MULTI-LAYER KEY MANAGEMENT FOR SECURE MULTIMEDIA MULTICAST COMMUNICATIONS

Yan Sun and K. J. Ray Liu

Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742

ABSTRACT

An important application of multicasting is group-oriented multimedia communication, such as video-on-demand and video conferencing. Before these services can be successfully deployed, security infrastructures must be developed to ensure access control to multicast content. In this paper, we present a multi-group key management scheme tailored to multimedia multicast services that distribute data in multi-layer or multi-object format. Compared with the existing tree-based key management schemes that are used for generic data multicasting services, the proposed scheme can reduce the communication overhead by a factor of $\frac{M+1}{2}$ and M is the number of layers. For example, when the multimedia data is encoded into three layers according to different quality requirements, the proposed scheme reduces the communication overhead associated with key updating by 50%.

1. INTRODUCTION

Soon, many multimedia applications will involve group scenarios, where users work and play together. Multicast communication, reducing demands on network and bandwidth resources by transmitting a single data stream to a selected set of receivers, is particular beneficial for group-oriented multimedia applications such as pay-per-view broadcast of sport events, video conferencing and communal gaming [1] [2].

Before these group-oriented multimedia multicast applications can be successfully deployed, *access control* mechanism must be developed such that only authorized users can access the group communication [3]. Access control is usually achieved by encrypting the content using an encryption key, known as the session key (SK) that is shared by all legitimate group members. Since the group membership will most likely be dynamic with users joining and leaving the services, it is necessary to change the SK in order to prevent the leaving user from accessing future communication and prevent the joining user from accessing prior communication [3]. This key updating process is usually referred to as *Key Management*.

In a typical multicast key management scheme, a trusted third party, known as the key distribution center (KDC), is responsible for securely communicating new key materials to the group members. Besides the session key, the KDC shares auxiliary keys, known as key encrypting keys (KEKs), which are used solely for the purpose of updating the session key and other KEKs. In addition, each user has a private key that is only known by himself and the KDC. A popular class of multicast key management schemes employ a tree hierarchy for the maintenance of keying material, and have small usage of communication, computation and storage resources [3, 4]. These key management schemes, however, are designed for generic data of a single data stream and are not efficient for many multimedia applications that distribute data in multi-layer or multi-object format [5].

Multi-layered or multi-object services, as is prevalent in multimedia applications, consist of users that subscribe to different objects or layers, or possibly multiple of them. For example, in an HDTV broadcasts, users with a normal TV receiver can receive the normal format, while other users with an HDTV receiver can receive both the normal format and the extra information needed to achieve HDTV resolution. As another example, the MPEG-4 standard allows for the composition of multiple media streams corresponding to different object planes [5]. Since traditional multicast key management schemes are not designed to handle the key management issues associated with multiple services occurring concurrently that have correlated memberships, they makes inefficient use of keys and does not scale well when there are many objects or layers.

In this paper, we will design a *multi-group key management* scheme that exploits the overlap in the memberships of the different objects or services, while incorporating new functionalities that are not present in conventional multicast key management. In the rest of this paper, the key management problem in multi-layer/object multimedia services is formulated in Section 2. The solution based on tradition key management schemes is provided in Section 3, and our multi-group key management scheme is described in Section 4. The comparison between these two schemes are presented in Section 5, followed by the conclusion in Section 6.

2. KEY MANAGEMENT IN MULTI-LAYER/OBJECT MULTIMEDIA MULTICAST SERVICES

In this section, we introduce notations describing multi-layer/object multimedia services, and formulate the performance metrics measuring communication overhead associated with key management schemes.

We define *Date Group* (DG) as the set of users who receive the same single data stream, and *Service Group* (SG) as the set of users who receive the same set of layers or objects. For example, in a movie multicast services where data is encoded into Base Layer (BL), Enhancement Layer 1 (EL1) and Enhancement Layer 2 (EL2) [5], there exist three multicast data streams that corresponds to three DGs. Particularly, BL DG, EL1 DG and EL2 DG contain the users who receive BL, EL1 and EL2, respectively. As depicted in Figure 1(a), users may subscribe to different quality levels. The users subscribing to the highest quality level join

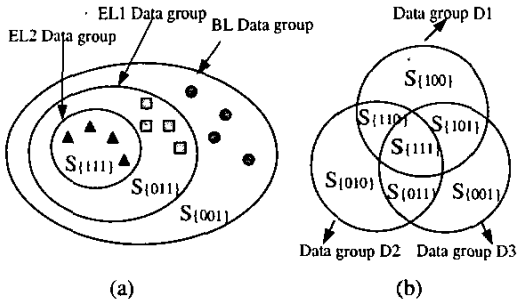

Fig. 1. Data groups and services groups in (a)multi-layer (b)multi-object multicast services

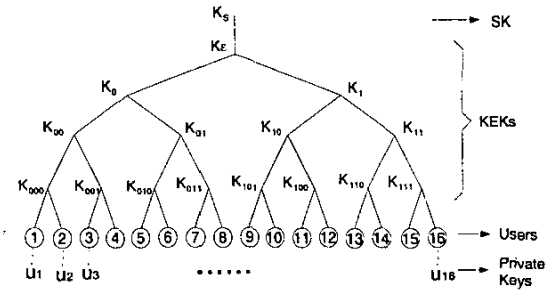

Fig. 2. A typical key management tree

all three DGs; the users subscribing to the moderate quality level join BL and EL1 DGs; and the users subscribing to the basic quality level only joins the BL DG. The users subscribing to the same quality level belong to the same SG.

In general, we use $D_1, D_2, \cdots, D_M$ to denote DGs and M is the total number of DGs. SGs are described by S_t, where t is a M by 1 non-zero binary vector, i.e. $t = [i_1^t, i_2^t, \cdots, i_M^t]^T$, i_j^t= 0 or 1 and $\prod_{j=1}^{M} i_j^t \neq 0$, and

$$S_t = \{D_1, i_1^t\} \bigcap \{D_2, i_2^t\} \bigcap \cdots \bigcap \{D_M, i_M^t\}$$

where $\{D_j, 0\} = \overline{D}_j$ and $\{D_j, 1\} = D_j$. These notations are illustrated in Figure 1(a) and (b). In addition, $n(S_t)$ is defined as the number of users in the SG S_t. Then, the number of users in D_j is $n(D_j) = \sum_t i_j^t \cdot n(S_t)$.

In multi-layer/object scenario, users may switch between SGs by dropping or subscripting some layers or objects. We introduce the notation, $S_{t1} \rightarrow S_{t2}$, to represent a user switching from SG S_{t1} to SG S_{t2}. Further, $S_t \rightarrow O$ represents the event that a user in SG S_t leaves the service.

The efficiency of the key management schemes is usually evaluated by communication, computation and storage overhead [3]. Among these performance criteria, communication overhead is particularly important. When group membership changes, new key information is transmitted to all users through *rekeying messages*. These rekeying messages must be delivered reliability and in a timely manner [6]. In applications where there are many users and frequent additions or deletions to the group membership, reducing communication overhead not only reduces the usage of bandwidth, but also improves the reliability of the services [6]. Therefore, the performance metric in this work is defined as the amount of rekeying messages when event X occurs, denoted by $C(X)$. Since key updating in the case of user joining can be achieved without sending any rekeying messages [4], we shall focus only on $C(S_{t1} \rightarrow S_{t2})$ and $C(S_t \rightarrow O)$.

3. TRADITIONAL KEY MANAGEMENT SCHEMES IN MULTIMEDIA SERVICE

A typical key tree used in traditional key management schemes [3, 7] is illustrated in Figure 2. The multicast data is encrypted using the session key K_s. Each user stores his private key u_i, the session key K_s, and a set of KEKs on the path from himself to the root of the key tree. In the example in Figure 2, user 16 possesses $\{u_{16}, K_s, K_\epsilon, K_1, K_{11}, K_{111}\}$. When he leaves the service, all his keys shall be updated. Let x^{old} denote the old version of key x, x^{new} denote the new version of key x, and $x\{y\}$ denote the key y encrypted by key x. Then, the key updating can be achieved by sending the following 8 rekeying messages [4, 7]:

- $u_{15}\{K_{111}^{new}\}$: user 15 acquires K_{111}^{new}.
- $K_{111}^{new}\{K_{11}^{new}\}$,$K_{110}^{old}\{K_{11}^{new}\}$: user 13,14,15 acquire K_{11}^{new}.
- $K_{11}^{new}\{K_1^{new}\}$,$K_{10}^{old}\{K_1^{new}\}$: user $9, \cdots, 15$ acquire K_1^{new}.
- $K_1^{new}\{K_\epsilon^{new}\}$,$K_0^{old}\{K_\epsilon^{new}\}$: user $1, \cdots, 15$ acquire K_ϵ^{new}.
- $K_\epsilon^{new}\{K_s^{new}\}$: all remaining users acquire K_s^{new}.

The above key updating procedure guarantees that all remaining users obtain the new session key and KEKs, while user 16 is unable to acquire the new keys. Each rekeying message has the same size as the length of the K_s and the $KEKs$.

Let d denote the degree of the tree and N denote the number of users on the key tree. We assume that the balanced tree structure are used [8]. Let $f(N)$ denote the amount rekeying message needed when one user leaves the service. It has been shown that $\lim_{N->\infty} f(N) = d \cdot \log_d(N)$ [3, 4], that is, the amount of rekeying message increases linearly with the logarithms of the group size.

Traditional key management schemes are designed for a single data stream. In order to manage access control for multiple-layer/objects multimedia services, a separate key tree must be constructed for each DG, as depicted in Figure 3. This approach shall be referred to as the *Independent-Trees* scheme.

When a user leaves the service, all the DGs that he has subscribed must update keys. When a user switches from S_{t1} to S_{t2}, the DGs, which is subscribed by the users in S_{t1} but not by the users in S_{t2}, should update keys. Using the notations introduced in Section 2, the communication overhead associated with the independent-tree schemes can be calculated as:

$$C^{ind}(S_t \rightarrow O) = \sum_{j=1}^{M} i_j^t f(n(D_j)), \quad (1)$$

$$C^{ind}(S_{t1} \rightarrow S_{t2}) = \sum_{j=1}^{M} \max(i_j^{t1} - i_j^{t2}, 0) \cdot f(n(D_j)). \quad (2)$$

The advantage of considering the separate keys for each DG lies primarily in the simplicity of implementation because it is a straightforward extension from the existing key management schemes. This scheme, however, does not exploit the special relations among the subscribers and makes inefficient use of keys due to the membership overlap among DGs. As an extreme example, if the leaving users having subscribed all layers/objects, key updating has to take place on all key trees.

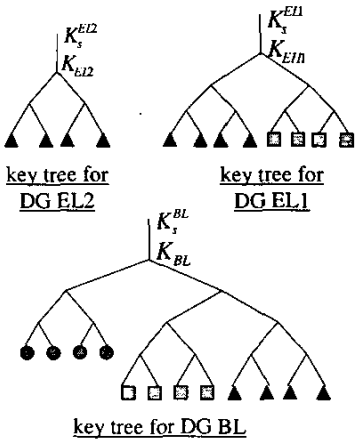

Fig. 3. Independent key management scheme for multimedia services containing three layers

4. MULTI-GROUP KEY MANAGEMENT SCHEME

In this section, we develop a multi-group key management scheme that employs one integrated key graph for maintaining key materials of all users. On either the key trees or the key graph, each node is associated with a key. In the rest of the paper, the same name will be used for the key and the node associated with this key.

The key graph is constructed by connecting separately designed subtrees in three steps.

Step1: For each DG D_j, generate one session key K_s^j and one KEK K_ϵ^j.

Step2: For the users in each SG S_t, construct a subtree with the root node denoted by K_t. These subtrees shall be called as the *service-group-subtrees*.

Step3: For each DG D_j, construct a subtree whose root is K_ϵ^j and whose leaves are $\{K_t, i_j^t = 1\}$.

This 3-step procedure is illustrated in Figure 4 for the services containing 3 layers and having 4 users in each SG.

The multi-group key graph can also be interpreted as M overlapped key trees, each of which has K_s^j as the root and the users in DG D_j as the leaves. Obviously, these M key trees can be used in the independent-tree scheme. This reveals the fact that the multi-group key graph removes the "redundancy" presented in independent-tree schemes. Therefore, it has the potential to reduce the communication overhead.

The rekey protocol of removing a user from a key graph has been presented in [7]. Briefly speaking, when a user leaves the service, all keys in the *keyset* of this user shall be updated from bottom to up by using their children node keys. As defined in [7], *keyset* refers to the set of keys associated with a edge node on the key graph and possessed by the user located at this edge node. In the case of user switching from SG S_{t1} to S_{t2}, this user shall be moved from the service-group-subtree of S_{t1} to a new location on the service-group-subtree of S_{t2}. We generalize the rekeying protocol in [7] as:

- Let ϕ_1 denote the keyset associated with the user's previous position, and ϕ_2 denote the keyset associated with the user's new position. Then, the keys in $\phi_1 \cap \overline{\phi_2}$ shall be updated from bottom to up by using their children node keys.

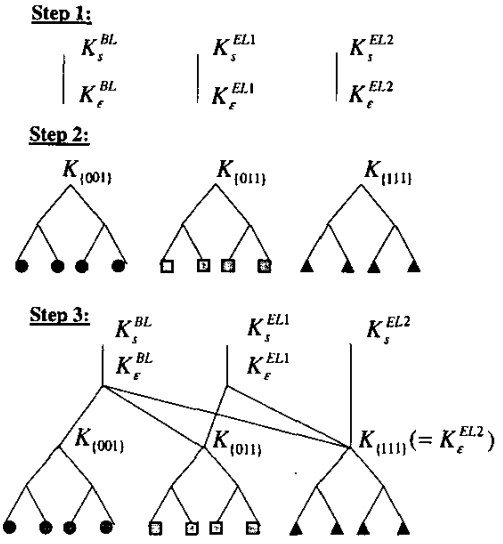

Fig. 4. Multi-group key management graph construction

To simplify the analysis, we assume that all subtrees are balanced trees with degree $d = 2$. We calculate the communication overhead of multi-group key management when one user leaves, as:

$$C^{mg}(S_t \rightarrow O) = f(n(S_t)) + \sum_{j=1}^{M} i_j^t f(c_j), \qquad (3)$$

where c_j is the number of SGs that receive D_j, and $c_j = \sum_t i_j^t$. We also calculate the upper bound of the communication overhead when users switch between SGs, as:

$$C^{mg}(S_{t1} \rightarrow S_{t2}) \leq f(n(S_{t1})) - 1 + \sum_{j=1}^{M} \max(i_j^{t1} - i_j^{t2}, 0) f(c_j).$$

5. PERFORMANCE COMPARISON

In this section, we compare the performance of the traditional independent-tree key management schemes and the proposed multi-group key management schemes in two special scenarios: the multi-layer services depicted in Figure 1(a) and the multi-object services depicted in Figure 1(b). In both cases, we assume that SGs contain the same amount of users, denoted by N.

Users leave the service

Figure 5 shows the communication overhead in the multi-layer scenario. The left plot is for the independent-tree scheme, and the right plot is for the multi-group scheme. It is observed that the proposed scheme can greatly reduce the communication overhead, especially when a user subscribing many DGs leaves the service. Particularly, when one user leaves the SG associated with the highest quality and $N = 100$, the proposed scheme reduces the communication overhead by 55%. In addition, the advantages of the propose scheme become larger when the system contains more users. In this special scenario, we can prove that:

$$\lim_{N \rightarrow \infty} \frac{\sum_t C^{ind}(S_t \rightarrow O)}{\sum_t C^{mg}(S_t \rightarrow O)} = \frac{M+1}{2}. \qquad (4)$$

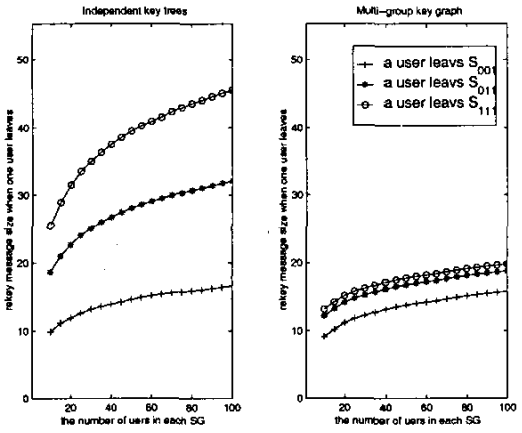

Fig. 5. Performance Comparison in Multi-layer services

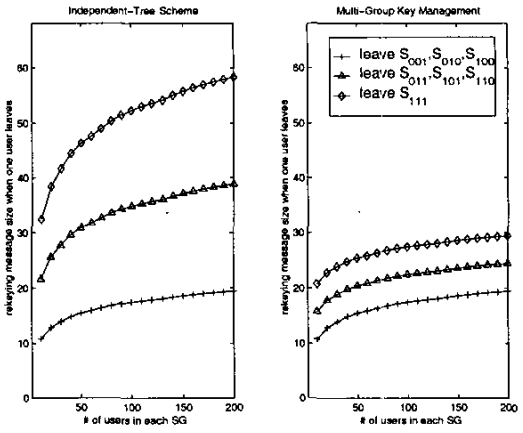

Fig. 6. Performance Comparison in Multi-object services

That is, when the system contains a large number of users, the average performance gain will be larger when the system contains more layers.

The performance of two key management schemes in multi-object scenario is shown in Figure 6, which leads to the similar observations as in the multi-layer scenario. In this case, we can prove that:

$$\lim_{N\to\infty} \frac{\sum_{t1}\sum_{t2} C^{ind}(S_{t1} \to S_{t2})}{\sum_{t1}\sum_{t2} C^{mg}(S_{t1} \to S_{t2})} = \frac{M}{2}. \qquad (5)$$

It should be noticed that the assumption that each SG contains the same amount of users is not always realistic in the multi-object case. The results in (5) only provides the insight of the asymptotical properties of the performance gain.

Users switch between service groups

Communication overhead of user switching between SGs is shown in Table 1 for the multi-layer scenario. When users switch from low quality level to high quality levels (the first three rows), no rekeying messages need to be sent when using the independence-tree scheme. In this case, the multi-group key graph scheme is less efficient because users have to be relocated on the key graph. When users switch to lower quality levels (the next three rows), multi-group key management scheme tends to be more efficient, which can be easily verified by the communication overhead derived in Section 4.

	Independent-tree	Multi-Group
$S_{001} \to S_{011}$	0	$f(N)-1=13$
$S_{011} \to S_{111}$	0	$f(N)-1=13$
$S_{001} \to S_{111}$	0	$f(N)-1=13$
$S_{111} \to S_{011}$	$f(N)=14$	$f(N)-1+f(1)=14$
$S_{011} \to S_{001}$	$f(2N)=16$	$f(N)-1+f(2)=16$
$S_{111} \to S_{001}$	$f(N)+f(2N)$ $=30$	$f(N)-1+f(1)$ $+f(2)=17$

Table 1. Communication overhead of user switching between SGs in multiple-layer scenario (M=3 and N=128)

6. CONCLUSION

In this paper, we presented a multi-group key management scheme for secure multimedia multicast applications that distribute data in multi-layer or multi-object format. We designed a 3-step key graph generation procedure, as well as a rekey protocol allowing users subscribing or dropping some layers/objects while remaining the subscription to others. Compared with the existing tree-based key management schemes that are designed for single data stream, the proposed scheme can reduce the communication overhead by a factor of $\frac{M+1}{2}$ in the multi-layer services and $\frac{M}{2}$ in the multi-object services. As an example, when the multimedia data is encoded into three layers according to different quality requirements, the proposed scheme reduces the communication overhead associated with key updating by 50%. For the systems containing more layers or objects, the advantage of the proposed scheme is even larger.

7. REFERENCES

[1] S. Paul, *Multicast on the Internet and its applications*, Kluwer Academic Publishers, 1998.

[2] W. Trappe, Jie Song, R. Poovendran, and K.J.R. Liu, "Key distribution for secure multimedia multicasts via data embedding," *Proc. IEEE ICASSP'01*, pp. 1449–1452, May 2001.

[3] M.J. Moyer, J.R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications," *IEEE Network*, vol. 13, no. 6, pp. 12–23, Nov.-Dec. 1999.

[4] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework:versatile group key management," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 17, no. 9, pp. 1614–1631, Sep. 1999.

[5] A. Puri and T. Chen, *Multimedia Systems, Standards, and Networks*, Marcel Dekker Inc, 2000.

[6] Y. Sun, W. Trappe, and K.J.R. Liu, "An efficient key management scheme for secure wireless multicast," *Proc. of IEEE International Conference on Communication*, vol. 2, pp. 1236–1240, 2002.

[7] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. on Networking*, vol. 8, pp. 16–30, Feb. 2000.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introducation to Algorithms*, MIT Press, 2nd edition, 2001.