

# Multi-layer multicast key management with threshold cryptography

Scott Dexter<sup>\*a</sup>, Roman Belostotskiy<sup>†a</sup>, Ahmet M. Eskicioglu<sup>‡a</sup>

<sup>a</sup>Department of Computer and Information Science, CUNY Brooklyn College  
2900 Bedford Avenue, Brooklyn, NY 11210

## ABSTRACT

The problem of distributing multimedia securely over the Internet is often viewed as an instance of secure multicast communication, in which multicast messages are protected by a group key shared among the group of clients. One important class of key management schemes makes use of a hierarchical key distribution tree. Constructing a hierarchical tree based on secret shares rather than keys yields a scheme that is both more flexible and provably secure. Both the key-based and share-based hierarchical key distribution tree techniques are designed for managing keys for a single data stream. Recent work shows how redundancies that arise when this scheme is extended to multi-stream (e.g. scalable video) applications may be exploited in the key-based system by viewing the set of clients as a “multi-group”.

In this paper, we present results from an adaptation of a multi-group key management scheme using threshold cryptography. We describe how the multi-group scheme is adapted to work with secret shares, and compare this scheme with a naïve multi-stream key-management solution by measuring performance across several critical parameters, including tree degree, multi-group size, and number of shares stored at each node.

**Keywords:** multicast security, group communications, secret sharing, key management, set graphs, encryption, scalable video.

## 1. INTRODUCTION

Multicast communication is an efficient method for delivery of multimedia data to a large group of recipients. Providing optimized network resources, it can be used for many emerging services on the Internet. Most of the commercial applications that benefit from multicast, however, require a secure communication system in which data confidentiality is achieved. Examples of applications where only authorized users have access to data include teleconferencing, pay TV, distance education, and real-time information update. Security of multicast communications poses an important challenge in key management [1,2,3]. Due to the lack of access control in the open Internet architecture, enforcing secrecy in group communication requires a major tool - encryption. A common symmetric group key is shared by all the group members so that only registered senders can send packets to the group and only registered receivers can receive packets from the group. Management of the group key has received significant attention in recent years, with a focus on scalable schemes. Important attributes for a key management system are:

- backward access control (i.e., preventing a new member from having access to past communications),
- forward access control (i.e., preventing a leaving member from having access to future communications),
- minimizing participants’ storage, communication and computation requirements.

Classification of the schemes proposed for scalable secure multicasting in the last two decades can be found in the current literature [3,4,5,6,7].

An efficient group of proposals for one-to-many applications, where data is multicast from a single source to multiple receivers, is based on hierarchical key distribution trees [8]. In this approach, called the *Centralized Tree-Based Key Management (CTKM)*, a unique key is assigned to each of the nodes in a member’s path to the root. The group manager

---

\* dexter@sci.brooklyn.cuny.edu

† belostotskiy@att.net

‡ eskicioglu@sci.brooklyn.cuny.edu

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 a member 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 group manager 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. The CKTM schemes have been proposed in three independent publications [9,10,11].

Hierarchical trees are also used in the *Centralized Key Management with Secret Sharing (CKMSS)* [12,13] where the group manager assigns unique secret shares (instead of unique keys) to the nodes in the hierarchical tree. This allows the reconstruction of different keys by communicating different activating shares for the same prepositioned information kept at the nodes of the tree. Once the group key is established, it is used until a member joins/leaves the multicast group or periodic rekeying occurs. For a given node in the tree, the activating share and the shares assigned to the node define a unique polynomial. This polynomial defines an encryption key different from those derived for the other nodes. For each rekeying, the group manager multicasts an activating share to enable the members to compute their fresh keys. These key are then used to encrypt the new shares needed for backward access control, forward access control, or periodic group key change.

Both the key-based and share-based hierarchical distribution tree techniques are designed for managing keys for a single data stream. Naïve extensions of these techniques to scalable video applications, such as using independent trees for each layer or managing the keys for only the base layer (abandoning forward and backward access control for the enhancement layers) [14], do not take advantage of redundancies in the composition of the groups corresponding to each layer. For example, clients with access to the highest layer of enhancement must also have access to all other layers, permitting some efficiency in key management across layers. Recent work [15] shows how these redundancies may be exploited in the key-based system by viewing the set of clients as a “*multi-group*”. Here, clients are partitioned into groups according to which subset of layers they are permitted to access, with each group forming a hierarchical tree as in the single-stream model. These groups are then combined into an overlapping tree-like structure that eliminates some redundancies in the key management process.

In this paper, we will discuss an adaptation of the multi-group key management scheme to use secret shares. In the following sections, we briefly introduce the theoretical underpinnings of this work, describe the protocols supporting our key management scheme, and present some simulation results demonstrating the utility of the scheme.

## 2. BACKGROUND

In [15], Sun and Liu observe that multimedia applications often entail broadcasting different sets of objects and/or layers to different groups of users. Each of these streams may need to be protected independently, which necessitates extending the key management infrastructure for a single stream to support the efficient protection of several independent (yet related) data streams.

The naïve extension is simply to maintain a key management tree for each stream to be protected. Sun and Liu make the further observation, however, that this may generate much redundancy when individual users subscribe to several data streams simultaneously, as the operations performed to update the tree are essentially duplicated for each stream. They propose to combine the separate trees for each subscription into a multi-group graph structure in which the server maintains all key material.

In the above work, a *Data Group (DG)* defines a set of users who receive the same single data stream, and a *Service Group (SG)* defines a set of users who receive the same set of layers. A typical example is a multicast scalable video service where the video is encoded into 3 layers: Base Layer (BL), Enhancement Layer 1 (EL1), and Enhancement Layer 2 (EL2). This encoding allows three multicast streams to be assigned to three DGs. The users purchasing the movie at the basic quality level belong to the BL DG, the users purchasing the movie at the mid-quality level belong to BL and EL1 DGs, and the users purchasing the movie at the best quality level belong to BL, EL1, and EL2 DGs. The users who pay for the same quality level belong to the same SG.  $D_1, D_2, \dots, D_M$  denote DGs, where  $M$  is the total number of DGs. SGs are represented by  $S_t$ , where  $t = [i_1^t, i_2^t, \dots, i_M^t]^T$  is a binary vector having  $M$  elements with

$i_j^t = 0$  or  $1$ ,  $\prod_{j=1}^M i_j^t \neq 0$ , and  $S_t = \{D_1, i_1^t\} \cap \{D_2, i_2^t\} \cap \dots \cap \{D_M, i_M^t\}$ , where  $\{D_j, 0\} = \overline{D_j}$  and  $\{D_j, 1\} = D_j$ . A simple example with this notation is given in Figure 1.

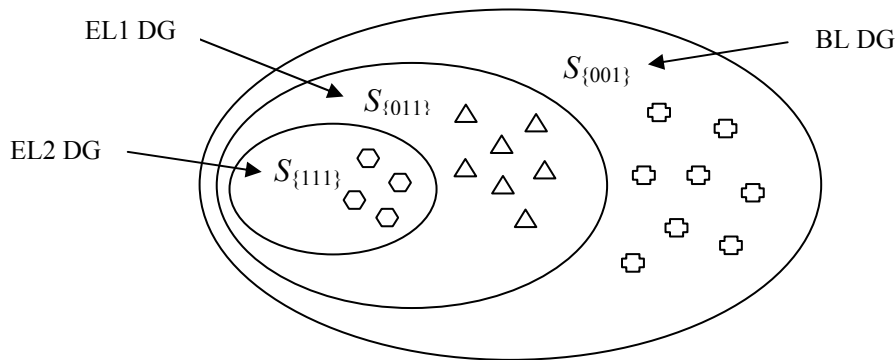


Figure 1. Data Groups and Service Groups in a multilayer multicast service

Figure 2 shows the key management in the traditional CKTM scheme in which a separate key tree is constructed for each DG. In this example, customers purchasing the movie at the EL2 level become members not only of the EL2 DG tree but also of the EL1 DG and BL DG trees, and customers purchasing the movie at the EL1 level are also members of the BL DG tree. Although independent trees are easier to implement, they introduce a substantial overhead due to the overlapping membership in different DGs. When a user decides to leave the service, all the DGs he belongs to must change the relevant keys. If a user switches from  $S_{t1}$  to  $S_{t2}$ , all DGs which include users in  $S_{t1}$  but not in  $S_{t2}$  need to perform rekeying.

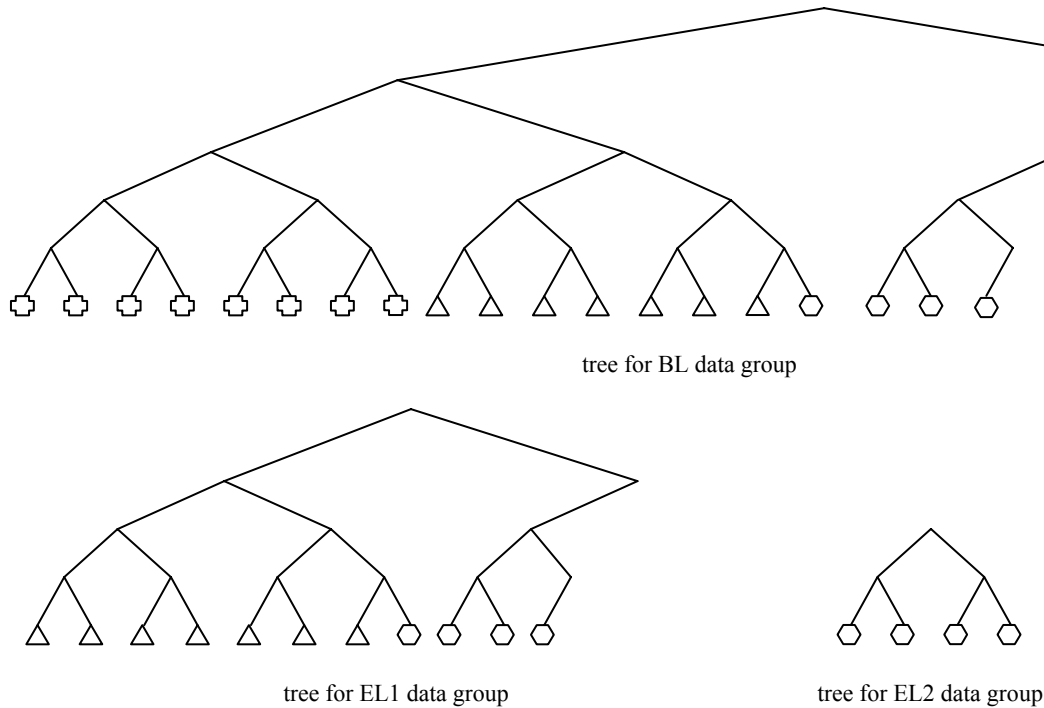


Figure 2. Traditional independent tree key management scheme for three layers

In the Sun and Liu approach, the independent trees are integrated into one key graph for the management of keys of the entire user population. The construction of the graph is shown in Figure 3. Separate trees are connected to form the key graph in 3 steps:

1. For each DG  $D_j$ , one session key  $K_s^j$  and one key encryption key (KEK)  $K_\epsilon^j$  are generated.
2. For the users in each SG  $S_t$ , a subtree with the root node having  $K_t$  is constructed.
3. For each DG  $D_j$ , a subtree whose root is  $K_\epsilon^j$  and whose leaves are  $\{K_t, i_j^t = 1\}$  is constructed.

Only two cases are considered for key management: A user switching from SG  $S_{t1}$  and SG  $S_{t2}$ , and a user in SG  $S_t$  leaving the service. When a user joins the service, the keys are updated without any rekeying as suggested in [16]. The rekeying process in the traditional CKTM scheme is generalized by updating the keys in  $\phi_1 \cap \bar{\phi}_2$ , where  $\phi_1$  denotes the set of keys owned by the user in his previous position, and  $\phi_2$  denotes the set of keys owned by the user in his new position.

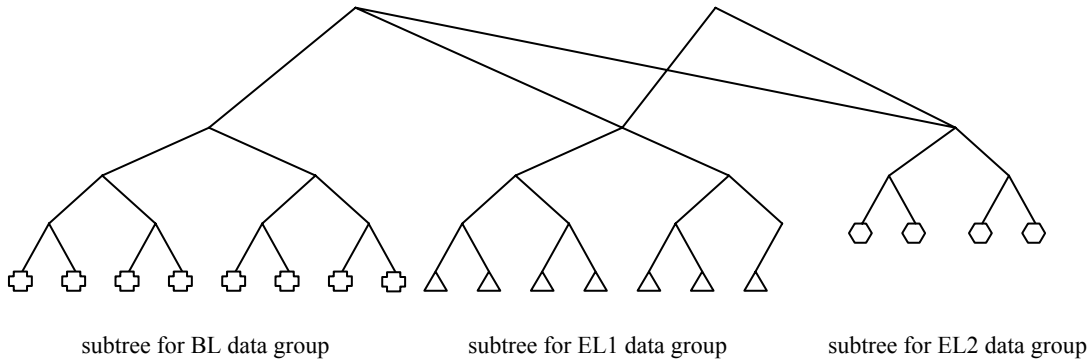


Figure 3. Integrated graph key management scheme for three layers

### 3. MULTI-GROUP KEY MANAGEMENT WITH SECRET SHARING (MGKMSS) SCHEME

We now describe our Multi-Group Key Management with Secret Sharing (MGKMSS) scheme using threshold cryptography. The essential difference between MGKMSS and the Sun and Liu scheme is that each node key is replaced by a set of shares. The assignment of unique shares to the nodes allows the generation of new node keys every time a different activating share is multicast.

In a  $(t, n)$  threshold scheme ( $t \leq n$ ), a trusted dealer divides a secret  $s$  into  $n$  secret shares  $s_i$  ( $1 \leq i \leq n$ ) in such a way that at least  $t$  shares are required to reconstruct  $s$ . It is assumed that each  $s_i$  is securely distributed to user  $P_i$  and stored as confidential information. A perfect threshold scheme is a threshold scheme in which a knowledge of  $(t-1)$  or fewer shares does not provide any advantage to the opponent to find the secret. In Shamir's  $(t, n)$  threshold scheme [17], the secret  $s$  is defined to be the coefficient  $a_0$  of a random  $(t-1)$ -degree polynomial  $f(x) = (a_{t-1}x^{t-1} + \dots + a_1x + a_0) \text{ mod } p$  over the finite Galois Field  $\text{GF}(p)$ . The trusted dealer divides the secret among  $n$  users as follows:

1. Choose a prime  $p$  larger than  $n$  and the secret  $s$ .
2. Construct  $f(x)$  by selecting  $(t-1)$  random coefficients  $a_i$  ( $1 \leq i \leq t-1$ ).
3. Compute the shares  $s_i$  by evaluating  $f(x)$  at  $n$  distinct points.
4. Securely distribute  $s_i$  to user  $P_i$  ( $1 \leq i \leq n$ ).

The secret  $s$  can be recovered by constructing the polynomial  $f(x) = \sum_{i=0}^{t-1} y_i \prod_{0 \leq j \leq t-1, j \neq i} (x - x_j)/(x_i - x_j)$  from any  $t$  of the  $n$  shares, and computing  $f(0)$ .

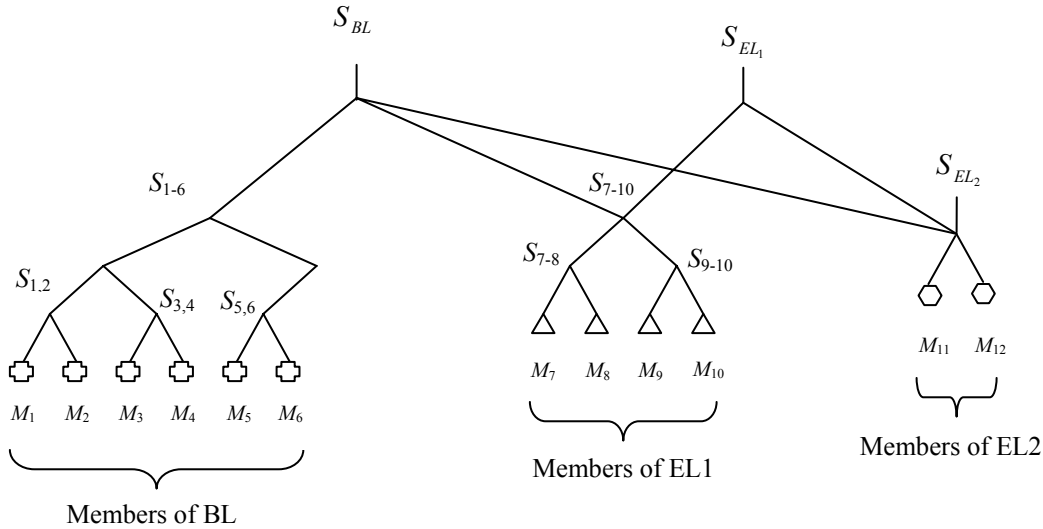
In a *prepositioned secret sharing scheme*, the secret shares are stored by the participants in advance of the activation of the scheme [18,19]. Even if all the pieces are exposed, the secret key cannot be recovered until some additional information is provided. The basic idea is to preposition the secret pieces of information, and allow an activating piece to recover a unique shared secret. In our implementation, the scheme is designed to recover the secret by requiring only one more piece (i.e., the “activating” share).

As the MGKMSS scheme is an extension of the CKMSS scheme, we retain two features:

1. A session key is not needed. The keys generated at the root nodes are used at the top of the key hierarchy.
2. Rekeying is performed when a new member joins the service. We believe that updating the keys locally introduces a security weakness. As a corollary of this, when a member changes service level, rather than rekeying  $\phi_1 \cap \overline{\phi_2}$ , we rekey all nodes in the symmetric difference of  $\phi_1$  and  $\phi_2$ .

The following example illustrates how new keys are obtained based on the new activating share multicast to the members of a group.

Example: Assume we have the tree structure given below. We will consider two cases: (i) EL2 member  $M_{11}$  leaving the service, and (ii) EL1 member  $M_{10}$  switching to the best quality service. The acronym “AS” stands for the activating share, and the single quotation denotes a new share or a new key.



Case (i): The following messages are sent to the users:

AS,  $\{S'_{BL}\}_{K_{1-6}}, \{S'_{BL}\}_{K_{7-10}}, \{S'_{BL}\}_{K'_{EL_2}}, \{S'_{EL_1}\}_{K_{7-10}}, \{S'_{EL_1}\}_{K'_{EL_2}}, \{S'_{EL_2}\}_{K_{12}}$ , where the fresh encryption keys are obtained using the activating share and the respective shares.

Case (ii): The following messages are sent to the users:

AS,  $\{S'_{7-10}\}_{K_{7-8}}, \{S'_{7-10}\}_{K'_{9-10}}, \{S'_{9-10}\}_{K_9}, \{S'_{EL_2}\}_{K_{EL_2}}$ , where the fresh encryption keys are obtained using the activating share and the respective shares.

Additionally, the following message is unicast to  $M_{10}$  (who is not able to decrypt the above messages):

AS,  $\{S'_{EL_2}\}_{K_{10}}$ , where  $K_{10}$  is the key obtained using the activating share and the member's individual share.

#### 4. SYSTEM SIMULATION

We performed a number of experiments to evaluate the performance of the MGKMSS scheme. The simulation code was developed in C++, and run on a multi-user Solaris 8 environment on a Sun Blade 100 workstation. A public-domain implementation of the 128-bit AES cipher by Szymon Stefanek [20] and Victor Shoup's Library for Number Theory (NTL) [21] were used in the simulation.

We simulated the behavior of the MGKMSS scheme for protecting 3 layers of scalable video (i.e., a base layer and two enhancement layers), and compared the results to the behavior of a key management scheme based on three independent trees (one for each layer).

In Shamir's scheme, we defined the shares to have the format  $(x, y) = (i, S_i)$ , where  $i$  is not a public index but a part of the secret share. The  $x$  coordinate is a 4-byte value and the  $y$  coordinate is a 16-byte value, making the total size of a share 20 bytes. The secret created using these shares is a 128-bit value.

The three design parameters are the following:

1. Tree degree
2. Initial group size
3. Number of shares assigned to each node

Each of these three parameters was allowed to vary as the others were kept constant. Table 3 shows the range of values and the constants for the parameters. Both the average processing times and the average message sizes were measured. Processing a request involves first locating the leaf corresponding to the joining/leaving member, then traversing the path from this leaf to the root. At each step, the activating share is combined with the shares stored at the node to produce an encryption key; this key is in turn used to encrypt newly generated random shares. The contents of the nodes are also updated to contain these new random shares.

Table 3. Parameter values

Parameters	Range of values	Constant
Tree degree	2, 3, 4, ..., 10	4
Initial group size	$2^5, 2^6, 2^7, \dots, 2^{14}$	1024
Number of shares per node	2, 3, ..., 8	2

In the experiments, the group manager first builds the initial graph using the given number of join requests. It then updates the tree in response to 5000 subsequent "join," "leave," and "change-service-group" requests. These requests are randomly generated to have a uniform distribution of "arrivals" and "departures" from each service group (where an arrival may be due to a brand new subscription or simply an upgrade to a new level of service, and similarly for departures).

Below, we discuss the effect of these parameter changes on average processing time and multicast message length for the join, leave, and change-service-group operations. The cost of the periodic rekeying operation is constant across all parameter values, as the server workload for this operation is minimal: only the activating share is multicast, there is no encryption cost, and the message size is constant at 20 bytes. (Note the contrast with key-based schemes where, for a tree of degree  $d$ ,  $d$  encrypted messages need to be multicast.)

### 4.1 Tree Degree

Figures 4 and 5 illustrate the effect of changing the tree degree in the two schemes (i.e., MGKMSS and independent trees). Note that as the degree of the tree is increased, the processing time per join goes down and the processing time per leave goes up (at a higher rate). As we would expect, processing time for the set graph structure is generally lower than that for the independent tree structure. Further, processing time for set graph operations is more uniform than that for the independent tree: while processing time in the independent tree depends on the “distance” a customer moves when changing his subscription (leaving layer 2 or layer 3 is significantly more expensive than leaving 1 layer), processing time in the set graph depends only the type of operation (joins are cheap, leaves are slightly more expensive, and changes involving a composition of joins and leaves are a bit more expensive still). Message lengths exhibit a similar relationship.

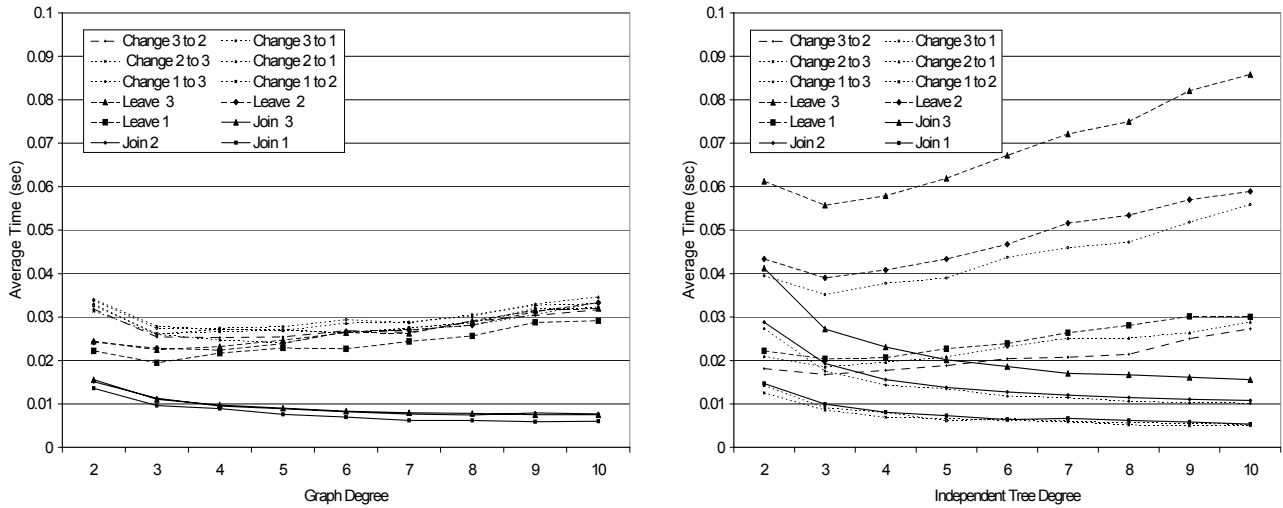


Figure 4. Processing time versus degree

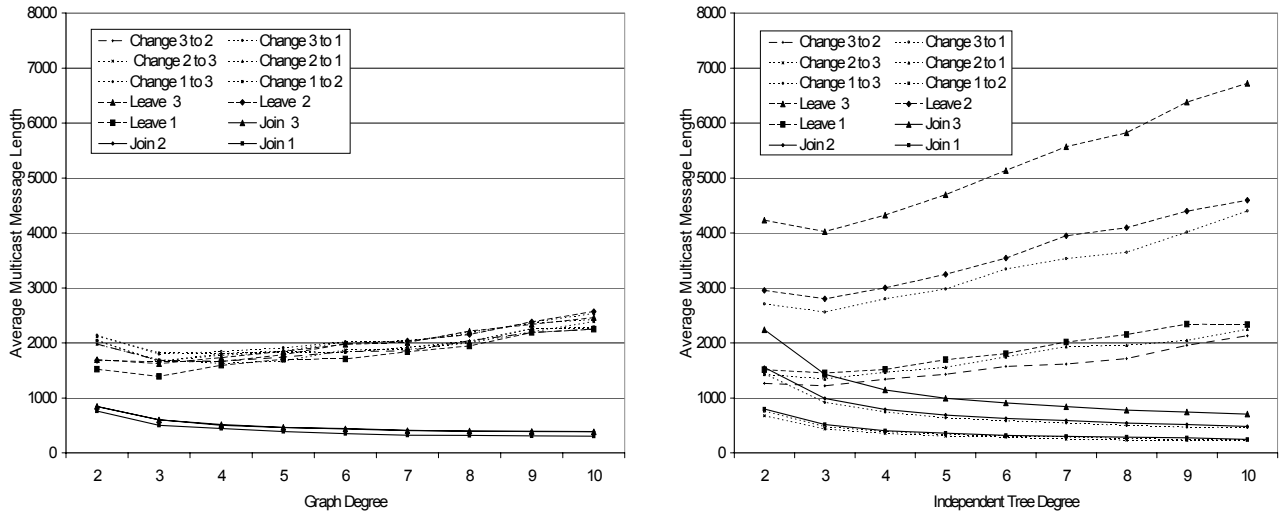


Figure 5. Multicast message length versus degree

## 4.2 Initial Group Size

Figures 6 and 7 show how the processing time and the message size vary with the initial group size. Note that the horizontal axis in these figures is log scale. Both schemes grow in logarithmic fashion, but the MGKMSS scheme clearly has superior performance.

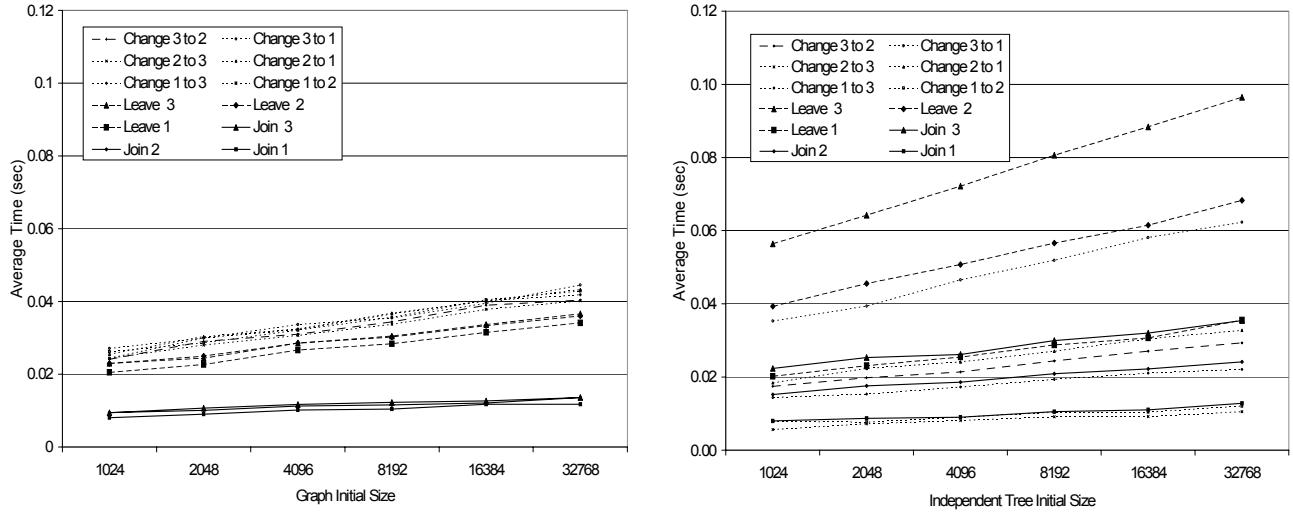


Figure 6. Processing time versus initial group size

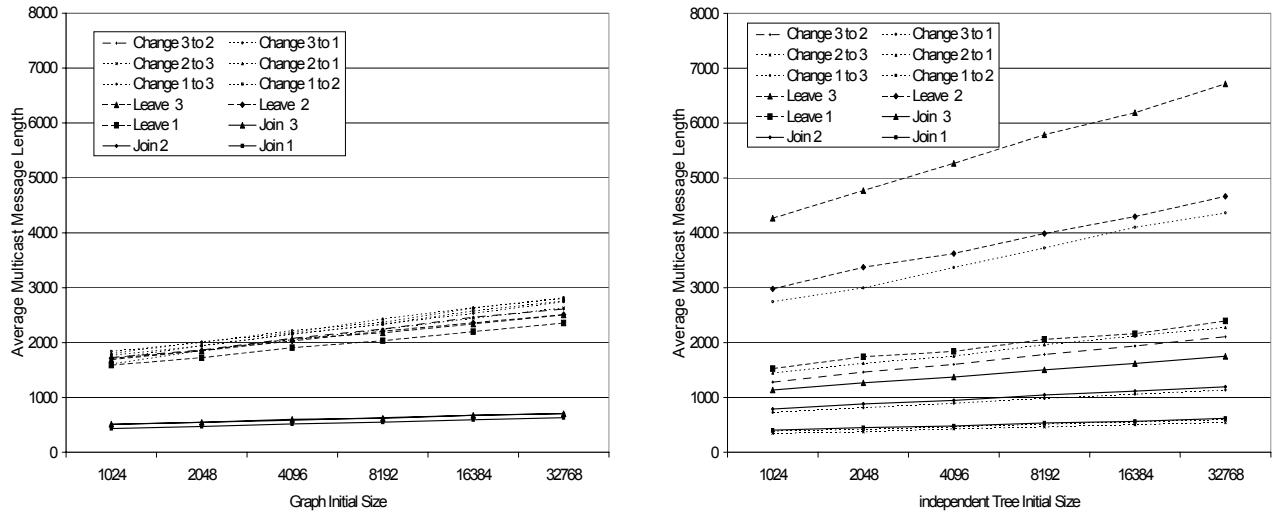


Figure 7. Multicast message length versus initial group size

### 4.3 Number of Shares per Node

Figures 8 and 9 show how the processing time and the message size vary with the number of shares per node. Both schemes are slightly superlinear, though again the MGKMSS scheme exhibits slower growth. This level of superlinear growth is acceptable, however, as it contributes directly to increased security.

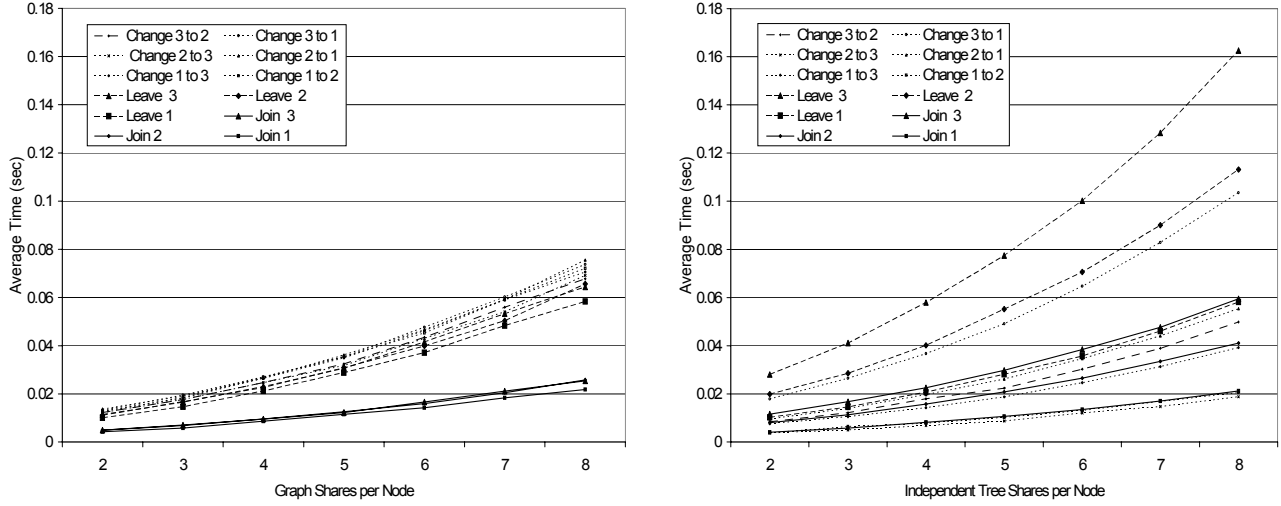


Figure 8. Processing time versus number of shares per node

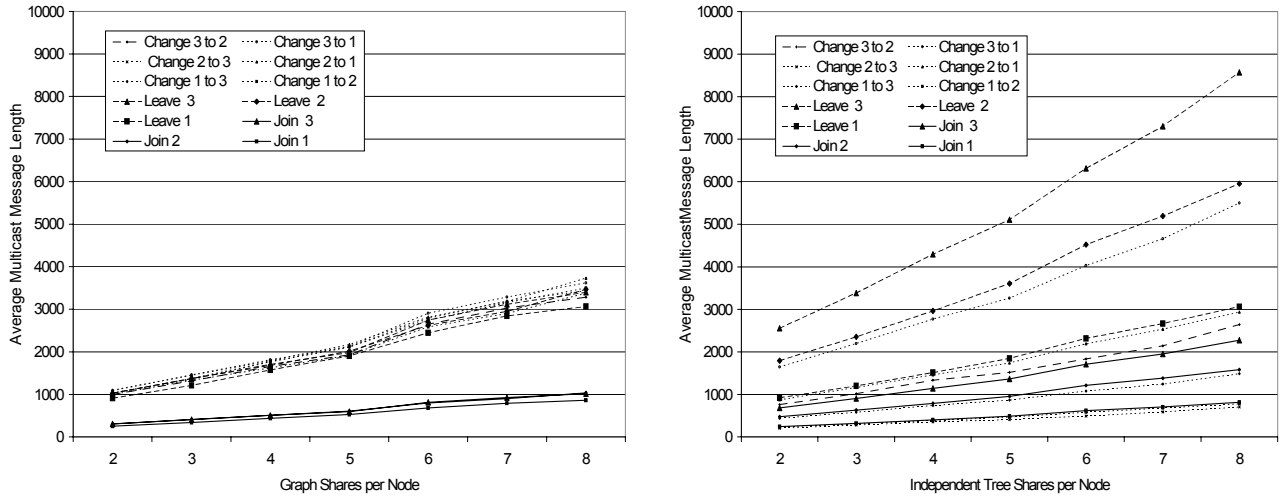


Figure 9. Multicast message length versus number of shares per node

## 5. CONCLUSIONS

We presented a new multi-group key management scheme (MGKMSS) using threshold cryptography for multicast services that distribute data in multiple layers (e.g., scalable video). Instead of storing keys at the nodes of the integrated key graph, the group manager assigns unique secret shares that are used for constructing different keys for different activating shares.

The behavior of the proposed scheme was compared with the naïve approach of employing independent trees for each of the three layers. In the simulation, three critical parameters were used: tree degree, initial group size, and number of shares assigned to each node. We observed that the MGKMSS scheme clearly outperforms the naïve approach in all cases. Specifically, the cost associated with processing time and message length in the graph scheme primarily depends only on the type of operation being performed rather than on which data groups are involved. Joins are the cheapest, leaves are more expensive, and moves across service groups have the highest cost.

Ideally, a multimedia protection system should be scalable in the sense that an incremental improvement in security imposes only an incremental increase in complexity. Although both schemes compared in this paper have a deviation from linear performance when the number of shares per node is increased, this superlinearity can be justified. Additional shares undoubtedly lead to an increase in the security level.

For three layers of video, the average amount of savings in processing time and multicast message length for the integrated key graph was computed to be around 20%. As the number of layers is increased, the benefits of the scheme would be higher.

## REFERENCES

- [1] 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.
- [2] T. Hardjono and G. Tsudik, "IP Multicast Security: Issues and Directions," *Annales de Telecom*, pp. 324-334, July-August 2000.
- [3] A. M. Eskicioglu, "Multimedia Security in Group Communications: Recent Progress in Key Management, Authentication, and Watermarking," *ACM Multimedia Systems Journal*, Special Issue on Multimedia Security, pp. 239-248, September 2003.
- [4] L. R. Dondeti, S. Mukherjee and A. Samal, "Survey and Comparison of Secure Group Communication Protocols," Technical Report, University of Nebraska-Lincoln, June 1999.
- [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, 7(6), pp. 503-511, 2000.
- [6] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, 35(3), September 2003, pp. 309-329.
- [7] K.-C. Chan and S.-H. G. Chan, "Key Management Approaches to Offer Data Confidentiality for Secure Multicast," *IEEE Network*, pp. 30-39, September/October 2003.
- [8] 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.
- [9] D. Wallner, E. Harder and R. Agee, "Key Management for Multicast: Issues and Architectures" RFC 2627, June 1999.
- [10] 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.
- [11] 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.

- [12] A. M. Eskicioglu and M. R. Eskicioglu, "Multicast Security Using Key Graphs and Secret Sharing," *Proceedings of the Joint International Conference on Wireless LANs and Home Networks and Networking*, pp. 228-241, Atlanta, GA, August 26-29, 2002.
- [13] A. M. Eskicioglu, S. Dexter, and E. J. Delp. "Protection of multicast scalable video by secret sharing: simulation results," *Proceedings of SPIE Security and Watermarking of Multimedia Content V*, Vol. 5020, pp. 505-515, Santa Clara, CA, January 21-24, 2003.
- [14] T. Kunkelmann and U. Horn, "Partial Video Encryption Based on Scalable Coding," *5th International Workshop on Systems, Signals and Image Processing*, Zagreb, Croatia, June 1998.
- [15] Y. Sun, and K.J. R. Liu, "Multi-layer Management for Secure Multimedia Multicast Communications," *Proc. IEEE International conferences on Multimedia and Expo (ICME03)*, Vol. II, pp 205-208, Baltimore, MD, July 2003.
- [16] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, B. Plattner, "The VersaKey Framework: Versatile Group Key Management," *Journal on Selected Areas in Communications: Special Issue on Middleware*, **17**(9), pp. 1614-1631, 1999.
- [17] A. Shamir, How to share a secret, *CACM*, **22**(11), pp. 612-613, November 1979.
- [18] G. J. Simmons, How to (really) share a secret, *Advances in Cryptology – CRYPTO '88 Proceedings*, pp. 390-448, Springer-Verlag, 1990.
- [19] G. J. Simmons, Prepositioned shared secret and/or shared control schemes, *Advances in Cryptology – EUROCRYPT '89*, pp. 436-467, Springer-Verlag, 1990.
- [20] <http://www.esat.kuleuven.ac.be/~rijmen/rijndael/>
- [21] <http://shoup.net/ntl>