously indicated. However, this procedure does not reduce any of the parameters, namely, the total number of keys maintained by the key server, the number of keys known per user and the number of messages needed for secure notification of rekeying after the next eviction. Thus, after many evictions, the system may be in an undesirable state with the parameters larger than necessary. One way to keep an overall good state in the system is to control the order of departures. That is, if the last user added were always the one evicted, then the state of the system could revert back to whatever it was before that user was added and no system degradation would occur. Unfortunately, since the departure of users cannot be controlled, the eviction algorithm must instead arrange for a random departure to appear like the departure of the last added user. That is, if an arbitrary user, say x, is evicted and user y is the last added user, the key server could perform the following rekeying operations: (1) perform the steps associated with the eviction of both x and y and then (2) add user y back into the group, but in such a way that user y knows only those keys that replaced the ones user x knew. So, in particular, if adding user y caused the use of an additional key, then the system reverts back to the earlier state, i.e., the one before user y was added. This ensures that the system is maintained with the desired optimum values of k, the number of keys known per user, m, the number of messages needed to communicate new keys upon a user's departure and k+m the total number of keys in the system. If desired, the key server could delay this return to optimum values of k and m until $\binom{k+m}{m} = n+\epsilon$ for some predetermined value of å. Such a delay may be desired if users are joining and leaving rapidly and causing n to fluctuate above and below $\binom{k+m}{k}$. That is, this strategy might be desired in order to prevent the repetitive addition and deletion of rows in the matrix A. 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.

Batch rekeying process: The basic key management algorithms described briefly in the previous section are efficient for handling individual departures or arrivals, but not for cases where a large number of users either arrive or depart at the same time. In such cases, when k users arrive or depart simultaneously, the total number of multicast messages needed for re-keying would be k times the number of re-key messages for each individual. If k were, say, half of the total number of users, then O(n log n) messages would be sent. This is clearly inefficient, as a key management system could ignore the benefits offered by the tree and instead simply rekey through unicasting to each member. This would take O(n) messages (Lee *et al*., 2002). Here we define how the batch processing works to give us better solutions than the previous methods for large number of departures and arrivals. Consider G as set of administrative keys known by user g which is supposed to be evicted from the group. Avoidance set A(G) for a subset G of users is defined as the set of nodes in a binary tree T which:

- For every user g not in G, includes at least one node on the path from g to the root of T and
- For every user g in G, does not contain any node on the path from g to the root of T

The basic idea is that, if G is a set of departing users, then the key manager needs to use administrative keys for encryption that are not known by any user in G and are such that every user not in G can decrypt one of the messages in order to learn the new replacement keys. That is, the keys corresponding to the set A(G) is a set of keys not known to the users in G and these keys can be used to communicate to all remaining users. We are interested in minimum size avoidance sets A(G) for a given subset G.

For example, in Fig. 5, a binary tree T is shown with eight users. If G = {user 0, user 1, user 2}, then the minimum size avoidance set A(G) consists of the nodes 011 and 1. That is, the keys corresponding to nodes 011 and 1 in T can be used to encode re-key messages which let all users, except those in G, know the new keys they need to know in order to continue. Furthermore, the two messages would contain all keys from 011 to the root (not including 011) and all keys from 1 to the root (not including 1). Specifically, the messages would be $E_{011}(<01>||<0>||<\lambda>)$ and $E_1(<\lambda>)$, where <x> denotes a new key to replace x and $E_y(z)$ denotes a message z encrypted by a key y and λ represents the root of the tree. A recursive algorithm called AVOIDANCE_SET is proposed by Heydari *et al*. (2006). It computes the minimum A(G) for any logical tree structure T and set of

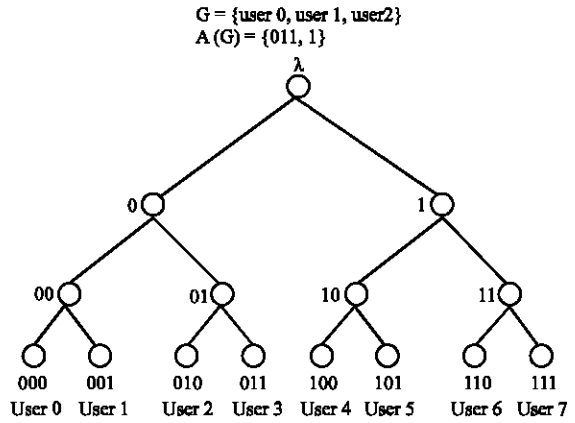

Fig. 5: For the set of departing users {0, 1, 2}, a minimum size avoidance set

departing users G. The algorithm AVOIDANCE_SET(T,x) has two arguments: the first one, T, is the logical tree for the secure multicast group and the second one, x, is a node in the tree T. To compute A(G) for the tree T and the set of users G, one calls the algorithm initially for x = Ω, i.e., the root of the tree.

```
AVOIDANCE_SET(T,x)
if x is a leaf and is in G then stop
else Begin
if the subtree of T with root x has a
leaf that is in G
then Begin
AVOIDANCE_SET(T, left child (x));
AVOIDANCE_SET (T, right child(x))
End;
else Add x to the set A (G) and stop
End;
```

This algorithm works in time linear in the number of nodes in the tree and hence it is also linear in the number of leaves, which is the number of users in the secure multicast group. It also computes the minimum size avoidance set A(G).

Heydari *et al.* (2006) showed that the avoidance set A(G) consists of at most n/2 nodes. They also proved that there is a set G of users for which A(G) must contain n/2 users. The Huffman code is used to create a new balanced, binary tree for key maintenance of n users in which a server sends a total of 2n-2 messages and where each message contains a single key.

The algorithm NEW_TREE is described below:

```
NEW_TREE(Ω)

begin
n ← |Ω|;
for i ← 1 to n-1 do
began
allocate a new node z;
left (z) ← x ← Extract-MIN (Ω);
right (z) ← y ← Extract-MIN (Ω);
height (z) ← 1+max (height (x), height (y))
Insert (Ω, z)
end;
end;
```

That is, let Ω be the collection of all sub trees of T whose roots are elements of A(G).

Analysis: In group evictions, using EBS systems are not efficient and we would better use the algorithms for batch re-keying. As an example, consider a tree in batch re-keying algorithm, with the height of 7 that can give service to 128 users. Each user would know 7 keys so we have k = 7. suppose that a group of 20 users should be evicted from the group. The number of re-keying message would be 2*A(G)-2 as stated before. A(G) would be at most n/2. 2*n/2-2 = 126 would be the number of reeking messages. In EBS(10,7,3) which can serve $\binom{10}{7}$ = 120 users, each user would send 3 re-keying messages. The number of re-keying message would be m*n'*k, where n' is the number of evicted users (3*20*7 = 420). On the other hand for group addition, if we want to add users to EBS system ,

It can be implemented by the following algorithm.

```
//Make different permutations of k "1" in m+k
  positions.
If the number_of_users > (k+m m)
   then ExtendMatrix
ExtendMatrix:
//Adds 0 to all other keys created
before
T = m+k
while m <> 1
        Create Key (T,m-1) // makes new series of the
        keys
   m = m-1
end while
```

So, it would be much more efficient and simpler than using algorithms for batch re-keying process. Since after each extend part we need to add to the height of our tree since the parent node of each user should be changed.

CONCLUSIONS

This study, presents 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 rekey messages. This study develops a general technique for determining optimal values of k and m as a function n and describes the trade-off between k and m. In addition, the work extended with algorithms for admitting and evicting group members and demonstrates the scalability of Exclusion Basis Systems (EBS). This study compared two different methods for group multicasting. The simplest solution to group multicasting is to consider the group as a set of individuals. The problem with this approach is that it is not scalable. 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. Hence, this research article concludes that EBS system is much more efficient than binary trees. Furthermore, the number of needed re-keying messages in EBS (n,k,m) system, when a user is evicted is also, in general, smaller than required in a binary tree-based system. The comparisons of EBS with algorithms for Batch Re-keying process shows that for group eviction Batch process is more efficient while for group join EBS is. We also proposed two algorithms. A simple and more efficient way of constructing the matrix and one for creating a matrix in EBS while extended join occurs. 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 rekeying operations unless at least some users know more than k keys. 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. Present results show that the use of a binary tree for key management is not optimal with respect to the number of keys per user or the number of rekey messages.

REFERENCES

Balenson, D., D. McGrew and A. Sherman, 2000. Key management for large dynamic groups: One-way function trees and amortized initialization. Internet Draft, draft-irtf-smug-groupkeymgmt-oft-00.txt, www.tools.ietf.org/html/draft-irtf-smug-groupkeymgmt-oft-00.

Canetti, R., J. Garay, G. Itkis, D. Micciancio and M. Naor *et al*., 1999. Multicast security: A taxonomy and efficient constructions. Proceedings of IEEE Infocom. 18th Annual Joint Conference of the IEEE Computer and Communications Societies. March 23, IEEE Press, pp: 708-716.

DeCleene, B., L. Dondeti, S. Griffin, T. Hardjono and D. Kiwior *et al*., 2001. Secure group communication for wireless networks. Proceeding of Military Comm. Conference (MILCOM), March 1999, Communication for Wireless Networks, pp: 708-716.

Heydari, M., H.L. Morales and I.H. Sudborough, 2006. Efficient algorithms for batch re-keying operations in secure multicast. Proceedings of the 39th Hawaii International Conference on System Sciences. http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/andtoc=comp/proceedings/hicss/2006/2507/09/25079toc.xml and DOI=10.1109/HICSS.2006.143.

Howarth, M., P. Iyengar, S. Sun and H. Cruickshank, 2004. Dynamics of key management in secure satellite multicast. Selected areas in communications. IEEE J., 22: 308-319.

Lee, P.P.C., J.C.S. Lui and D.K.Y. Yau, 2002. Distributed collaborative key agreement protocols for dynamic peer groups. Proceeding of 10th IEEE International Conference, November 2002, Network Protocols, pp: 322-331.

Li, X.S., Y.R. Yang, M. Gouda and S. Lam, 2001. Batch rekeying for secure group communications. Proceeding of 10th International WWW Conference. http://www10.org/cdrom/papers/pdf/p521.pdf.

Mingyan, Li., R. Poovendran and C. Berenstein, 2002. Design of secure multicast key management schemes with communication budget constraint. IEEE Commun. Lett., 6: 108-110.

Morales, L., I.H. Sudborough, M. Eltoweissy and M.H. Heydari, 2003. Combinatorial optimization of multicast key management. Proceedings of the 36th Hawaii International Conference on System Sciences.

Moyer, M., J.J.R. Rao and P. Rohatgi, 2007. Maintaining balanced key trees for secure multicast. Internet Research Task Force (IRTF), Internet draft, Draft-irtf-smug-key-tree-balance-00.txt.

Ng, W.H.D., H. Cruickshank and Z. Sun, 2006. Scalable balanced batch rekeying for secure group communication. Elsevier's Comput. Security, 25: 265-273.

Pegueroles, J. and F. Rico-Novella, 2003. Balanced batch lkh: New proposal, implementation and performance evolution. Proceeding of IEEE Symposium Computers and Communication (ISCC). 2003, Computers and Communication, pp: 815-815.

Setia, S., S. Koussih and S. Jajodia, 2000. Kronos: A scalable group rekeying approach for secure multicast. Proceeding of IEEE Symposium Security and Privacy.

Stinson, D.R. and V. Trung, 1998. Some new results on key distribution patterns and broadcast encryption. Designs Codes Cryptogr., 14: 261-279.

Varshney, U., 2002. Multicast support in mobile commerce application. Computer, 35: 115-117.

Wallner, D., E. Harder and R. Agee, 2007. Key management for multicast issues and architectures. IETF RFC 2627. http://tools.ietf.org/html/rfc2627.

Wong, C., M. Gouda and S. Lam, 2006. Secure group communication using key graphs. IEEE/ACM Trans. Networking, 8: 12-23.

Zhang, X.B., S. Lam, D.Y. Lee and Y.R. Yang, 2003. Protocol design for scalable and reliable group rekeying. IEEE/ACM Trans. Networking, 11: 908-922.