

MULTICAST SECURITY USING KEY GRAPHS AND SECRET SHARING

AHMET M. ESKICIOGLU

*Department of Computer and Information Science, Brooklyn College
The City University of New York, 2900 Bedford Avenue, Brooklyn, NY 11210, USA
Email: eskicioglu@sci.brooklyn.cuny.edu*

MEHMET R. ESKICIOGLU

*Department of Computer Science, University of Manitoba, Winnipeg, MB, R3T 2N2 Canada
Email: rasit@cs.umanitoba.ca*

Multicast communication is increasingly used in applications where the volume of network traffic needs to be minimized. Security of the packets delivered from a source to a large group of receivers presents one of the most challenging research problems for the network architecture. A majority of the proposals for scalable secure multicasting makes use of hierarchical key distribution trees. The Centralized Tree-Based Key Management scheme assigns a unique key to each of the nodes in a member's path to the root. The group key is used by a central server to encrypt multicast data until a member joins or leaves the group. For multicast applications such as pay-per-view where the content has very high value, the group key may need to change frequently. In this paper, we introduce a new approach based on secret sharing in which the server assigns unique secret shares to the nodes in the distribution tree. Our proposal is a prepositioned shared secret scheme that allows the reconstruction of different keys by communicating different activating shares for the same prepositioned information, i.e., the shares stored at the key nodes in the tree.

1 Introduction

The availability of digital technologies and widening Internet bandwidth in recent years has increased the demand for new multimedia services. The Internet service providers are now deploying the new technologies for group communication. Service types include teleconferencing, pay-per-view, video-on-demand, and real-time delivery of stock market information. Security is an important requirement for the distribution networks when the delivery includes either confidential or commercial data.

Secure multicast communication [1,2,3,4] involves efficient packet delivery from one or more sources to a large group of receivers having the same security attributes. The security of the packets is made possible using a common *group key* shared by the members at the specified destinations. Several factors such as the multicast application type and group dynamics influence the way the group key is generated and distributed to members. The desirable attributes for a key management system include:

- securely rekeying the members when a new member joins the group (backward access control),

- securely rekeying the members when a member leaves the group (forward access control),
- minimizing the storage, communication and computation requirements of the participants.

A number of schemes has been proposed for scalable secure multicasting [2,5,6]. An important group of efficient proposals includes hierarchical key distribution trees [7]. They can be classified into hierarchical key based schemes and hierarchical node based schemes. A hierarchical key based scheme assigns a set of keys to each member depending on the location of the member in the tree. Hierarchical node based schemes define internal tree nodes that assume the role of subgroup managers in key distribution. One particular scheme called the *Centralized Tree-Based Key Management* (CTKM) scheme has been proposed in three separate publications [8,9,10].

2 Key Graphs and Secret Sharing

We will use the CTKM scheme formally described by Wong et al. [9] to build our proposal. The server creates and stores a k -ary tree structure having a key at each node. The n leaves of the tree contain the n symmetric keys the server has established with the members of the group. Each member keeps a subset of the server's keys. The elements of the set owned by member i are the keys found along the directed path from the member to the root of the tree, including the leaf key and the root (group) key. The server uses the group key to encrypt the multicast data traffic. All the others keys in the tree are auxiliary keys needed only for efficient key updates. We will use the following notation in the next two sections:

$s: t$ task t is carried out by s .
 $s \rightarrow m: x$ message x is sent from sender s to member (or set of members) m .
 $s \leftrightarrow m$ mutual authentication between s and m ; s sends the leaf key to m .
 $\{x\}_k$ message x is encrypted with the key k .

2.1 Wong et al. Scheme

Definition 1: A key graph is a directed acyclic graph G having two types of nodes: m -nodes (representing members) and k -nodes (representing keys). These nodes are characterized by the number of outgoing and incoming edges they have. Each m -node has one or more outgoing edges but no incoming edge. Each k -node has one or more incoming edges and one outgoing edge. A k -node that does not have an outgoing edge is called a *root*.

Definition 2: A secure group (M, K, R) is a mathematical construct with the following properties: A 1-1 correspondence exists between M and the set of m -nodes in G ; a 1-1 correspondence exists between K and the set of k -nodes in G ; (m, k) is in R if and only if G has a directed path from the u -node associated with m to the k -node associated with k .

Definition 3: A special class of a secure group (M, K, R) is a *tree* whose key graph G is a single-root tree. The height h of the tree is the length of the longest directed path in the tree, and the degree d of the tree is the maximum number of incoming edges of a node in the tree.

Wong et al. investigate three strategies for the construction of rekey messages and their secure distribution to members: user-oriented, key-oriented and group-oriented.

- *User-oriented strategy:* For each user, the server constructs a rekey message that contains precisely the new keys needed by the user and encrypts them using a key kept by the user.
- *Key-oriented strategy:* The server encrypts each new key individually (except the keys for the joining member). A user may therefore have to get multiple rekey messages containing the keys it needs.
- *Group-oriented strategy:* The server constructs a single rekey message that contains all the new keys (except the keys for the joining member).

Undoubtedly, the group-oriented rekeying is the most appropriate strategy from the server's point of view as the workload of the server is minimized. The client, however, favors the user-oriented strategy because it receives a shorter message that does not contain any redundancy. Our overview of the Wong et al. scheme and its comparison with our proposal will be based on the group-oriented strategy since the computational costs of the server and the members are obtained assuming group-oriented or key-oriented rekeying (which show identical performance) [9].

For each strategy, a protocol is described for join and leave operations. It has been observed that the most efficient key graph for group key management is a k -ary tree with $k \cong 4$. A member who decides to join (leave) a group sends a join (leave) request to the server:

- A *join request* from a member initiates a cryptographic process where the two parties perform mutual authentication[†] and agree on an individual symmetric key for secret communications. The server creates a new member node and a new key node for the individual key. It also finds an existing key node, the "*joining point*," in the tree, and attaches the individual key node to the joining point as a child.
- A *leave request* from a member invokes the server to delete the member node and the key node for its individual key from the tree. The parent node that loses its child is the "*leaving point*."

After each join (leave), the server needs to rekey the group for backward (forward) access control.

[†] Authentication is another objective of multicast security. Several approaches exist for authenticating the members during registration and authenticating the source of multicast traffic (packets carrying data or keys).

Example 1: Consider the hierarchical key distribution tree in Figure 1 with $k = 3$ and $n = 9$. It defines the following secure group where each member is assigned three symmetric keys:

$$\begin{aligned}
 M &= \{m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9\} \\
 K &= \{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{123}, k_{456}, k_{789}, k_{1-9}\} \\
 R &= \{(m_1, k_1), (m_1, k_{123}), (m_1, k_{1-9}), (m_2, k_2), (m_2, k_{123}), (m_2, k_{1-9}), (m_3, k_3), (m_3, k_{123}), (m_3, k_{1-9}), \\
 &\quad (m_4, k_4), (m_4, k_{456}), (m_4, k_{1-9}), (m_5, k_5), (m_5, k_{456}), (m_5, k_{1-9}), (m_6, k_6), (m_6, k_{456}), (m_6, k_{1-9}), \\
 &\quad (m_7, k_7), (m_7, k_{789}), (m_7, k_{1-9}), (m_8, k_8), (m_8, k_{789}), (m_8, k_{1-9}), (m_9, k_9), (m_9, k_{789}), (m_9, k_{1-9})\}
 \end{aligned}$$

Using the group-oriented rekeying strategy, we will look at how the group server in Example 1 distributes a new group key for the join/leave operations and periodically replaces the group key.

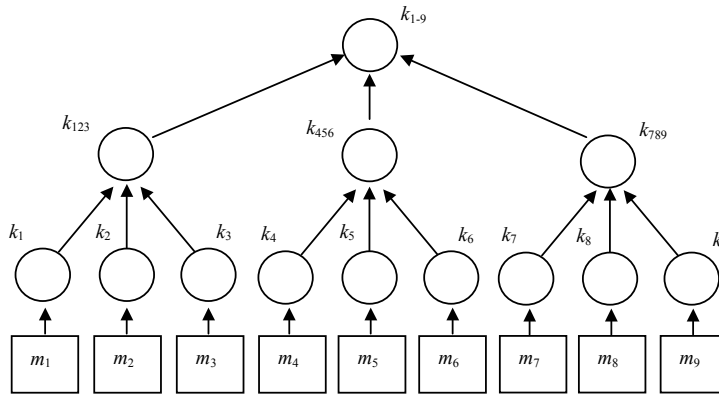


Figure 1. Hierarchical key distribution tree

Leaving a tree key graph: The leaving member will be m_9 . The server deletes the member node and the key node for the individual key from the key graph, and replaces k_{789} at the “leaving point” by k_{78} and k_{1-9} by k_{1-8} . It then constructs and multicasts the following message to the remaining eight members:

$$\begin{aligned}
 L_0: & \{k_{1-8}\}_{k_{123}}, \{k_{1-8}\}_{k_{456}}, \{k_{1-8}\}_{k_{78}} \text{ and } L_1: \{k_{78}\}_{k_7}, \{k_{78}\}_{k_8} \\
 s \rightarrow \{m_1, \dots, m_8\}: & L_0, L_1
 \end{aligned}$$

Joining a tree key graph: The joining member will be labeled m_9 . The server establishes k_9 with the member, creates a new member node and a new key node, and attaches the key node to the existing “joining point.” After changing k_{1-8} to k_{1-9} and k_{78} to k_{789} , it constructs and sends the following two messages:

$$\begin{aligned}
 s \rightarrow \{m_1, \dots, m_8\}: & \{k_{1-9}\}_{k_{1-8}}, \{k_{789}\}_{k_{78}} \\
 s \rightarrow m_9: & \{k_{1-9}, k_{789}\}_{k_9}
 \end{aligned}$$

Changing the group key periodically: For periodic group key change, the server constructs and multicasts the following message to the entire group:

$$k'_{1-9} : \quad \text{new group key}$$

$$s \rightarrow \{m_1, \dots, m_8\} : \{k'_{1-9}\}_{k_{123}}, \{k'_{1-9}\}_{k_{456}}, \{k'_{1-9}\}_{k_{789}}$$

Although this key management may be appropriate for some of the multicast applications, using a single symmetric key for a multicast session until a join or leave occurs may result in weaker security for applications such as pay per view where the multicast content has very high-value. A typical example is a new release of a Hollywood movie that is multicast. In normal circumstances, it is very unlikely for the customers to join after the movie has begun or to leave before the end of the movie. Hence, in the absence of joins and leaves, the movie would be protected with only one symmetric group key.

It is desirable to change the key periodically (or as needed) but this requires rekeying, i.e., multicasting multiple messages. The number of messages depends on the structure of the key tree; for example, if the tree is a full and balanced k -ary tree, k messages have to be multicast. For large values of k , this may be prohibitive for short periods. The conditional access systems for satellite and cable transmissions are known to change the content encryption key every few seconds.

2.2 Centralized Key Management with Secret Sharing (CKMSS) Scheme

Now, we will introduce our proposal based on secret sharing. A (t, n) threshold scheme ($t \leq n$) [11] is a method by which n secret shares S_p , ($1 \leq i \leq n$), are computed from a secret S in such a way that at least t shares are required to reconstruct S . In a *perfect* threshold scheme, knowledge of $(t-1)$ or fewer shares does not change the probability distribution of the possible values of the secret. Shamir's (t, n) threshold scheme [12] uses a random $(t-1)$ -degree polynomial over the finite Galois Field $GF(p)$, i.e., $f(x) = (a_{t-1}x^{t-1} + \dots + a_1x + a_0) \bmod p$:

1. Choose a prime p larger than n and the secret S .
2. Define S to be the constant term a_0 .
3. Construct $f(x)$ by selecting $(t-1)$ random coefficients a_1, \dots, a_{t-1} .
4. Compute the shares by evaluate $f(x)$ at n distinct points, and distribute them to n members.

For the recovery of the secret S , the polynomial

$$f(x) = \sum_{i=0}^{t-1} y_i \prod_{0 \leq j \leq t-1, j \neq i} \frac{(x - x_j)}{(x_i - x_j)}$$

is first constructed from any t of the n shares. S is then computed as $f(0)$. We will now modify Definitions 1-3 to formally express CKMSS:

Definition 1': A set graph is a directed acyclic graph G having two types of nodes: m -nodes (representing members) and s -nodes (representing sets). These nodes are characterized by the number of outgoing and incoming edges they have. Each m -node has one or more outgoing edges but no incoming edge. Each s -node has one or more incoming edges and one outgoing edge. The s -node that does not have an outgoing edge is called a *root*.

Definition 2': A secure group (M, S, R) is a mathematical construct with the following properties: A 1-1 correspondence exists between M and the set of m -nodes in G ; a 1-1 correspondence exists between S and the set of s -nodes in G ; and (m, s) is in R if and only if G has a directed path from the m -node associated with m to the s -node associated with s .

Definition 3': A special class of a secure group (M, S, R) is a *tree* whose set graph G is a single-root tree. The *height* of the tree is length of the longest directed path in the tree, and the *degree* of the tree is the maximum number of incoming edges of a node in the tree.

Example 2: Consider the hierarchical key distribution tree in Figure 1 with the node keys replaced by the node sets. It defines the following secure group where each member is assigned three sets of shares:

$$\begin{aligned}
 M &= \{ m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9 \} \\
 K &= \{ s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{123}, s_{456}, s_{789}, s_{1-9} \} \\
 R &= \{ (m_1, s_1), (m_1, s_{123}), (m_1, s_{1-9}), (m_2, s_2), (m_2, s_{123}), (m_2, s_{1-9}), (m_3, s_3), (m_3, s_{123}), (m_3, s_{1-9}), \\
 &\quad (m_4, s_4), (m_4, s_{456}), (m_4, s_{1-9}), (m_5, s_5), (m_5, s_{456}), (m_5, s_{1-9}), (m_6, s_6), (m_6, s_{456}), (m_6, s_{1-9}), \\
 &\quad (m_7, s_7), (m_7, s_{789}), (m_7, s_{1-9}), (m_8, s_8), (m_8, s_{789}), (m_8, s_{1-9}), (m_9, s_9), (m_9, s_{789}), (m_9, s_{1-9}) \}
 \end{aligned}$$

The CKMSS scheme is a *prepositioned shared secret scheme* [13,14], which makes it possible to reconstruct different keys by communicating different activating shares for the same prepositioned information (i.e., the shares stored at the key nodes). This idea has also been applied for key transport in conditional access and message authentication systems [15,16]. We use the following procedures for the computation of the fresh keys:

Chosen key: If only one key is to be generated, it is possible to choose the key (as the constant term of the polynomial) and then pick an activating share on the constructed polynomial. Let S be the set of shares to be used.

1. Choose $a_0 = K$.
2. Construct the polynomial $f(x)$ passing through K and all the elements in S .
3. Assign a point on $f(x)$ as the activating share that is different from K and all the elements of S .

Chosen activating share: If multiple keys are to be generated, the activating share is chosen and the keys are obtained after the construction of the corresponding polynomials. Let l be the number of keys to be generated, and S_1, S_2, \dots, S_l be the sets of shares to be used.

1. Choose the activating share.
2. for $i = 0$ to l do
 - begin
 - Construct the polynomial $f_i(x)$ passing through the activating share and all the elements of S_i ;
 - Compute $f_i(0)$;
 - Assign $f_i(0) = K_i$;
 - end.

Using the group-oriented rekeying strategy, we will look at the behavior of the server in Example 2 to distribute a new group set for the join/leave operations and periodically replace the group set.

Leaving a tree key graph: The leaving member will be m_9 . The server deletes the member node and the set node for the individual set from the set graph, and replaces s_{789} at the “leaving point” by s_{78} and s_{1-9} by s_{1-8} . It then constructs and multicasts the following message to the remaining eight members:

$$L_0: \{s_{1-8}\}_{k_{123}}, \{s_{1-8}\}_{k_{456}}, \{s_{1-8}\}_{k_{78}} \text{ and } L_1: \{s_{78}\}_{k_7}, \{s_{78}\}_{k_8}$$

$$s \rightarrow \{m_1, \dots, m_8\}: \text{activating share, } L_0, L_1$$

where the fresh keys k_{123} , k_{456} , k_{78} , k_7 and k_8 are obtained using the activating share and the sets s_{123} , s_{456} , s_{78} , s_7 and s_8 , respectively. The leave protocol for the CKMSS scheme is given in Figure 2.

Notation

s_m :	individual set of shares
k_m :	key obtained using the activating share and s_m
x_{j+1} :	deleted set node for s_m
x_j :	the leaving point
x_0 :	the root
x_{i-1} :	the parent of x_i , $i = 1, \dots, j$
S'_0, \dots, S'_j :	sets chosen as the new share sets of x_0, \dots, x_j
J_1, \dots, J_r :	fresh keys at the children of x_i in the updated tree.
L_i :	$\{S'_i\}_{J_1}, \dots, \{S'_i\}_{J_r}$
$memberset(s)$:	set of members holding set s in the set of share sets for the tree

Protocol Steps

1. $m \rightarrow s$: activating share, $\{leave - request\}_{k_m}$
2. $s \rightarrow m$: activating share, $\{leave - granted\}_{k_m}$
3. s : find the leaving point and remove s_m from the tree
4. $s \rightarrow memberset(S'_0)$: activating share, L_0, \dots, L_j

Figure 2. Leave protocol for the CKMSS scheme (with group-oriented rekeying)

Joining a tree key graph: The joining member will be labeled m_9 . The server establishes s_9 with the member, creates a new member node and a new set node, and attaches the set node to the existing “joining point.” After changing s_{1-8} to s_{1-9} and s_{78} to s_{789} , it constructs and sends the following two messages:

$$s \rightarrow \{m_1, \dots, m_8\}: \text{activating share, } \{s_{1-9}\}_{k_{1-8}}, \{s_{789}\}_{k_{78}}$$

$$s \rightarrow m_9: \text{activating share, } \{s_{1-9}, s_{789}\}_{k_9}$$

where the fresh keys k_{1-8} , k_{78} and k_9 are obtained using the activating share and the sets s_{1-8} , s_{78} and s_9 , respectively. The join protocol for the CKMSS scheme is given in Figure 3.

Notation

s_m :	individual set of shares
k_m :	key obtained using the activating share and s_m
x_j :	the joining point
x_0 :	the root
x_{i-1} :	the parent of x_i , $i = 1, \dots, j$
S_0, \dots, S_j :	old share sets of x_0, \dots, x_j
S'_0, \dots, S'_j :	sets chosen as the new share sets of x_0, \dots, x_j
K_0, \dots, K_j :	keys obtained using the activating share and S_0, \dots, S_j , respectively
<i>memberset</i> (s):	set of members holding set s in the set of share sets for the tree

Protocol Steps

1. $m \rightarrow s$: join request
2. $s \leftrightarrow m$: mutual authentication and delivery of s_m
3. s : find a joining point and attach s_m
4. $s \rightarrow \text{memberset}(S_0)$: activating share, $\{S'_0\}_{K_0}, \dots, \{S'_j\}_{K_j}$
5. $s \rightarrow m$: activating share, $\{S'_0, \dots, S'_j\}_{k_m}$

Figure 3. Join protocol for the CKMSS scheme (with group-oriented rekeying)

After the execution of the join and leave protocols, once the new group set is distributed to all the members, the server can start multicasting encrypted data. A fresh group key is generated every time a different activating share is used.

Changing the group key periodically: For periodic group key change, the server constructs and multicasts the following message to the entire group:

$$s \rightarrow \{m_1, \dots, m_9\}: \text{activating share}$$

The activating share is used by the members to generate the new group key k_{1-9} . The group key change protocol for the CKMSS scheme is given in Figure 4.

Notation

x_0 :	the root
S_0 :	current set of x_0

Protocol Steps

1. $s \rightarrow \text{memberset}(S_0)$: activating share

Figure 4. Periodic group key change protocol for the CKMSS scheme

2.3 Security of the CKMSS Scheme

The level of security of the scheme is determined by the number of shares at each node (and hence by the degree of the polynomial), providing scalability for applications with different security attributes. In the analysis that follows, we will consider an isolated situation where the hacker tries to find the prepositioned information, i.e., the set of shares, at a given key node.

Assume that the activating share is sent in-the-clear. It can be captured by the hacker in an attempt to find the shares kept at the node by the member. The scheme is exposed to brute-force attacks for lower degree polynomials.

In polynomial construction, the value of prime p is chosen such that $p > 2^{key_length}$, where key_length denotes the length of the secret key.

Case $t = 2$:

$f_i(x) = a_{i1}x + a_{i0}$: the polynomials of degree one.

The scheme is most vulnerable for first-degree polynomials. If the hacker finds any two keys by cryptanalysis, he can use them with the corresponding activating shares to construct two straight lines. The intersection of these lines gives the prepositioned information. Because of the linearity property, the first-degree polynomials associated with the keys K_i , $i = 1, 2, 3, \dots$, all pass through the same intersection point.

Case $t = 3$:

$f_i(x) = a_{i2}x^2 + a_{i1}x + a_{i0}$: the polynomials of degree two.

The prepositioned information is the two points of intersection of *all* the second-degree polynomials associated with the keys K_i . Each pair $\{(0, K_i), (x_i, y_i)\}$, $i = 1, 2, 3, \dots$, introduces two linear equations in 3 variables or, in reduced form, one linear equation in 2 variables. To find the prepositioned information, one needs to construct all the equations and obtain the solution of the set. In real applications, the key will change frequently leading to a very big set of K_i 's (and second-degree polynomials). Hence, it is practically not possible to derive a complete set of equations to find the "key." Furthermore, this set would represent an *undetermined* linear system.[‡]

Case $t > T$ (where T is a threshold):

Cryptanalysis becomes increasingly more difficult for higher values of t . For each polynomial of degree $(t-1)$, $(t-1)$ of the t pieces of the shared secret are kept by the member. The only data available to estimate these pieces is a pair of points, i.e., the activating share and the recovered key. In general, each pair can be used to

[‡] A linear system $Ax=b$, where A is a $m \times n$ matrix, x a n -dimensional vector, and b a m -dimensional vector, is *undetermined* if $m < n$. In our case, the undetermined system has an infinite number of solutions.

construct a set of 2 linear equations in t variables. As the point $(0, K_i)$ determines the value of a_{i0} (which is one of the variables), the set can actually be reduced to 1 linear equation in $(t-1)$ variables.

The above analysis represents a worst-case scenario for the multicast architecture. Because of the dynamic nature of the join and leave operations, the only permanent set of shares kept by each member m_i is its individual set s_i (which is kept until the member leaves the secure group). For a given member, this set is used by the server when there is a need to send a message to that member only. The other sets of shares would normally change dynamically.

The robustness of the scheme can be increased by several modifications:

1. Defining the encryption key as a secret function of the shared secret would hide the real value of the key. In Shamir's threshold scheme, the coefficient a_0 (the y-intercept of the graph of the polynomial) is defined to be the key. Generalization of this definition is possible to allow other ways of defining the key.
2. Making the degree of the polynomial a secret system parameter (preferably time-dependent) would introduce an additional dimension of cryptanalytic difficulty. The number shares kept at each node is an important piece of information for key recovery.
3. Using a secret function of the activating share to construct the encryption key would result in increased security level. An unkeyed hash function can be conveniently used for this purpose. The server would use the hash value of the activating share to generate the encryption key but transmit the share instead.
4. Multiple redundant activating shares can be sent to conceal the identity of the actual activating share. A predefined process would then be needed by the members to filter out the unneeded shares.

3 Comparison of CTKM and CKMSS

Three measures are used to compare CTKM and CKMSS: *Storage cost*, *communication cost* and *computational cost*. Based on these measures, we observe the following similarities and differences between the two schemes:

Similarities:

- *Storage*: The number of encryptions and decryptions required by join/leave operations are the same in both schemes.
- *Communication*: The size of the messages sent on join/leave operations are the same in both schemes.

Differences:

- *Storage*: In the CKMSS scheme, neither the server nor the members need to store the node keys generated after each rekeying. They can be deleted as soon

as they are used in the decryption process. The sets (both the group set and the auxiliary ones), however, need to be kept until they are replaced. There is a 1-1 correspondence between the number of keys generated for each member and the number of sets held by each member.

- *Communication*: An additional communication cost in the CKMSS scheme for join/leave operations is the delivery of the activating share. The two schemes have different requirements in periodic rekeying. The communication cost for the CKMSS scheme is the delivery of the activating share and the communication cost for the CTKM scheme is the delivery of d encrypted messages.
- *Computation*: An additional computational cost in the CKMSS scheme for join/leave operations is the processing needed for the construction of the polynomials. There is a 1-1 correspondence between the number of polynomials constructed by the server and the number of encryptions performed by the server. There is also a 1-1 correspondence between the number of polynomials constructed by each member and the number of decryptions performed by each member. The two schemes have different computational requirements to recover the group key in periodic rekeying. The CKMSS scheme needs one polynomial construction for the server and one polynomial construction for each member whereas the CTKM scheme needs d encryptions for the server and one decryption for each member.

The above observations are summarized in Table 1. In the comparison, quantitative results from [9] are used. We assume that the tree is full and balanced.

Table 1. Comparison of CTKM and CMKSS schemes

	CTKM	CMKSS
# of keys held by the server	$dn/(n-1)$	————
# of keys held by each member	h	————
# of share sets held by the server	————	$dn/(n-1)$
# of share sets held by each	————	h

(a) Storage Cost

	CTKM	CMKSS
Join	$O(\log_e(n))$	$O(\log_e(n))$ and $O(1)$
Leave	$O(d \log_e(n))$	$O(d \log_e(n))$ and $O(1)$
Periodic rekeying	$O(d)$	$O(1)$

(b) Communication cost

	CTKM			CMKSS				
Encryption/Decryption		Requesting member	Non-requesting member	Server		Requesting member	Non-requesting member	Server
	Join	$h-1$	$d/(d-1)$	$2(h-1)$	Join	$h-1$	$d/(d-1)$	$2(h-1)$
	Leave	0	$d/(d-1)$	$d(h-1)$	Leave	0	$d/(d-1)$	$d(h-1)$
		All members		Server		All members		Server
	Periodic rekeying	1	d	Periodic rekeying	0		0	
Polynomial construction	—				Requesting member	Non-requesting member	Server	
	—			Join	$h-1$	$d/(d-1)$	$2(h-1)$	
	—			Leave	0	$d/(d-1)$	$d(h-1)$	
	—				All members		Server	
	Periodic rekeying	1		Periodic rekeying	1		1	

(c) Computational cost

4 Conclusions

We presented a new centralized tree-based key management scheme for multicast security. Based on secret sharing, it replaces the keys at the nodes of the hierarchical tree by secret shares that are used for constructing different keys for different activating shares.

Some notable characteristics of the CMKSS scheme are:

- For join/leave operations, there are additional costs for storage (to hold the sets), computation (to construct the polynomials) and communication (to deliver the activating share). Polynomial construction requires the solution of a set of linear equations whose size depends on the degree of the polynomial. By construction, the set always has a unique solution, and there are efficient numerical solvers in the relevant literature. Delivery of the activating share is a trivial task when compared to message sizes of $O(\log_d(n))$ and $O(d \log_d(n))$.
- For periodic rekeying, only one message (i.e., the activating share) needs to be multicast to the entire group as opposed to k encrypted messages, each containing the group key, for a full and balanced k -ary tree. The savings may be substantial for a period of a few seconds. In the Wong et al. scheme, the group key is recovered after each member decrypts the message containing the group key. The proposed CKMSS scheme, however, does not need to send any encrypted messages for the protection of the group key as the group key is constructed when the activating share is received by the members. This is in contrast with the use of a “fixed” Group Key Encryption Key (GKEK) [17] that

can be shared by the multicast group for use in periodic rekeys. Two drawbacks of GKEK are noted:

- 1) Potential security drawback – all future group keys are encrypted with the same key.
 - 2) No benefit to a multicast group if a rekey is necessary due to the known compromise of one of the members.
- Two other well-known schemes present potential problems in frequent key updates. In the Dual Encryption Protocol (DEP) [18], the distribution of the group key is protected by dual encryption under the corresponding subgroup key and the key encryption key (KEK). Replacement of the KEKs is a complex and costly procedure, and should be done infrequently. In IOLUS [19] the new subgroup key for each subgroup is multicast encrypted under the old subgroup key, creating a chain of ciphertexts. This is an important security weakness as a compromise in one key would lead to the disclosure of all the keys used in the following links.
 - For sufficiently high values of the polynomial degree and for sufficiently large values of prime p , the problem of finding the prepositioned information at a node would be intractable. An increase in the degree of the polynomial implies an increase in the number of stored secrets, and an increase in the value of p provides a larger space for choosing the locations of the secrets in the xy -plane.
 - All the keys used in encrypting the new secret shares for join/leave operations are newly generated keys. This increases the cryptographic strength of the key management scheme. In the Wong et al. scheme, only the compromised keys are replaced, the others are left unchanged.
 - Depending on the memory limitations of the members and secure requirements of the application, each key node may be assigned a different number of shares for scalability.
 - The scheme can also be adapted to hierarchical node based schemes. This would allow the subgroup managers to establish their own key management scheme based on secret sharing.

References

1. P. S. Kruus, "A Survey of Multicast Security Issues and Architectures," Proceedings of the 21st National Information Systems Security Conference, Arlington, VA, October 1998.
2. L. R. Dondeti, S. Mukherjee and A. Samal, "Survey and Comparison of Secure Group Communication Protocols," Technical Report, University of Nebraska-Lincoln, June 1999.
3. M. J. Moyer, J. R. Rao and P. Rohatgi, "A Survey of Security Issues in Multicast Communications," IEEE Network, pp. 12-23, November/December 1999.

4. T. Hardjono and G. Tsudik, "IP Multicast Security: Issues and Directions," *Annales de Telecom*, pp. 324-334, July-August 2000.
5. D. Bruschi and E. Rosti, "Secure Multicast in Wireless Networks of Mobile Hosts and Protocols and Issues," *ACM Baltzer MONET Journal*, Special Issue on Multipoint Communication in Wireless Networks, 2000.
6. S. Rafaei, "A Decentralized Architecture for Group Key Management," PhD Appraisal, Computing Department, Lancaster University, England, September 2000.
7. L. R. Dondeti, S. Mukherjee and A. Samal, "Comparison of Hierarchical Key Distribution Schemes," *Proceedings of IEEE Globecom Global Internet Symposium*, Rio de Janeiro, Brazil, December 1999.
8. D. Wallner, E. Harder and R. Agee, "Key Management for Multicast: Issues and Architectures" RFC 2627, June 1999.
9. C. K. Wong, M. G. Gouda and S. S. Lam, "Secure Group Communications Using Key Graphs," Department of Computer Sciences, The University of Texas at Austin, Technical Report TR-97-23, July 1997.
10. G. Caronni, M. Waldvogel, D. Sun and B. Plattner, "Efficient Security for Large and Dynamic Groups," Technical Report No. 41, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, February 1998.
11. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
12. A. Shamir, "How to share a secret", *CACM*, Vol. 22, No. 11, November, pp. 612-613, 1979.
13. G. J. Simmons, "How to (really) share a secret," *Advances in Cryptology – CRYPTO '88 Proceedings*, Springer-Verlag, pp. 390-448, 1990.
14. G. J. Simmons, "Prepositioned shared secret and/or shared control schemes," *Advances in Cryptology – EUROCRYPT '89 Proceedings*, Springer-Verlag, pp. 436-467, 1990.
15. A. M. Eskicioglu, "A Key Transport Protocol for Conditional Access Systems," *Proceedings of SPIE Security and Watermarking of Multimedia Content III*, Vol. 4314, pp. 139-148, San Jose, CA, January 22-25, 2001.
16. Eskicioglu, A. M., "A Prepositioned Secret Sharing Scheme for Message Authentication in Broadcast Networks," *Communications and Multimedia Security Issues of the New Century*, IFIP TC6/TC11 Fifth Joint Working Conference on Communications and Multimedia Security (CMS'01), pp. 363-373, Darmstadt, Germany, May 21-22, 2001.
17. A. Ballardie, "Scalable Multicast Key Distribution," RFC 1949, May 1996.
18. L. R. Dondeti, S. Mukherjee and A. Samal, "A Dual Encryption Protocol for Scalable Secure Multicasting," *Fourth IEEE Symposium on Computers and Communications*, Red Sea, Egypt, July 6-8, 1999.
19. S. Mitra, "Iolus: A Framework for Scalable Secure Multicasting," *Proceedings of the ACM SIGCOMM '97*, Cannes, France, pp. 277-288, September 1997.