JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 22, 675-689 (2006)

Storage Efficient Key Management Technique for Secure Multicasting

GANAPATHI PADMAVATHI AND SAMUKUTTY ANNADURAI*
Department of Computer Science
Avinashilingam Deemed University
Tamil Nadu, India
E-mail: mail_padma@yahoo.com
**Government College of Engineering*
Tamil Nadu, India
E-mail: sannalaxmi@yahoo.co.in

Multicast communication will be the communication paradigm of future networks. Secure multicasting is a challenging issue. The main objective of secure multicasting is to distribute the group key to the current members of the group in a scalable manner with minimum overheads. The key distribution methods can be either centralized or distributed. Of these, the centralized methods are simple and robust. In the centralized models, the central controller is an important entity that takes care of key distribution and management. The burden on the central controller is very significant in the centralized models due to the overheads incurred by key distribution. To reduce the load on the central controller, a grouping mechanism based on the behavior of members and a novel key distribution pattern is employed. With this approach, the storage efficiency is improved and the communication bounds are preserved. A comparison in terms of the performance parameters, such as storage and communication updates of the proposed key tree, is made here between our model and the existing architectures. Our model has been simulated, and the results have been found to be optimal.

Keywords: multicast security, group key, communication efficiency, storage efficiency, key distribution

1. INTRODUCTION

Multicast communication is gaining importance for the reason that it will be the preferred communication paradigm of future networks. Many web-based applications, like real time and multimedia applications, are multicast applications. It is an ideal form of communication when an identical message needs to be delivered to a group of receivers [6, 9, 10]. It minimizes the sender overheads a lot, and it is bandwidth efficient. Many important net-based applications, like teleconferencing, distance education, collaborative work, distributed interactive games, stock quotes, software updates and shared white boards, are multicast applications.

When identical data is delivered to the members of group, according to the centralized models, the Central Controller (CC) generates a common session key, encrypts the

Received April 26, 2004; revised March 10, 2005; accepted May 2, 2005.
Communicated by Shiuhpyng Shieh.

key and sends the key to the registered members. Every eligible member receives the key for the session. This common key is also called the ***group key***. Whenever the lifetimes of the members expire or the group's members change, the session key must be changed. The new key must be sent to the eligible members for further decryption. This process is called ***group rekeying***. Thus group rekeying is done when a new member joins the group and an existing member leaves the group. Moreover, it is done so that the new comers will not be able to read the past communications and a leaving member will not be able to read future communications. This is called Perfect Forward Secrecy (PFS), and Perfect Backward Secrecy (PBS) respectively. This key change operation must be scalable, and the key distribution must not increase the overheads, such as the storage and communication overheads. Therefore, multicast security is a key management and distribution problem [6, 9, 10].

The distribution of keys to members is done either in a centralized manner or distributed manner. Centralized schemes are widely used, and they are simple and robust. Many centralized and distributed key management techniques have been proposed in the literature. Of them, some of the important ones are the Group Key Management Protocol (GKMP) proposed by Hugh Harney and Carl Muckenhirn [5], the One-way Function Tree (OFT) of McGrew and Shermann [1], the Logical Key Hierarchy (LKH) developed by Wallner *et al*. [11], the Key Graph (KG) of Wong *et al*. [4], Chang *et al*.'s Boolean function minimization technique [3], Suman Banerjee's Clustering method [2] and the Logical Key Tree with Clusters proposed by Poovendran *et al*. [7]. Most of these models deal with the problem by means of virtually rooted trees with members occupying the leaf positions. The root node corresponds to the session key/group key. The intermediate nodes of the tree store the sub-group keys. In virtual key tree models like LKH and KG, the members occupying the leaf positions store all the node keys in the path from the root node. Therefore, a logical tree of degree 'n' will store $(nN - 1)/(n - 1)$ keys at the GC for a group of size N. Consequently the storage efficiency is of order $O(N)$. When a single member in a leaf node leaves, this corresponds to $n * \log(N) - 1$ update communications. This is also due to the key change operations at the intermediate nodes. Thus, the communication efficiency of a virtual key tree scheme is $O(\log N)$. Hence, storage and communication are the important performance parameters of key distribution models. In all of the above approaches, group members are treated alike, and key distribution takes place without discriminating between or grouping members according to their behavior. However, the behavior of the group members is an important criterion when key changes are considered. The general virtual key arrangements are either ***star*** or ***tree*** arrangements. In the proposed approach, a mixed organization called the ***crossbreed*** **arrangement** is proposed. This structure optimizes the storage requirements while preserving the communication constraints, compared with other methods. The main contributions of this paper are:

- a discussion of the existing virtual key tree architectures;
- a proposed membership grouping and suitable key management architecture;
- a protocol steps for membership operations like join/leave and reconfiguration;
- a comparison of the proposed method with previous methods.

The paper is organized as follows: Section 2 presents the basic concepts in key

management and the existing approaches to centralized key tree organization. Section 3 explains the proposed approach. In section 4, we analyze the experimental and numerical results obtained with the method. Section 5 gives conclusions.

2. EXISTING CENTRALIZED TREE BASED KEY MANAGEMENT TECHNIQUES

Key management techniques are generally classified as centralized or distributed. In this section, we will discuss the important centralized key management techniques. The centralized models are further classified as node based models or key based models. Of them, the centralized key based architectures are taken for study. These models have a central entity, which is responsible for key generation and distribution. Centralized traffic and failure of the central controller are unavoidable drawbacks. These models are suitable for small and moderate groups. In the node-based model, the entire group is divided into various sub groups, and a sub group leader/controller controls each sub group. This is generally done to minimize the centralized traffic and load. Each sub-group may employ different cryptographic techniques. Inter group management and extensive key processing procedures are the major drawbacks of this model. Each method has its own advantages and disadvantages, and they are application dependent. The proposed technique is a hybrid one that enjoys the advantages of both the centralized tree based method and node based method. The functionalities of the central controller are distributed; thus, large group sizes can be handled without many overheads. Moreover, the session key is common, and it follows the uniform key distribution pattern, which, unlike the distributed technique, gives a single group opinion.

2.1 Logical Key Hierarchy (LKH)

Wallner *et al*. [11] proposed a logical structure called Logical Key Hierarchy (LKH). The Central Controller or the Key Distribution Center (KDC) maintains a tree of keys. The keys are arranged in a rooted tree form with the members occupying the leaf positions. Each leaf holds a Key Encryption Key (KEK) associated with one member. The root of the tree is the group key.

Fig. 1 shows a rooted binary tree with eight members. Every member receives and maintains all the keys from the root node to the parent node. The members are assigned a set of encryption keys based on their positions. For a balanced binary tree, each member stores at most $\log_2 N + 1$ keys, where $\log_2 N$ is the height of the tree. For example, user u_1 has the key set $\{k_1, k_{12}, k_{1\text{-}4}, k_{1\text{-}8}\}$. If a member leaves the group or a particular key is invalidated, key update messages are sent by the KDC to the members, instructing them to change the sub-group keys and root key. For a group of size N, the exact storage value for an n-ary tree is $(nN - 1)/(n - 1)$. The communication constraint is $(n - 1)\log_n N$. Therefore, the storage performance scales to $O(N)$, and the communication performance scales to $O(\log N)$.

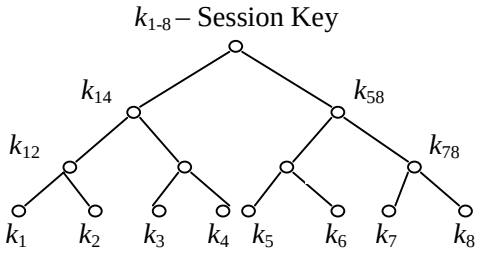

Fig. 1. Logical key hierarchy.

2.2 Key Graph (KG)

Wong *et al.*'s Key Graph [4] is as well based on the centralized approach. There is a trusted server, called the key server, that is responsible for group access control and key management. A hierarchical approach is employed to improve scalability. Instead of a hierarchy of group security agents, covering each sub-group, the KG deals with a hierarchy of keys. The key server knows user group U and key set K, and it maintains a user-key relation, R. Each user has a key, called an individual key, which is shared with the key server and used for pair wise confidential communication. Every member has a minimum of 3 keys, namely, *an individual key, a sub-group key and the group key*. According to this approach, the secure group is a triplet (U, K, R). All the members share the group key. Therefore, the key management problem is actually a key-covering problem, which is generally NP-complete. Two special approaches are available to solve the key-covering problem: *the star* and *tree*.

The star structure shown in Fig. 2 uses only two keys: *an individual key and the group key*. The tree structure shown in Fig. 3 has three keys, namely, *an individual key, a sub-group key and the group key*. For example, according to the star graph, the keys stored by user u_1 are k_1 and k_{1234}. In the tree graph, the keys stored by user u_1 are k_1, k_{12} and k_{1234}.

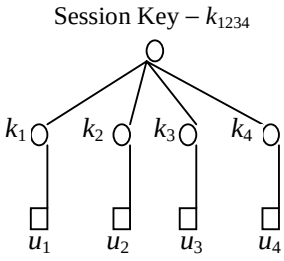

Fig. 2. Star graph.

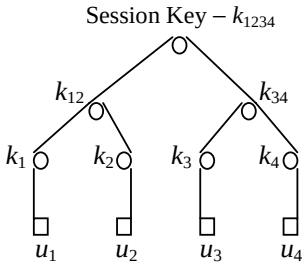

Fig. 3. Tree graph.

The key graph method maintains the user-key relation and adopts three rekey strategies, namely, the *User-oriented, Key-oriented and Group-oriented* strategies. The intermediate nodes are the intermediate keys, which are necessary for sub-group key assignment. To distribute the generated keys, the server needs to perform approximately $d(\log(N))$

operations, where N is the number of users and d is the depth of the tree in a fully balanced tree. Hence, the average computation cost per join/leave is log(N). Actually, there is slight difference in the cost between a join operation and a leave operation. Generally, a join operation is executed at the beginning of a session.

2.3 Logical Key Tree with Clusters

Poovendran *et al*. proposed a hybrid tree [7], where multiple members are assigned to a leaf. The group members are divided into clusters of size M, with every cluster assigned to a unique leaf node. Therefore, for a group of size N, the number of clusters is $\lceil N/M \rceil$. A tree is built with depth $\log_n[N/M]$ and degree n. A binary tree with a cluster size of 4 is shown in Fig. 4. In this case, within each cluster with M members, all the members are assigned a cluster KEK, which is called a common cluster key. This common cluster key is used to update the Session Encryption Key (SEK) within a cluster with a single encryption operation and decryption operation. In this way, the storage requirement can be reduced from $O(N)$ to $O(N/\log N)$.

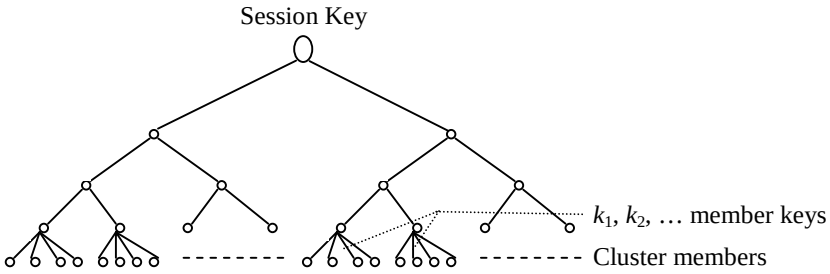

Fig. 4. Logical key tree with clusters.

Generally, the CC performs a pair wise key exchange operation with the members using the clustering method. The CC uses a random seed r as an index for the pseudo-random function f_r to generate the KEK for the individual members. Therefore, only the seed needs to be exchanged and processed. Consequently, the specific design objectives of the centralized secured multicast models become:

- minimization of the storage and communication overheads;
- reduction of the centralized traffic and the burden on the Central Controller so that failures can be graceful.

The above-mentioned objectives are met in this work with two different approaches. First, to reduce the amount of centralized traffic, an efficient grouping mechanism is employed. As a result, the static members act as sub-group heads. The dynamic members are classified under the static heads, using a classification mechanism. The static members perform partial key control operations for their sub-group, thus reducing the burden on the centralized controller. Second, a virtual key distribution technique is employed in order to minimize the overheads.

3. PROPOSED TECHNIQUE

The three important components of the multicast security protocol are initial group establishment, the key distribution protocol and the membership dynamism. In most applications, a sizable set of members are available throughout the communication session, and most of them are about to join or leave during the session. Therefore, based on their behavior, the members are classified as static or dynamic, respectively. The members are generally arranged in three different layers with the source occupying the topmost layer, the static members in the next layer and the dynamic members grouped in the bottom layer.

3.1 Initial Group Establishment and Protocol Design

The proposed membership grouping is shown in Fig. 5. The important component of the security protocol is the protocol for the virtual grouping of members. The grouping protocol groups members in specific sub-groups based on a specific classification mechanism.

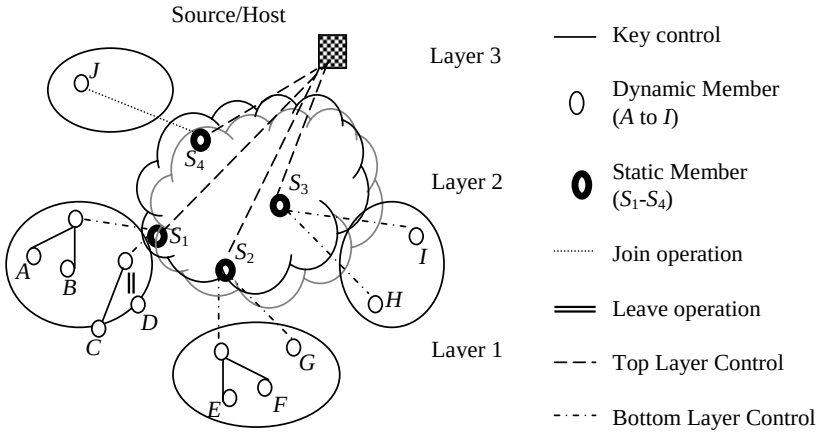

Fig. 5. Proposed grouping method.

Based on this idea, the dynamic members can be classified into particular subgroups based on distance. This can generally be done based on the distance in router hopes from source. The proposed grouping algorithm is shown in Table 1.

The proposed method adopts an authentication mechanism that classifies the members and groups them in layers. A token is issued based on the request following authentication. The details of authentication and access control are beyond the scope of this paper. The sub-group size can be fixed in order to avoid overloading subgroups. Here, the distance function for any two members, $\boldsymbol{x}$ and $\boldsymbol{y}$, may be denoted as $\boldsymbol{d(x, y)}$. The distance $\boldsymbol{d(x, y)}$ may correspond to the router hops between any two members, $\boldsymbol{x}$ and $\boldsymbol{y}$, along the multicast topology from the host or source S in the IP multicast model. The notations used in this protocol are given in Table 2.

Table 1. Grouping algorithm.

The Dynamic members are attached to their heads (Static) based on the following conditions:
A particular member X_1 (Static) is a parent of A (Dynamic) iff
1. $X_1 \in$ {Static}.i.e. in layer 2.
2. For all Xi in layer 2 that satisfy condition (i),
   $d(X_1, A) = \min\{d(Xi, A)$, for $i = 2, 3, \ldots, n.\}$, where n-number of Static members.
3. For every sub-group, the average size can be fixed to k. A particular member will be classified into a sub-group S_r, provided that the sub-group size is within the bound. Otherwise, the member will be classified into the next nearest sub-group, S_{r-1} or S_{r+1}, depending on their sizes.

Table 2. Notations used in this protocol.

S.No.	Notation Used	Meaning
1.	$\rightarrow$	Unicast
2.	$\leftrightarrow$	Multicast
3.	$\Leftrightarrow$	Authentication
4.	$CK(0)$	Session key/Common key at instance 0
5.	H	Host/Source
6.	Si	Static Member i
7.	KEK_{Si}	Key Encryption Key of the i^{th} Static member
8.	KEK_{SGi}	Key Encryption Key of the i^{th} Sub-Group
9.	SGi	i^{th} sub-group
10.	$SGKi(t)$	Sub-group key for the i^{th} group of Subscribers at the t^{th} instance/time
11.	$\{X\}_Y$	Message X is encrypted with Y
12.	$x \rightarrow y : z$	x sends message z to y

Table 3. Key types used in this protocol.

Key Types used	Purpose	Owner of the key	Receiver of the key
Common Key	To encrypt the contents	Host	Members (Static and Dynamic)
Sub-Group Key	For secret communication within a sub-group	Static Member	Dynamic Members
Key Encryption Keys	To encrypt the Common key	Host	Static Members
		Static Members	Dynamic Members

3.2 Key Distribution

The next important phase in the security protocol is key distribution. The key types used and their purposes in the protocol are summarized in Table 3.

The Keys used in this approach are: *the Common key* (*Session key*), *Sub-group key and Individual key encryption key*. The protocol steps for initial group establishment and key distribution are given in Table 4.

Table 4. Protocol steps for initial group establishment.

$H \Leftrightarrow$ {list of group members}
$H \leftrightarrow \{S1, S2, S3, S4\} : \{KEK_{S1}(CK(0)), KEK_{S2}(CK(0)), \ldots, KEK_{Sn}(CK(0))\}$. (For initial establishment)
$Si \leftrightarrow SGi : \{KEK_{SGi}(SGKi(0))\}$, for all the sub-groups $i = 1, 2, 3, 4$ and sub-group members. (For initial establishment)
$Si \leftrightarrow SGi : \{CK(0)\}_{SGKi(0)}$, for $i = 1, 2, 3, 4$.

3.3 Dynamic Membership Management

Members commonly join a multicast session and or leave in the middle of the session in many IP Multicast applications. Therefore, the dynamic membership management is an important component of any security architecture. A key change with each join/ leave ensures that a joining entity is not able to access previously multicast data, and that a leaving entity is not able to continue to access data multicast after it leaves the group. This is an application dependent policy decision. However, a multicast security protocol must be prepared to change the keying material upon each and every join/ leave to protect the integrity of the current group. The dynamic membership management is a required because it is difficult to form static groups. A member wishing to leave in the middle of a session must be allowed to do so. During a membership change, sub-group keys are formed based on the current members of the group. The subscribers' key encryption keys make it possible to decrypt the sub-group keys. This prevents past members or future members of the sub-groups from obtaining the sub-group keys. Member join/leave operations will be explained with an example.

Member leave: As discussed earlier, membership changes are generally performed in layer 1. As shown Fig. 5, if a subscriber '*D*' leaves the session under S_1, the common key and the sub-group key need to be changed. The protocol steps are given in Table 5. It is understood that the common key is sent only to the eligible static members and to those who are currently connected to the host. Following are the required re-key operations:

1. Static member S_1 makes a new common key request. Simultaneously, S_1 sets up a new sub-group key using a pair wise mechanism with each of the remaining members in the sub-group (*A*, *B*, *C*).
2. The host generates a new common key and multicasts it to all the members in layer 2. Members in layer 2 encrypt the new common key with the old common key.
3. Upon receiving the new common key, the members in layer 2, S_1, S_2, S_3 and S_4, extract the new common key and multicast it to the members of their sub-group (dynamic) in layer1. The new common key can be decrypted with the current sub-group key.
4. The members can decrypt the new common key using their current sub-group key.

Table 5. Protocol steps for the member leave operation.

Protocol steps for leave operation:
$D \rightarrow S_1$: {leave request}.
$S_1 \leftrightarrow SG_1$: $\{KEK_{SGi}(SGK_1(0))\}$, for all the members of sub-group 1.
$S_1 \rightarrow H$: {common key request}.
$H \leftrightarrow \{S_1, S_2, S_3, S_4\}$: $\{CK(\text{new})\}_{CK(\text{old})}$.
$Si \leftrightarrow SGi$: $\{CK(\text{new})\}_{SGKi(\text{old})}$ for i = 2, 3, 4.
$S1 \leftrightarrow SG_1$: $\{\text{CK(new)}\}_{SGK1(\text{new})}$.

Table 6. Protocol steps for the member join operation.

Protocol steps for join operation:
$J \rightarrow$ {join request}.
As J joins S_4 according to the authentication mechanism,
$S_4 \leftrightarrow SG_4$: $\{KEK_{SG4i}(SGK_4(0))\}$, for all the members of sub-group 4.
$S_4 \rightarrow H$: {common key request}.
$H \leftrightarrow \{S_1, S_2, S_3, S_4\}$: $\{CK(\text{new})\}_{CK(\text{old})}$.
$Si \leftrightarrow SGi$: $\{CK(\text{new})\}_{SGKi(\text{old})}$, for i = 1, 2, 3.
$S_4 \leftrightarrow SG4$: $\{CK(\text{new})\}_{SGK4(\text{new})}$.

Member join: Suppose subscriber J joins the group as in the Fig. 5, the member is authorized to join the group under sub-group S_4, and the sub-group head S_4 asks the host for a new common key. As discussed above, the sub-group key changes for the affected sub-group. The protocol steps in the above method (steps 2 and 3) are performed. Tables 5 and 6 show the protocol steps.

Reconfiguration: The grouping must be reconfigured in any of the following situations:

1. In an emergency, a static member leaves the group before it times out.
2. The size of the sub-group is out of bound.

The layer key and sub-group keys are generated and distributed using the method discussed above.

As discussed previously, the key distribution mechanism basically depends on the type of the virtual architecture employed. Moreover, the adopted key management technique should be scalable, and the overheads of, for example, storage and the communication must be reduced as the group size increases. The present work concentrates on improving the efficiency of the protocol under these performance constraints. The Source/ Key Server creates a virtual structure called a crossbreed tree. The proposed crossbreed tree is a rooted tree as shown in Fig. 6.

In this scheme, each static member is assigned a sub-group key, which is common to all sub-group members. The common sub-group key is used to update the session key within a sub-group through one encryption and one decryption operation. This is done by organizing the static members in a star arrangement under the root. Here, leaf keys are

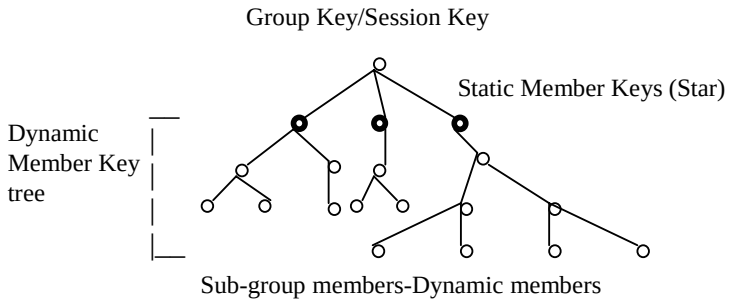

Fig. 6 Proposed crossbreed tree.

allotted to the static members, and the dynamic members form sub-groups under the static heads. A key tree for the dynamic members is maintained by the static heads to facilitate sub-group maintenance.

3.4 Specific Advantages of the Proposed Key Tree

Following are the advantages of the proposed approach:

1. The proposed approach reflects the actual behavior of group members.
2. Unlike the previous methods, the members and the grouping are clearly determined based purely based on their behavior. Generally, when both static and dynamic members form a group, the number of key changes for a static member due to the dynamic member will be significant and will depends on the ratio between them.
3. The virtual key tree can be constructed with the static member keys arranged in a star and the dynamic member keys arranged in a tree. This approach reduces the depth of the virtual key tree, as there is a trade off between the number of stored keys and the depth of the tree.
4. As the static members perform partial key control operations, the amount of centralized traffic and the burden on the central controller are minimized.

4. ANALYSIS

The overall efficiency of any key management technique depends on the following metrics:

1. the number of keys with the server;
2. the number of keys for an individual member;
3. the number of keys for intermediaries/ sub-group controllers;
4. the number of message updates or communication updates due to the key change operations.

The proposed key tree will be analyzed and a comparison between it and the existing techniques made in this section. First, we will analyze the important parameters. The notations used are as defined below:

The group size = n.
The ratio of static to dynamic members = 1 : r.
The total sub-group size = 1 + r.
The number of sub-groups ***ns*** = $n/(1 + r)$.
In other words, the number of static members = $n/(1 + r)$.
Therefore, the number of dynamic members = $n - n/(1 + r) = nr/(1 + r)$.

4.1 Key Storage Requirements

In order to reduce the sender and network resources, secure multicast encryption requires that all group members share a single data encryption key. As discussed earlier, this key must be updated whenever there is a join/leave. When a single group controller performs the rekeying operation to protect the integrity of group communication, the focus must be efficient means of providing the same. However, in dynamic multicast groups, the overheads cannot be calculated in a simple manner, because the group has members that also leave in the middle of the session. In the absence of any other shared key, the group center can use the individual public keys of the valid members. In a simple understanding and assumption, for a group of size n, $n - 1$ encryptions are needed each time a member leaves the group. When a single member leaves, the group center has to encrypt the new group key with the shared key of the individual members. This simple approach requires that *two* keys be stored at the *group center* and requires $n - 1$ encryptions when a member leaves. This is the case for the star structure.

Instead, if the number of encryptions at the group center and if the network resources are measured each time a member leaves, the group center can construct all possible subsets of users. For the above case, there are 2^n possible subsets. Organizing the keys in the form of hierarchy results in the formation of subsets. Once these subsets have been formed, each set is assigned a unique key encryption key. When a member leaves the group, the group center identifies the subset that does not contain the leaving member and uses the corresponding key encryption key to update the group key to the rest of the members. This scheme requires a single encryption. However, at least 2^n keys have to be stored by the group center and 2^{n-1} keys to be stored by every user.

If the key organization mechanism used is the tree type, then $2^n - 1$ is the number of intermediate nodes in a binary tree. The root or the server stores $2 * n - 1$ keys. For a d-ary tree with depth $h = \log_d n$, the general storage size is $1 + 1 + d + d^2 + \ldots + d^h$. Hence, the order of the storage requirement in this case is $O(n)$, where n is the group size [8]. In a rooted tree based key distribution scheme of degree d, each member is assigned $2 + \log_d n$ keys. The depth of the tree is $h = \log_d n$. Deletion of a single member requires that change operations be performed on the intermediate nodes, which leads to $2 + \log_d n$ key updates. The key server (KS) or the CC, whose main job is key management, has to store $d(n + 1) - 2/(d - 1)$ keys [8]. In other words, the KS has to store $2n - 1$ key encryption keys, leading to $O(2n)$.

In the hybrid clustering tree with cluster size m, the depth of the tree to be built is $\log_d(n/m)$. A user needs to store $1 + \log_d(n/m)$ key encryption keys [7]. The number of message updates in this case is given by $(d - 1)\log_d(n/m)$ within the tree and $m - 1$ within the cluster. The total number of keys stored includes the keys on the tree plus the seeds of the clusters, which is equal to $\sum d^i + n/m$, where $i = 0$ to $\log_d(n/m)$.

In the proposed method, the server storage size is $O(n/r) << O(n)$. The proposed tree is a crossbreed tree and is compared with the star graph and tree graph in Table 8. The total number of keys to be stored by the CC is the important criterion for measuring the storage requirement of any architecture. The calculated total key storage requirement for each of the four cases, namely, the star, tree, hybrid cluster tree and crossbreed tree, is shown in Table 8.

Table 8. Comparisons of key storage requirements.

Key Storage	Star	Tree	Hybrid Cluster Tree	Crossbreed
Total no. of keys stored by the CC (No. of users * No. of keys per User)	$2 * n$	$(1 + \log_d n) * n$	$n * [2 + \log_d(n/m)]$	$2 * n/(1 + r) + [n/(1 + r) * \{1 + \log_d r\}]$
No. of keys per User	2	$1 + \log_d n$	$2 + \log_d(n/m)$	2 (static) $1 + \log_d(r)$ (dynamic)

For uniformity in the analysis, the key tree degree was assumed to be the same as d. For the same reason, the cluster size in the hybrid cluster tree and the ration between the static to dynamic members in the crossbreed tree are assumed to be the same. For a size of n users,

Case 1: Star

The number of keys with the user = 2. (*SEK* + *KEK*)
The total number of keys required = $2 * n$.

Case 2: Tree

The number of keys with the user = $1 + \log_d n$ (all the keys in the path from the root to the leaf and the user key).
Total number of keys stored = $n * (1 + \log_d n)$.

In the crossbreed tree, the total number of keys maintained can be calculated as follows:

Case 3: Crossbreed

Number of keys with static members = 2.
Number of static members = $n/(1 + r)$.
Total number of keys for static members = $2 * n/(1 + r)$.
The number of keys stored by the dynamic member = the height of the tree maintained by the static members (h) = $1 + \log_d(r)$.
Total number of keys stored by dynamic members = $n/(1 + r) * (1 + \log_d(r))$.
Total number of keys stored = $1 + n/(1 + r) + n/(1 + r) * (1 + d + d^2 + d^3 + \ldots + d^h)$ $\approx O(n/r)$.

Case 4: Hybrid Cluster Tree

The number of keys stored by the user = $1 + \log_d(n/m) + 1 = 2 + \log_d(n/m)$

(all the keys from the root node to the cluster head + the cluster key), where m is the cluster size.
Total number of keys stored = $n * [2 + \log_d(n/m)]$.

It can be observed that the key storage requirement of the proposed crossbreed tree is lower than those of the other three methods. The storage efficiency scales to $O(n/r) << O(n)$. In the worst case, when $r = 1$, the storage is proportional to $O(n)$.

Actually, the minimum storage can be thought of as an n-ary tree with depth one; in other words, a star graph is the best-suited structure. However, it is not suitable for large groups due to the increased communication overheads. The star arrangement is efficient in terms of the total CC storage. On the other hand, the tree is efficient in terms of communication. In the tree structure, the intermediate nodes store the sub-group keys. The height of the tree determines the number of intermediate nodes. As the degree of the tree increases, the height of the tree decreases; therefore, there is a trade off between the height of the tree and the degree of the tree. This is also an important factor when the total key storage and the traffic in the network are taken into consideration. Moreover, the key storage requirement and the amount of communication can be expressed as functions of the parameters, usually the degree, defining a key distribution scheme [7, 8]. A comparison of the CC storage requirements of the four methods is shown graphically in Fig. 7. The model was simulated, and it was observed that on average, the percentage of improvement was between 80% - 88% compared to the hybrid cluster tree and tree, and 19% compared to the star architecture.

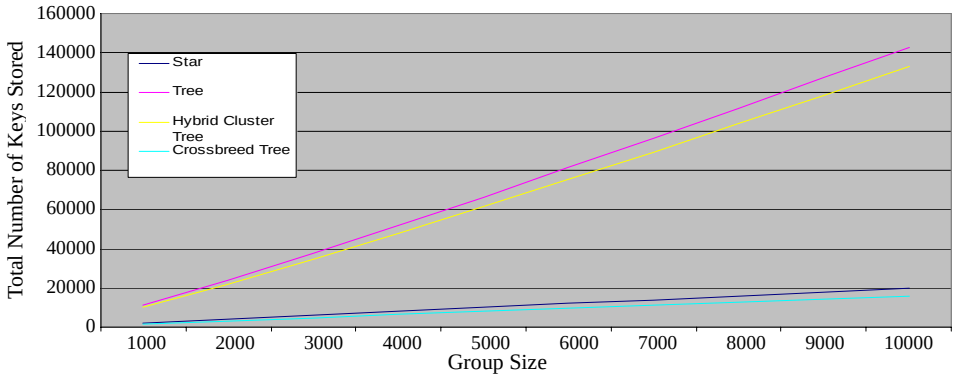

Fig. 7. Comparison of total key storage requirements.

4.2 Communication Updates

The important operations in key management involve joining and leaving members. These dynamic membership operations correspond to key changes, which ultimately correspond to the communication overhead. The overhead due to leave operations is greater than that due to join operations as most of them take place at the beginning of group communication.

Table 9. Comparisons of update communications.

Criteria	Star	Tree	Hybrid Cluster Tree	Crossbreed Tree
Total number of key update messages	$n - 1$	$(d - 1) * \log_d n$	$(m - 1) + (d - 1) * \log_d(n/m)$	$n/(1 + r) - 1$ (among static members) $(d - 1) * \log_d(r)$ (within the sub-group)

When a member is deleted, the total number of key update messages is that shown in Table 9.

The number of update messages exchanged when a member leaves based on the crossbreed tree is $1 + (d - 1) * \log_d r$. The update communication is a function of r. The first order derivative of the communication updates also shows that the greater the value of r, the smaller the amount of update communication. The amount of update communication in the proposed method is proportional to $O(\log r)$. As observed in the above table, the amount of update communication is higher for the star structure because it is not suitable for moderate or big group sizes.

It can be clearly seen that the proposed method achieves significant improvement in terms of the amount of update communication also. The model is simulated for small, moderate and large groups. More than 1000 (2^{10}) requests with all possible combinations of joins/leaves are simulated.

5. CONCLUSIONS

Secure communication is a major issue today as most Internet applications are multicast applications. Centralized key management techniques are simple and robust. Reducing the overheads with a centralized controller and minimizing the overall performance are the important objectives of any secured multicast model. By means of an efficient and flexible approach, these objectives have been met. A membership grouping has been proposed that reflects the actual behavior of the group members. A novel key distribution architecture called the crossbreed tree has been proposed. As minimizing the storage overheads is an important requirement, an effort has been made to do so without increasing the communication overheads. The overheads achieved are minimal compared to those of earlier approaches. A brief analysis of the method has also been given. The results show significant improvement over other methods that employ similar approaches. The model has been simulated, and the results obtained have been found to be optimal.

REFERENCES

1. D. Balenson, D. McGrew, and A. Sherman, "Key management for large dynamic groups: one-way function trees and amortized initialization," *Areas Communication, Special Issue on Middleware*, Vol. 17, 1999, pp. 1614-1631.
2. S. Banerjee and B. Battarcharjee, "Scalable secure group communication for IP multicast," Technical Report, No. CS-TR 4252, Department of Computer Science, University of Maryland, College Park, 2001.

3. I. Chang, R. Engal, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure internet multicast using Boolean function minimization techniques," in *Proceedings of IEEE INFOCOM*, Vol. 2, 1999, pp. 689-698.
4. C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communication using key graphs," *IEEE/ACM Transactions on Networking*, Vol. 8, 2000, pp. 16-30.
5. H. Harney and C. Muckenhirn, "Group key management protocol (GKMP) architecture," RFC 2094, 1997.
6. M. J. Moyer, J. R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications," *IEEE Network*, Vol. 13, 1999, pp. 12-23.
7. M. Li, R. Poovendran, and C. Berenstein, "Design of secure multicast key management schemes with communication budget constraint," *IEEE Communications Letters*, Vol. 6, 2002, pp. 108-110.
8. R. Poovendran and J. S. Baras, "An information theoretic approach for design and analysis of rooted tree based multicast key management schemes," *IEEE Transactions on Information Theory*, Vol. 47, 2001, pp. 2824-2835.
9. S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys*, Vol. 35, 2003, pp. 309-329.
10. W. Trappe, J. Song, R. Poovendran, and K. J. R. Liu, "Key management and distribution for secure multimedia multicast," *IEEE Transactions on Multimedia*, Vol. 5, 2003, pp. 544-557.
11. D. Wallner, E. Harder, and R. Agee, "Key management for multicast: issues and architectures," RFC 2627, 1998.

G. Padmavathi has been a staff member of the Computer Science Department, Avinashilingam Deemed University, and Coimbatore-641 043, INDIA, for 18 years. She received her master's degree in Applied Science and her M.Phil. degree in Computer Science from Bharathiyar University, Coimbatore. She has contributed 20 papers at the national level and 5 papers at the international level. She has 4 publications in the areas fault tolerant real time systems, cryptography and network security.

S. Annadurai is Principal of Government Engineering College, Tirunelveli, Tamil Nadu, INDIA. He received his doctorate degree in Computer Science and Engineering from Anna University, Chennai, in 1995. He has more than 150 publications at the national and international levels. He is and expert committee member for the government of Tamil Nadu and is an active member in many professional organizations. His research interests include image processing, soft computing and network security.