Scalable Hierarchical Access Control in Secure Group Communications

Yan Sun and K. J. Ray Liu
Department of Electrical and Computer Engineering
and the Institute of Systems Research
University of Maryland, College Park, MD 20742
Email: ysun, kjrliu@glue.umd.edu

Abstract— **Many group communications require a security infrastructure that ensures multiple levels of access privilege for group members. Access control in hierarchy is prevalent in multimedia applications, which consist of users that subscribe to different quality levels or different sets of data streams. In this paper, we present a multi-group key management scheme that achieves such a hierarchical access control by employing an integrated key graph and by managing group keys for all users with various access privileges. Compared with applying existing tree-based group key management schemes directly to the hierarchical access control problem, the proposed scheme significantly reduces the communication, computation and storage overhead associated with key management and achieves better scalability when the number of access levels increases. In addition, the proposed key graph is suitable for both centralized and contributory environments.**

I. Introduction

The rapid progress in the technologies underlying multicast networking has led to the development of many group-oriented applications, such as video conferencing, pay-per-view broadcast of sport events, and communal gaming [1] [2]. For the purpose of security or billing, many group-oriented communications require the *access control* mechanism such that only authorized members can access group communications [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 group, the encryption keys shall be updated to prevent the leaving/joining user from accessing the future/prior communications [3]. The issues of establishing and updating the group keys are addressed by group *Key Management* schemes [3]–[5].

There already exist many key management schemes, such as the centralized schemes presented in [2], [4]–[12] and the contributory schemes presented in [13]–[21]. These schemes address the access control issues in a single multicast session. They focus on establishing and updating keys with dynamic membership and provide all group members the same level of access privilege. That is, the users who possess the decryption keys have the full access to the content, and the users who do not have the decryption keys cannot interpret the data.

In practice, many group applications contain multiple related data streams and have the members with various access privileges. These applications prevail in various scenarios.

- Multimedia applications distributing data in multi-layer coding format [22]. For example, in a video broadcast, users with a normal TV receiver can receive the normal format, while others with HDTV receivers can receive both the normal format and the extra information needed to achieve HDTV resolution.
- Multicast programs containing several related services. For example, the cellular phone service provider offers a set of extra broadcast services, such as weather, news, traffic and stock quote.
- Communications in hierarchically managed organizations, such as military group communications where participants have different access authorizations.

In those applications, group members subscribe to different data streams, or possibly multiple of them. Thus, it is necessary to develop group access control mechanism that supports the multi-level access privilege, which is referred to as the *hierarchical access control*. Previously, hierarchical access control has been studied in static scenarios where each type of access privilege is represented by one class [23]–[26]. Those studies focus on the relationship between the keys that are assigned to different classes [23]–[27], but do not address the dynamic membership problem that is essential in multicast communications. Recently, a tree-based hierarchical access control scheme that supports dynamic multicast groups was presented in [28]. This scheme, however, is restricted to the Bell and La Padula access model [29] and is not suitable for many commercial group-oriented applications with diverse access policies.

Hierarchical access control problem in multicast communications can be converted into a set of single-session access control problems, which have already been solved by many existing group key management schemes [2]–[21]. By doing so, the key management for each data stream is performed separately. This method leads to inefficient use of keys and does not scale well when the number of data streams increases, as we will demonstrate in the later sections.

In this paper, we develop a *multi-group key management* scheme that manages keys for all members with different access privileges. Particularly, we design an *integrated key graph* that maintains the keying material for all members, and incorporate new functionalities that are not present in conventional multicast key management, such as user re-

0-7803-8355-9/04/$20.00 ©2004 IEEE.

location on the key graph. The proposed multi-group key management scheme achieves forward and backward security [19] when users (1) join the group communication with certain access privilege; (2) leave the group; and (3) change access privileges by adding or dropping the subscription of one or several data streams. In addition, the idea of the integrated key graph can be used in both centralized and contributory environments. This paper will first present the centralized multi-group key management scheme and then discuss its extension in the distributed scenarios. Compared with using single-session access control solutions, such as a variety of tree-based key management schemes [6] [19], the proposed scheme reduces the usage of the communication, computation and storage resources, and is scalable when the number of data streams increases.

The rest of the paper is organized as follows. The hierarchical access control problem is formulated in Section II. The centralized multi-group key management is presented in Section III, IV, and V. Particularly, Section III describes the construction of the integrated key graph and the rekey algorithm. Section IV analyzes the performance of the proposed scheme and the asymptotical behavior. Section V provides the simulation results and compares the proposed scheme with the tree-based solutions in various application scenarios. The contributory key management scheme that uses the integrated key graph is presented in Section VI, followed by the conclusion in Section VII.

II. System Description

In this section, we first introduce the basic concepts that describe the group communication systems containing multiple data streams and users with different access privileges. Then, for such systems, we formulate the hierarchical access control problem with dynamic group membership.

A. System Description

Let $\{r_1, r_2, \cdots\}$ denote the set of *resources* in the system. In the group communication scenario, each resource corresponds to a data stream that is transmitted in one multicast session. Each multicast session is associated with a multicast address and a multicast routing tree [1]. The routing trees for different multicast sessions can be jointly constructed [30].

From the data transmission points of views, the users belonging to the same multicast session form a *Data Group* (DG). That is, one DG contains the users that can access to a particular resource. It is clear that the DGs can have overlapped membership because users may subscribe multiple resources. The users are also divided into non-overlapping *Service Groups* (SG) according to access privilege. One SG contains the users that are authorized to access the exactly same set of resources. In this paper, the DGs are denoted by $\{D_1, D_2, \cdots, D_M\}$, where M is the total number resources. Users in the DG D_m are authorized to obtain the resource r_m. The SGs are denoted by $\{S_1, S_2, \cdots, S_I\}$, where I is the total number SGs. It is easy to prove that $I \leq 2^M - 1$. The typical access relationships in group communications are illustrated through the following examples.

Example 1. Multimedia applications that distribute data in multi-layer format [22].

- *Resources*: {base layer (r_1), enhancement layer 1 (r_2), enhancement layer 2 (r_3)}.
- *Service Groups*: {users subscribing basic quality (S_1), users subscribing moderate quality (S_2), users subscribing high quality (S_3)}.
- *Capability lists*: S_1 access $\{r_1\}$; S_2 access $\{r_1, r_2\}$; S_3 access $\{r_1, r_2, r_3\}$.
- *Data Groups*: D_1 access $\{r_1\}$; D_2 access $\{r_2\}$; D_3 access $\{r_3\}$.

Example 2. Multicast programs containing several related services.

- *Resources*: {News (r_1), Stock quote (r_2), Traffic/Weather (r_3)}.
- *Service Groups*: Users can subscribe any combination of the resources. Thus, there are total 7 SGs, denoted by $S_1, S_2, \cdots, S_7$.
- *Capability lists*: S_1 access $\{r_1\}$; S_2 access $\{r_2\}$; S_3 access $\{r_3\}$; S_4 access $\{r_1, r_2\}$; S_5 access $\{r_1, r_3\}$; S_6 access $\{r_2, r_3\}$; S_7 access $\{r_1, r_2, r_3\}$.
- *Data Groups*: D_1 access $\{r_1\}$; D_2 access $\{r_2\}$; D_3 access $\{r_3\}$.

To make clear mathematical representations, t_m^i is defined as:

$$t_m^i = \begin{cases} 1, & \text{the SG } S_i \text{ can access the resource } r_m \\ 0, & \text{otherwise} \end{cases},$$

for $i = 1, \cdots, I$ and $m = 1, \cdots, M$. In addition, we define a virtual service group, S_0, which represents users who do not participate in any group communications. Clearly, $t_m^0 = 0$ for $m = 1, \cdots, M$.

Based on these definitions, the group size of SGs and DGs must satisfy:

$$n(D_m) = \sum_{i=1}^{I} t_m^i \cdot n(S_i), \tag{1}$$

where $n(S_i)$ denotes the number of users in the SG S_i and $n(D_m)$ denotes the number of users in the DG D_m.

B. Hierarchical Access Control in Multicast Communications

In order to achieve hierarchical access control while not transmitting multiple copies of data, it is necessary to encrypt different resources using separate keys [31]. Thus, the users in each DG share a key, referred to as the *data group key*. The data group key of D_m, denoted by K_m^D, is used to encrypt the resource r_m. Obviously, the users in SG S_i must possess $\{K_m^D, \forall\ m : t_m^i = 1\}$.

In the applications containing multiple SGs, users not only join/leave the service, but also switch between SGs by subscribing or dropping data streams. We introduce the notation, $S_i \rightarrow S_j$, which represents a user switching from SG S_i to SG S_j. Since S_0 represents the users who do not participate any

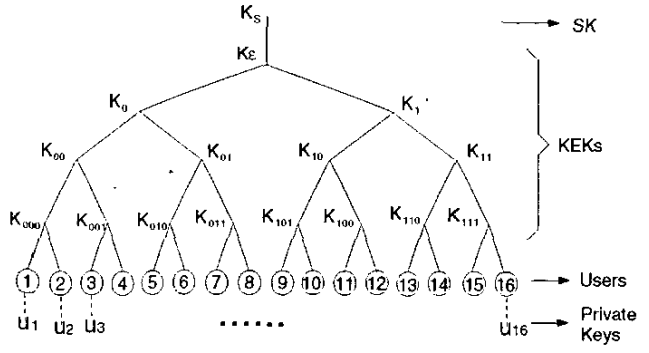


Fig. 1. A typical key management tree

group communications, this notation also describes the cases of user join ($S_0 \rightarrow S_i$) and departure ($S_i \rightarrow S_0$).

Similar to the single-session access control problem addressed by traditional key management schemes [3], hierarchical access control in this work should guarantee the forward and backward security [19]. When a user switches from SG S_i to S_j, it is necessary to

- update the data group keys of $\{D_m, \forall\ m : t_m^i = 0 \text{ and } t_m^j = 1\}$, such that the switching user cannot access the previous communications in those DGs;
- and update the data group keys of $\{D_m, \forall\ m : t_m^i = 1 \text{ and } t_m^j = 0\}$, such that the switching user cannot access the future communications in those DGs.

III. Centralized Multi-group Key Management Scheme

Popular key management schemes are classified as centralized schemes and contributory schemes [8]. Centralized key management, such as [2], [4]–[12], depends on a centralized server, referred to as the key distribution center (KDC), which generates and distributes encryption keys. The contributory key management schemes do not rely on centralized servers. Instead, every group member makes independent contribution and participates the process of group key establishment, as in [13]–[21].

Hierarchical access control can be achieved in either centralized or contributory manner. While the contributory solution will be discussed in Section VI, this section and the following two sections will be dedicated to the centralized schemes.

A. Employing independent key trees to achieve Hierarchical access control

To reduce the communication, computation and storage overhead, tree structure is widely used in centralized key management schemes to maintain the keying material and coordinate the key update [2], [4]–[9].

A typical key tree used in centralized key management schemes [2]–[6], [8] is illustrated in Figure 1. Each node of the key tree is associated with a key. The root of the key tree is associated with the session key. Each leaf node is associated with a user's private key. The intermediate nodes are associated with key-encrypted-keys (KEK), which are auxiliary keys and

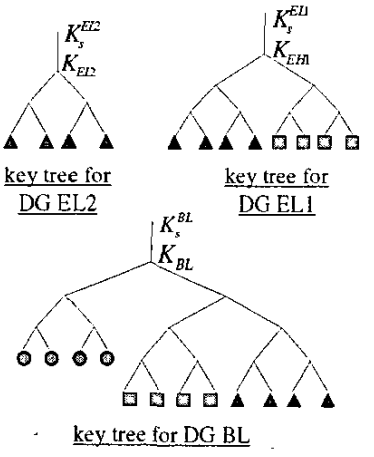


Fig. 2. Independent-tree key management scheme for layered coded multimedia services

only for the purpose of protecting the session key and other KEKs. To make the presentation concise, we do not distinguish the node and the key associated with this node in the remainder of the paper.

Each user stores his private key, the session key, and a set of KEKs on the path from himself to the root of the key tree. When a user leaves the service, the KDC generates new versions of his keys and distributes new keys to the remaining users by sending a set of rekey messages in the multicast channel [4], [5]. The communication overhead associated with key updating can be described by *rekey message size*, defined as the amounts of rekey messages measured in the unit as the same size as the SK or KEKs. It has been shown that the rekey message size increases linearly with the logarithm of the group size [3]–[5]. When a user joins the service, the KDC chooses a leaf position on the key tree to put the joining user. In [6], the KDC updates the keys along the path from the new leaf to the root by generating the new keys from the old keys using a one-way function [6]. This rekeying procedure for user join does not need to transmit additional rekey messages.

When using tree-based schemes to achieve hierarchical access control, a separate key tree must be constructed for each DG, with the root being the data group key and the leaves being the users in this DG. This approach is referred to as the *Independent-tree* key management scheme, and is illustrated in Figure 2.

The main advantage of employing separate key trees is the simplicity in implementation and group management. This scheme, however, does not exploit the relationship among the subscribers and makes inefficient use of keys due to the overlap in DG membership. As an extreme example, if a user that subscribes all DGs leaves, key updating has to take place on all key trees.

B. Multi-group Key Management Scheme

To achieve efficient hierarchical access control, we propose a *multi-group* key management scheme that employs one integrated key graph accommodating key materials for all

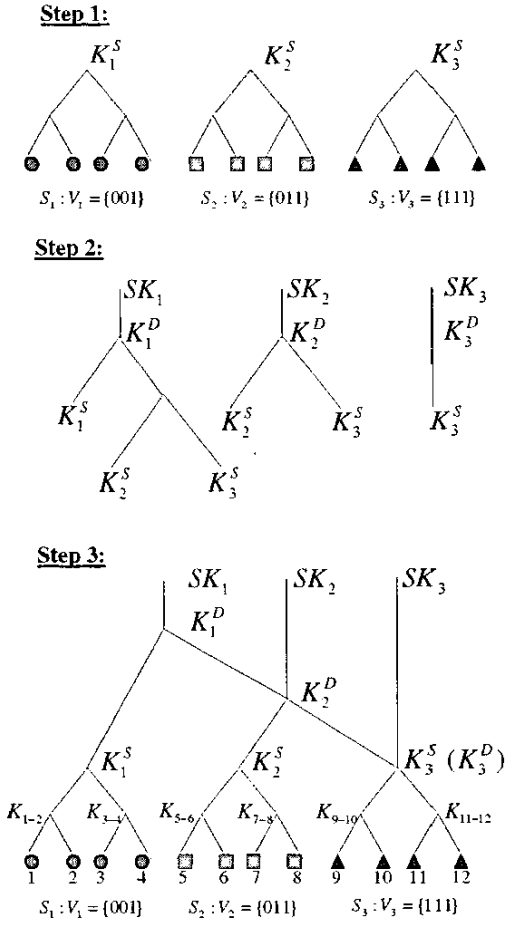


Fig. 3. Multi-group key management graph construction

users. This key graph comprises several key trees, and is constructed in three steps.

Step1: For each SG S_i, construct a subtree, whose leaf nodes are the users in S_i and whose root is associated with a key, denoted by K_i^S. These subtrees are called the *SG-subtrees*.

Step2: For each DG D_m, construct a subtree whose root is the DG key K_m^D and whose leaves are $\{K_i^S, \forall i : t_m^i = 1\}$. These subtrees are referred to as the *DG-subtrees*.

Step3: Generate the key graph by connecting the leaves of the DG-subtrees and the roots of the SG-subtrees.

This 3-step procedure is illustrated in Figure 3 for the Example 1 in Section II-A. Here, each SG has 4 users. It is noted that duplicated structures may appear on DG-subtrees and can be merged to reduce the number of keys on the key graph. In the example shown in Figure 3, K_3^S and K_3^D are merged because they are on the same line. The DG-subtrees of D_2 and D_1 have the same structures that connect K_2^S and K_3^S. Thus, the parent node of K_2^S and K_3^S on DG-subtree of D_2 is merged with K_2^D.

This multi-group key graph can also be interpreted as M overlapped key trees, each of which has K_m^D as the root and the users in DG D_m 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 the independent-tree scheme. Therefore, it has the potential to reduce the overhead associated with key updating.

As defined in [4], *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 our key graph, the keyset of a user in SG S_i is the keys on the pathes from himself to the roots of the DG-subtrees of $\{D_m, \forall m : t_m^i = 1\}$. It is noted that the keyset of the users in S_0 is simply an empty set.

Besides user join and departure, the rekey algorithm in the multi-group key management scheme must address users' relocation on the key graph. Next, we describe the rekey algorithm for $S_i \rightarrow S_j$, which already includes the cases for user join and departure. Here, the switching user is moved from the SG-subtree of S_i to a new location on the SG-subtree of S_j. Let ϕ_i denote the keyset associated with the user's previous position, and ϕ_j denote the keyset associated with the user's new position. Then,

- the KDC updates the keys in $\overline{\phi_i} \cap \phi_j$ using one-way functions, similar as that in [6] for user join,
- and the KDC generates new versions of the keys in $\phi_i \cap \overline{\phi_j}$ and distributes these new keys from bottom to up by using their children node keys, similar as that in [6] for user departure.

We illustrate this rekey algorithm through an example, where user 8 switches from SG S_2 to S_1. On the SG-subtree of S_1, we assume that the leaf node associated with user 4 is split to accommodate user 8. Then, user 4 and 8 will share a new KEK, denoted by K_{4-8}. On the SG-subtree of S_2, user 7 will be moved up and occupy the node that is previously associated with K_{7-8}. In this case, ϕ_2 is $\{K_{7-8}, K_2^S, K_2^D, SK_2, K_1^D, SK_1\}$ and ϕ_1 is $\{K_{4-8}, K_{3-4}, K_1^S, K_1^D, SK_1\}$.

Let the notation x^{new} represent the new version of key x, $\{y\}_x$ represent the key y encrypted by key x, and u_k represent the private key of user k. As in [6], each key is associated with a revision number.

In this example, the KDC generates the new keys, K_{3-4}^{new} and $K_1^{S,new}$, from the old keys using a one-way function, and increases the revision numbers of those new keys. Thus, the user 1,2,3,4 will know about the key change when the data packet indicating the increase of the revision numbers first arrives, and compute the new keys using the one-way function. No rekey messages are necessary for delivering K_{3-4}^{new} and $K_1^{S,new}$.

In addition, the KDC generates new keys, $\{K_{4-8}^{new}, K_2^{S,new}, K_2^{D,new}$, and SK_2^{new}, and distributes them through a set of rekey messages as:

$$\{K_{4-8}^{new}\}_{u_8},\ \{K_{4-8}^{new}\}_{u_4},\ \{K_2^{S,new}\}_{K_{5-6}},\ \{K_2^{S,new}\}_{u_7}$$

$$\{K_2^{D,new}\}_{K_2^{S,new}},\ \{K_2^{D,new}\}_{K_3^S},\ \{SK_2^{new}\}_{K_2^{D,new}}$$

In the example, the rekey message size is 7.

It is noted that $\overline{\phi_i} \cap \phi_j$ may contain new KEKs that are created for accommodating the switching user. These new KEKs should be encrypted using users' private keys and distributed through sending rekey messages. In addition, $\phi_i \cap \overline{\phi_j}$ may contain the KEKs that are eliminated after the relocation of the switching user. Obviously, these keys should not be updated.

IV. Performance Measures and Analysis

Communication, computation and storage overhead associated with key updating are major performance measures for key management schemes [3]–[5]. In the hierarchical access control scenarios, we define the performance measures as:

- Storage overhead at the KDC, denoted by R_{KDC} and defined as the expected number of keys stored at the KDC.
- Rekey overhead at the KDC, denoted by M_{KDC} and defined as the expected number of rekey messages transmitted by the KDC per key updating.
- Storage overhead of users, denoted by $R_{u\in S_i}$ and defined as the expected number of keys stored by the users in the SG S_i.
- Rekey overhead of users, denoted by $M_{u\in S_i}$ and defined as the expected number of rekey messages received by the users in the SG S_i per key updating.

Here, R_{KDC} and $R_{u\in S_i}$ describe the storage overhead, while M_{KDC} and $M_{u\in S_i}$ reflect the usage of communication and computation resources.

A. Storage Overhead

Similar to most key management schemes [3]–[6], [8], the key tree investigated in this work is fully loaded and maintained as balanced as possible by putting the joining users on the shortest branches.

Let $f_d(n)$ denote the length of the branches and $r_d(n)$ denote the total number of keys on the key tree when the key tree has degree d and accommodates n users. Since the key tree is balanced, $f_d(n)$ is either L_0 or L_0+1, where $L_0 = \lfloor \log_d n \rfloor$. Particularly,

- the number of users who are on the branches with length L_0 is $d^{L_0} - \lceil \frac{n-d^{L_0}}{d-1} \rceil$;
- and, the number of users who are on the branches with length L_0+1 is $n - d^{L_0} + \lceil \frac{n-d^{L_0}}{d-1} \rceil$.

Thus, the total number of keys on this key tree is calculated as:

$$r_d(n) = n + 1 + \frac{d^{L_0}-1}{d-1} + \lceil \frac{n-d^{L_0}}{d-1} \rceil. \quad (2)$$

Using the fact that $\frac{n-d^{L_0}}{d-1} \le \lceil \frac{n-d^{L_0}}{d-1} \rceil < \frac{n-d^{L_0}}{d-1} + 1$, we have

$$\frac{dE[n]-1}{d-1} + 1 \le E[r_d(n)] < \frac{dE[n]-1}{d-1} + 2, \quad (3)$$

where the expectation, $E[.]$, is taken over the distribution of n and the length of the branches on the key trees. The left-hand-side equality achieves when $\log_d(n)$ is an integer. In addition, since $\log_d(n)$ is a concave function and $\lfloor \log_d n \rfloor \le \log_d n$, it is clear that

$$E[f_d(n)] \le E[\log_d n] + 1 \le \log_d E[n] + 1. \quad (4)$$

With equation (3) and (4), we are ready to analyze the storage overhead. When using the separate key trees (i.e. independent-tree scheme), the KDC stores all keys on total M key trees, and users in S_i store subsets of keys on the key trees that are associated with $\{D_m, \forall m : t_m^i = 1\}$. Thus,

$$R_{KDC}^{ind} = \sum_{m=1}^{M} E\left[r_d(n(D_m))\right], \quad (5)$$

$$R_{u\in S_i}^{ind} = \sum_{m=1}^{M} t_m^i \left(E[f_d(n(D_m))] + 1\right). \quad (6)$$

In the Multi-group key management scheme, the DG-subtree of D_m has $c_m = \sum_i t_m^i$ leaf nodes. Before removing the redundancy on the DG-subtrees, there are in total $\sum_{m=1}^{M} r_d(c_m)$ keys on the DG-subtrees. Also, the total number of keys on the SG-subtrees is $\sum_{i=1}^{I} r_d(n(S_i))$. Therefore, after merging duplicated structures on the DG-subtrees, the storage overhead at the KDC is

$$R_{KDC}^{mg} \le \sum_{i=1}^{I} E[r_d(n(S_i))] + \sum_{m=1}^{M} E\left[r_d(c_m)\right]. \quad (7)$$

A user in the SG S_i stores $f_d(n(S_i))$ keys on the SG-subtree and up to $\sum_{m=1}^{M} t_m^i (f_d(c_m)+1)$ keys on the DG-subtrees. Therefore, the users' storage overhead of the multi-group scheme is:

$$R_{u\in S_i}^{mg} \le E[f_d(n(S_i))] + \sum_{m=1}^{M} t_m^i (E[f_d(c_m)]+1). \quad (8)$$

Without loss generality, we demonstrate the storage overhead of the independent-tree and the multi-group key management in the applications containing multiple layers, as described in Example 1 in Section II-A. In this case, $t_m^i = 1$ for $m \le i$ and $t_m^i = 0$ for $m > i$. We also assume that each layer contains the same amount of users, denoted by $n(S_i) = n_0$. Thus, $n(D_m) = (M-m+1)n_0$. Using (6) and (8), the users' storage overhead is calculated as:

$$R_{u\in S_i}^{ind} = \sum_{m=1}^{i} \left(E[f_d((M-m+1)\cdot n_0)] + 1\right), \quad (9)$$

$$R_{u\in S_i}^{mg} = E[f_d(n_0)] + \sum_{m=1}^{i} \left(E[f_d(M-m+1)] + 1\right). \quad (10)$$

When the group size is large, i.e. $n_0 \to \infty$, equation (4) (9) and (10) tell that

$$R_{u\in S_i}^{ind} \sim O(i \cdot \log(n_0)),\ R_{u\in S_i}^{mg} \sim O(\log(n_0)). \quad (11)$$

Using (5) and (7), the storage overhead at the KDC is calculated as:

$$R_{KDC}^{ind} = \sum_{m=1}^{M} E[r_d(m \cdot n_0)], \quad (12)$$

$$R_{KDC}^{mg} \leq M \cdot E[r_d(n_0)] + \sum_{m=1}^{M} E[r_d(m)]. \quad (13)$$

From (3), it is seen that $\lim_{n\to\infty} r_d(n) = \frac{d}{d-1}n$. Therefore,

$$R_{KDC}^{ind} \sim O(\frac{d}{d-1}\frac{M(M+1)}{2}n_0), \quad (14)$$

$$R_{KDC}^{mg} \sim O(\frac{d}{d-1}M \cdot n_0). \quad (15)$$

By using the integrated key graph instead of the separate key trees, the multi-group key management scheme reduces the storage overhead at both the KDC and the users' side. As indicated in (14) and (15), storage advantage of the proposed scheme becomes larger when the applications contain more SGs, i.e. requiring more levels of access control. The proposed scheme in fact scales better when the number of layers (M) increases. As we will show later in Section V, this property is also valid for the rekey overhead.

B. Rekey Overhead

The rekey overhead defined earlier in this Section is closely related with the users' statistical behaviors, such as the probability for user joining/leaving/switching. Without specifying the model for users' dynamic membership in this section, we calculate the amount of rekey messages transmitted by the KDC when one user switches from S_i to S_j, denoted by $C_{i,j}$.

Switching from SG S_i to SG S_j is equivalent to adding the subscription to the DG $\{D_m, \forall m : t_m^i = 0 \text{ and } t_m^j = 1\}$ and dropping the subscription to the DG $\{D_m, \forall m : t_m^i = 1 \text{ and } t_m^j = 0\}$. When using the tree-based key management schemes, the rekey message size is calculated as:

$$C_{ij}^{ind} = \sum_{m=1}^{M} \max(t_m^i - t_m^j, 0) \cdot (d \cdot f_d(n(D_m))). \quad (16)$$

It is noted that when $t_m^i = 1$ and $t_m^j = 0$, the term $(\max(t_m^i - t_m^j, 0))$ equals to 1 and $d \cdot f_d(n(D_m)$ rekey messages are necessary to update keys on the key tree associated with the DG D_m.

In the multi-group key management scheme, when a user switches from S_i to S_j and $i \neq j$,

- The amount of messages that update the keys on the SG-subtree of S_i is up to $(d \cdot f_d(n(S_i)) - 1)$.
- The amount of messages that distribute the new KEKs on the SG-subtree of S_j is up to 2.
- If this user drops the subscription of the DG D_m, i.e. $(\max(t_m^i - t_{in}^j, 0)) = 1$, the amount of rekey messages that update keys on the DG-subtree of D_m is up to $(d \cdot f_d(c_m) + 1)$.
- If this user remains the subscription of the DG D_m, i.e. $t_m^i = t_m^j = 1$, the amount of rekey messages that update keys on the DG-subtree of D_m is up to $(d \cdot f_d(c_m))$.

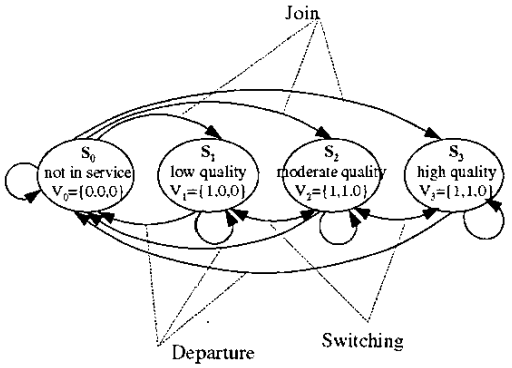


Fig. 4. Discrete Markov chain model for Multi-layer applications.

Therefore, when using the multi-group scheme and $i \neq j$, we have

$$C_{ij}^{mg} \leq \sum_{m=1}^{M} (\max(t_m^i - t_m^j, 0) \cdot (d \cdot f_d(c_m) + 1) + t_m^i t_m^j d \cdot f_d(c_m)) + d \cdot f_d(n(S_i)) + 1. \quad (17)$$

Similar as in Section IV-A, we analyze the rekey overhead in a multi-layer scenario with $n(S_i) = n_0$. In this case, the rekey message size for one user leaving, i.e. $S_j \to S_0$, is computed from (16) and (17) as:

$$C_{0j}^{ind} = \sum_{m=1}^{j} d \cdot E[f_d((M - m + 1)n_0)], \quad (18)$$

$$C_{0j}^{mg} \leq d \cdot E[f_d(n_0)] + 1 + \sum_{m=1}^{j} (d \cdot E[f_d(M - m + 1)] + 1). \quad (19)$$

When $n_0 \to \infty$, we can see that

$$C_{0j}^{ind} \sim O(i \cdot d \cdot \log(n_0)), \; C_{0j}^{mg} \sim O(d \cdot \log(n_0)). \quad (20)$$

The comprehensive comparison between the proposed scheme and the independent-tree scheme will be provided in Section V through simulations.

V. Simulations and Performance Comparison

In this section, the performance of the proposed multi-group key management scheme are compared with the existing tree-based key management schemes in various application scenarios.

A. Statistical dynamic membership model

In this work, we assume that when a user switches between SGs, the SG that he switches to depends only on his current SG. In addition, [32] [33] studied the characteristics of the membership dynamics of MBone multicast sessions and suggested that the users arrival process and membership duration can be modelled by Poisson and exponential distribution respectively, in a short period of time.

Therefore, the users' statistical behavior can be described by an embedded Markov chain [34]. Particularly, there are a total of $I + 1$ states, denoted by $\tilde{S}_i$, $i = 0, \cdots, I$. When a user is in the SG S_i, he is in the state $\tilde{S}_i$. After a user enters state $\tilde{S}_i$, i.e. subscribes or switches to SG S_i, this user stays

at state $\tilde{S}_i$ for time T_i, which is governed by an exponential random variable. When time is up, the user moves to state $\tilde{S}_j$. The selection of $\tilde{S}_j$ only depends on the current state $\tilde{S}_i$ and is not related with previous states.

In practice, it is usually not necessary to update keys immediately after membership changes. Many applications allow the join/departure users to receive limited previous/future communications [35]. For example, a joining user may receive a complete group-of-picture (GOP) [22] although partial of this GOP has been transmitted before his subscription. Those situations prefer *batch rekeying* [35], which is to postpone the update of keys such that the rekeying overhead is reduced by adding or removing several users altogether.

In this work, batch rekeying is implemented as periodic updating of keys. The time between key updates are fixed and denoted by B_t. For the users who join/leave/switch SGs in the time interval $((k-1)B_t, kB_t]$, the key updating will take place at time kB_t, and k are positive integers. When using batch rekeying, from the key updating points of view, we can prove that the previous continuous Markov model can be simplified as a discrete Markov chain model [34], as illustrated in Figure 4. In this model,

- The transition matrix is denoted by $P = [p_{ij}]_{(I+1)\times(I+1)}$, where p_{ij} is the probability that one user moves from SG S_i to S_j in the time interval $(kB_t, (k+1)B_t]$ given that this user is in S_i at time kB_t.
- The n-step transition probability matrix is denoted by $P(n)$, and obviously, $P(n) = P^N$. The element at the i^{th} row and j^{th} column of $P(n)$ is denoted by $p_{ij}(n)$.
- The stationary state probability is a 1-by-$(I+1)$ vector, denoted by $\pi = [\pi_0, \pi_1, \cdots, \pi_I]$.

We notice that most practical applications have the following properties.

- Since users should be able to subscribe every SG, $p(n)_{0j} \neq 0$ for some positive finite n, and for any j.
- Since users should be able to leave from every SG, $p(n)_{i0} \neq 0$ for some positive finite n, and for any i.
- Since a user can always stay in his current SG, $p_{ii} > 0$.
- The expected time that a user stays in the group communication, i.e. the mean recurrence time [34] of the state S_0, is finite.

Because of these properties, we can show that this Markov chain is irreducible, aperiodic and positive recurrent. As a result, the stationary state probability mass function (pmf) exists [34] and is the unique solution of

$$\pi P = \pi, \text{ and } \sum_i \pi_i = 1 \qquad (21)$$

B. Performance with different group size

We first study the applications containing multiple layers, as described in Example 1 in Section II-A. The users in SG S_i have access to the DG $D_1, D_2, \cdots, D_i$. In addition, we add the following constraints on the transition matrix.

- Users join the service to different SGs with the same probability, i.e. $P_{0j} = \alpha, \forall j > 0$.

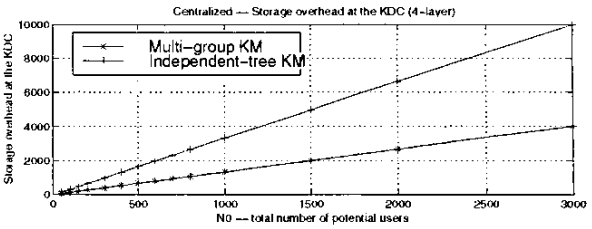


Fig. 5. Storage overhead at the KDC

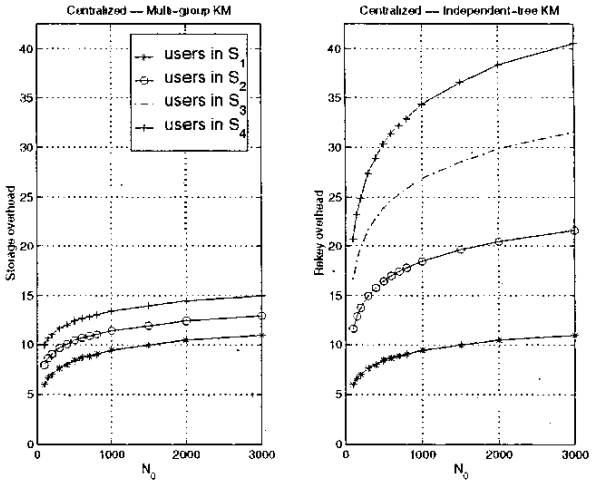


Fig. 6. Storage overhead at the users in each SG

- Users leave the service from different SGs with the same probability, i.e. $P_{i0} = \beta, \forall i > 0$.
- While a user remains in the service, he adds/drops only one data stream a time, i.e. $P_{i,j} = 0, \forall i, j > 0$ and $|i - j| > 1$. Also, he switches between SGs with the same probability, i.e. $P_{i,j} = \gamma, \forall i, j > 0$ and $|i - j| = 1$.

Thus, the transition matrix is described by only three variables. For example, the multi-layer service with $M = 3$ has the transition matrix as:

$$P = \begin{bmatrix} 1-3\alpha & \alpha & \alpha & \alpha \\ \beta & 1-\beta-\gamma & \gamma & 0 \\ \beta & \gamma & 1-\beta-2\gamma & \gamma \\ \beta & 0 & \gamma & 1-\beta-\gamma \end{bmatrix}$$

In all simulations, batch rekeying is applied and the key trees are binary. The stationary state is chosen as the initial state, i.e. S_i contains $N_0\pi_i$ users at the beginning of the service. N_0 is the total group size.

In Figure 5, 6, 7 and 8, the multi-group scheme and the independent-tree scheme are compared for different group size. The results are averaged over 300 realizations, and the number of layers is 4. In these simulations, we choose $\alpha = 0.005$, $\beta = 0.01$, and $\gamma = 0.001$.

Figure 5 shows that the storage overhead at the KDC, R_{KDC}, increases linearly with the group size, which can be verified by (5) (7) and (3). The multi-group scheme reduces R_{KDC} by more than 50%.

Figure 6 shows that the users' storage overhead, $R_{u\in S_i}$, increases linearly with the logarithm of the group size, which

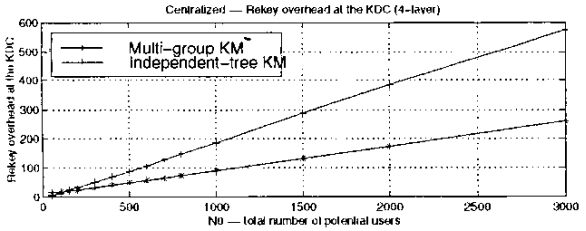


Fig. 7. Rekey overhead at the KDC

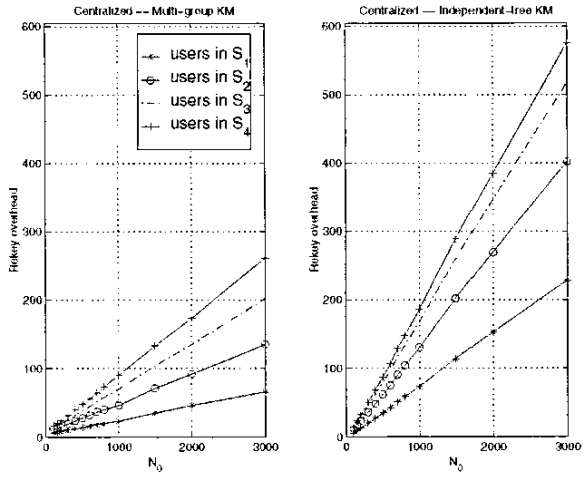


Fig. 8. Rekey overhead at the users in each SG

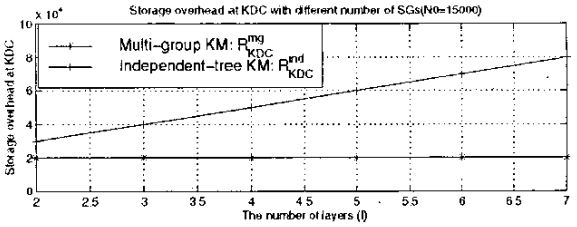
can be verified by (9) and (10). The users who subscribe only one layer have the similar storage overhead in both schemes. The multi-group scheme reduces storage overhead for the users subscribing multiple layers. The storage advantage of the proposed scheme is larger for the users subscribing more layers.

The KDC's rekeying overhead (R_{KDC}) and the users' rekey overhead ($R_{u \in S_i}$) are shown in Figure 7 and 8, respectively. In both cases, the multi-group scheme reduces the rekey overhead by more than 50%.

C. Scalability with increase in the number of layers

Next, we change the number of layers (M) while maintaining roughly the same number of users in the service by choosing the join probability α as $0.02/M$. The values of β and γ are the same as those in Section V-B.

Figure 9(a) and Figure 10(a) show the storage and rekey overhead at the KDC, respectively. When M increases, the storage and rekey overhead of the multi-group scheme do not change much, while the overhead of the independent-tree scheme increases linearly with M. It is not surprising that the multi-group scheme scales better when M increase. By removing the redundancy in DG membership, the scale of the key graph mainly depends on the total group size, not the number of layers or data streams. On the other hand, by constructing M separate key trees, the independent-tree scheme requires larger storage and rekey overhead when M increases.



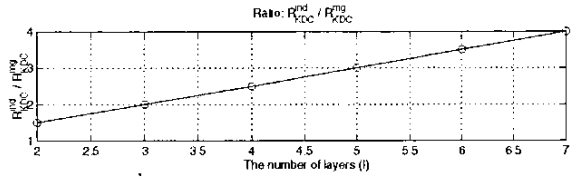
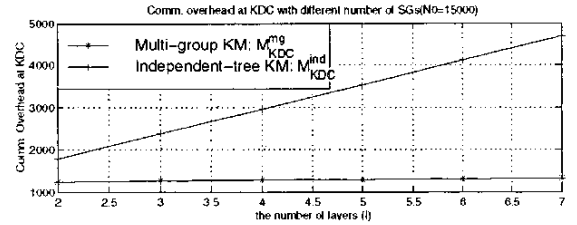


Fig. 9. Storage overhead at the KDC with different number of SGs



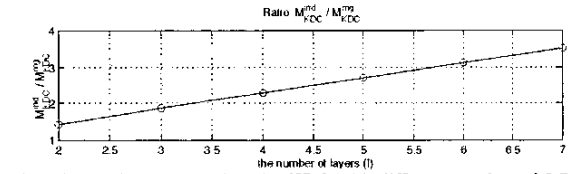


Fig. 10. Rekey overhead at the KDC with different number of SGs

Figure 9(b) shows that the ratio between R^{ind}_{KDC} and R^{mg}_{KDC} increases linearly with M, which agrees with equation (14) and (15). This is also true for the rekey overhead, as shown in Figure 10(b).

D. Performance with different transition probability

In the previous simulations, we set $\gamma = 0.1\beta$, which means that the users are more likely to leave the service than to switch SGs. Figure 11 shows the rekey overhead for different values of γ. Note that γ describes the probability of users' switching between SGs. In this simulation, $M = 4$, $N_0 = 1000$, and the values of α and β are the same as those in the previous experiments.

When γ is very small, the multi-group scheme reduces the rekey overhead by about 50%, as we have shown in the previous simulations. When γ is less than 2β, the rekey advantage of the multi-group scheme decreases with the increase of γ. This is because the multi-group scheme introduces larger rekey overhead when users switch SGs by simply adding the subscription to more data streams. To see this, let a user move from S_1 to S_2. When using the independent-tree scheme, this

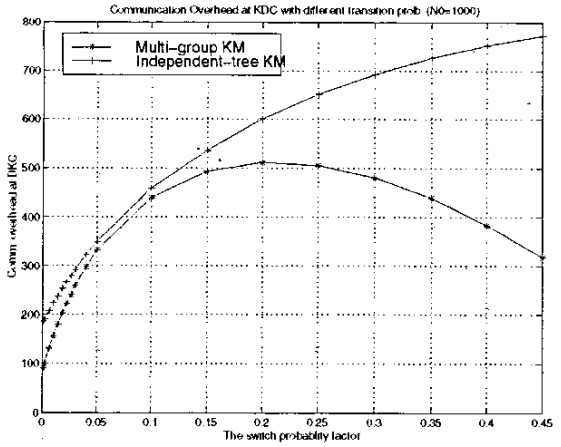


Fig. 11. Rekey overhead at the KDC with different transition prob.

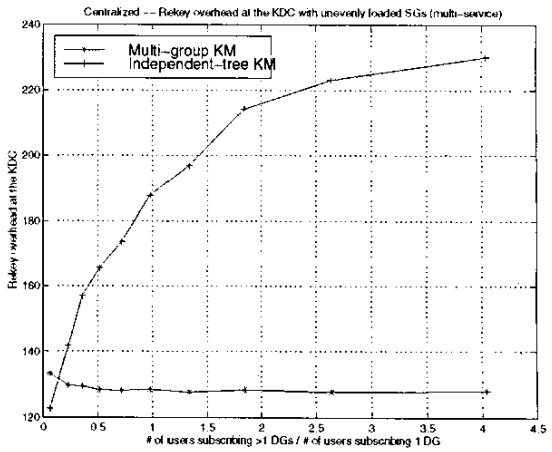


Fig. 12. Rekey overhead at the KDC with unevenly loaded SGs in multi-service applications

user only need to be added to the key tree associated with the DG D_2 and almost no rekeying messages are necessary. When using the multi-group scheme, we need to update keys on the SG-subtree of S_1 and the DG-subtree of D_1. Therefore, the performance gain decreases when more users tend to switch SGs.

When γ continues to increase, however, the rekey overhead of the multi-group scheme decreases. Particularly, when $\gamma = 0.45$, the performance gain of the multi-group scheme is about 50% again. Large values of γ describe the scenarios where users are much more likely to switch SGs than to stay in the current SG. When a significant potion of users switches away from a SG, the size of the SG-subtree is greatly reduced. In this case, removing a large potion of users from the key tree with batch rekeying requires less rekey messages than just removing several users.

E. Simulation of Multi-service applications

We also simulated the multi-service scenarios as described in Example 2 in Section II-A. The users can subscribe any one or multiple DGs and switch between any SGs. The transition matrix is 8 by 8, with $P_{j0} = 0.01, \forall j > 0$ and $P_{i,j} = 0.00017, \forall i, j > 0$ and $i \neq j$. N_0 is fixed to be 1500. The values of $P_{0i}, \forall i > 0$, are adjusted such that the SGs contain varying number of users while $\sum_{i=1}^{I} P_{0i}$ is maintained to be the same.

The horizontal axis in Figure 12 is the ratio between the number of users subscribing multiple DGs $(n(S_4) + n(S_5) + n(S_6) + n(S_7))$ and the number of users subscribing only one DG $(n(S_1) + n(S_2) + n(S_3))$. Larger is the ratio, more overlapping is in the DG membership. Figure 12 shows that the advantage of the multi-group scheme is larger when more users subscribe multiple DGs.

VI. Contributory Solutions for Hierarchical Access Control

In many scenarios, it is not preferred to rely on a centralized server that arbitrates the establishment of the group key. This might occur in applications where group members do not explicitly trust a single entity, or there are no servers or group members who have sufficient resources to maintain, generate and distribute keying information. Thus, the distributed solutions of the key management problem have seen considerable attention [6], [13]–[21].

Many contributory schemes are inspired by the Diffie-Hellman (DH) key exchange protocol [36]. To extend two-party DH protocol to the group scenario, the contributory key management schemes in [13], [16]–[18] arrange users in a logical ring or chain structure, and accumulate the keying material while traversing group members one by one. In [19]–[21], logical tree structures are introduced and the number of rounds for establishing the group key is reduced to the logarithm of the group size. Due to their scalability, the tree-based schemes are selected as the basic building blocks to address the hierarchical access control problem in the distributed environments.

A. Tree-based contributory key management schemes

The tree-based scheme in [21] is based on applying two-party DH protocol amongst two subgroups of users. In particular, the users in the first subgroup, who share a common subgroup key K_i, send $\{g^{K_i} \ mod \ p\}$ to the users in the second subgroup; and the users in the second subgroup, who share a common subgroup key K_j, send $\{g^{K_j} \ mod \ p\}$ to the users in the first subgroup. Here, g is the exponential base and p is modular based in the DH protocol [36]. Then, users in two subgroups compute a new key: $K_{ij} = g^{K_i K_j} \ mod \ p$. These two subgroups can be merged into a larger subgroup that shares the common key K_{ij}.

The key tree used in [19] [21] is similar to that in the centralized schemes, as shown in Figure 1. The intermediate keys and the group key are generated from bottom to up as follows. In the first round, users are grouped into pairs and perform two-party DH. Thus, two users form a subgroup. In each of the following rounds, the subgroups formed in the previous round are paired up and each pair of subgroups perform DH and are merged into a larger subgroup with

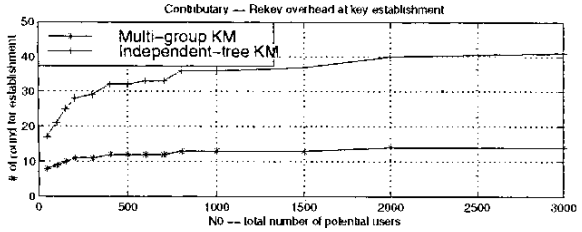


Fig. 13. The number of rounds performed to establish the group key

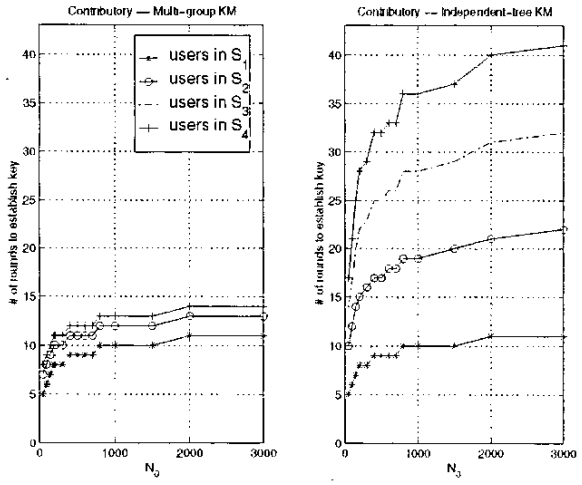


Fig. 14. The number of rounds performed by the users in each SG for key establishment

a shared key. Finally, all users are merged into one group and share the group key K_ϵ. When a user joins or leaves the service, the group key is regenerated in the similar way except that some existing intermediate keys do not need to be recalculated [21] [19]. In the example shown in Figure 1, K_ϵ is established in 4 rounds. When user 16 leaves the service, user 15 generates a new private key and 3 rounds should be performed to compute K_{11}^{new}, K_1^{new}, and K_ϵ^{new}.

B. Contributory Multi-group key management scheme

The multi-group key management schemes can be extended to the contributory environment by using the same key graph construction procedure presented in Section III-B. Similar as in the centralized environments, separate key trees for each DG must be constructed when using existing tree-based contributory schemes [19]–[21], and the multi-group contributory scheme maintains one integrated key graph for all users.

The key establishment protocols are straightforward extensions from the existing protocols in tree-based contributory schemes [19]–[21]. When users join/leave/switch, the keys that need to be recalculated are the same as the keys that need to be updated in the protocol presented in Section III-B. New keys are calculated by applying the DH protocol between subgroups from bottom to up.

For contributory key management schemes, the number of rounds is usually used to measure the communication, computation, and latency [37] associated with key establishment and updating [18]–[20].

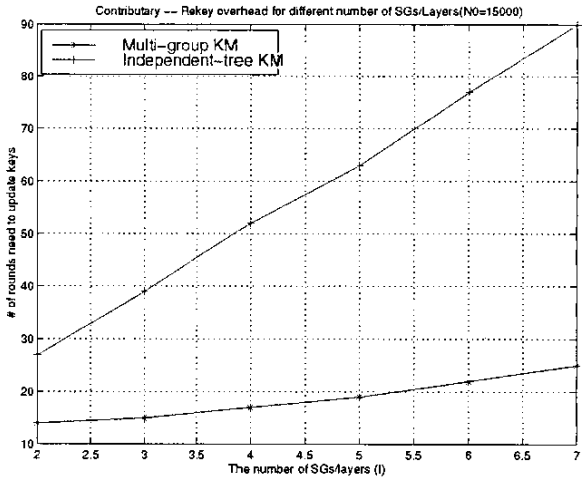


Fig. 15. The number of rounds performed to establish the group key with different number of SGs/Layers

With the same simulation setup as that in Section V-B, the performance of the independent-tree and multi-group contributory key management schemes are compared for different group size. Figure 13 shows the total number of rounds needed to establish the group key. The total number of rounds describes the latency in key establishment [37]. To demonstrate the users' computation overhead, Figure 14 shows the number of rounds performed by the users in each SG. In each round, a user performs two modular exponentiation.

With the same simulation setup as that in Section V-C, the number of rounds for key updating is shown in Figure 15 for different number of layers.

Compared with the tree-based contributory scheme, the multi-group contributory scheme significantly reduces the computation and latency associated with key establishment and update. The advantage of the multi-group contributory scheme is larger when M increases.

VII. Conclusion

This paper presented a multi-group key management scheme that achieves hierarchical access control in secure group communications, where multiple data streams are distributed to group members who have various access privileges. We designed an integrated key graph, as well as the rekey protocol. The proposed scheme achieves the forward and backward security, while allowing users to subscribe/drop the group communications and change access levels. Compared with using the existing tree-based key management schemes that are designed for a single multicast session, the proposed scheme can greatly reduce the overhead associated with key management. In the multi-layer services containing 4 layers, we observed more than 50% reduction in the usage of storage, computation, and communication resources in the centralized environments, and about 50% reduction in the number of rounds needed to establish and update keys in the contributory environments. More importantly, the proposed scheme scales better than the existing tree-based schemes, when the

group applications contain more data streams and require the mechanism to manage more complicated access control policy.

ACKNOWLEDGEMENT

This work was supported in part by the Army Research Office under Award No. DAAD19-01-1-0494.

REFERENCES

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

[2] W. Trappe, J. 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] 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.

[5] D.M. Wallner, E.J. Harder, and R.C. Agee, "Key management for multicast: issues and architectures," Internet Draft Report, Sept. 1998, Filename: draft-wallner-key-arch-01.txt.

[6] 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.

[7] D. McGrew and A. Sherman, "Key establishment in large dynamic groups using one-way function trees," Technical Report 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.

[8] R. Canetti, J. Garay, G. Itkis, D. Miccianancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," *Proc. IEEE INFOCOM'99*, vol. 2, pp. 708–716, March 1999.

[9] A. Perrig, D. Song, and D. Tygar, "ELK, a new protocol for efficient large-group key distribution," in *Proc. IEEE Symposium on Security and Privacy*, 2001, pp. 247 –262.

[10] G. H. Chiou and W. T. Chen, "Secure broadcasting using the secure lock," *IEEE Trans. Software Eng.*, vol. 15, pp. 929–934, Aug 1989.

[11] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proc. ACM SIGCOMM '97*, 1997, pp. 277–288.

[12] S. Banerjee and B. Bhattacharjee, "Scalable secure group communication over IP multicast," *JSAC Special Issue on Network Support for Group Communication*, vol. 20, no. 8, pp. 1511 –1527, Oct. 2002.

[13] I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," *IEEE Transactions on Information Theory*, vol. 28, pp. 714–720, Sep. 1982.

[14] D. G. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A secure audio teleconference system," in *Proceedings on Advances in cryptology*, 1990, pp. 520–528, Springer-Verlag New York, Inc.

[15] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution scheme," *Advances in Cryptology- Eurocrypt*, pp. 275–286, 1994.

[16] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-hellman key distribution extended to group communication," in *Proceedings of the 3rd ACM conference on Computer and communications security*, 1996, pp. 31–37, ACM Press.

[17] M. Steiner, G. Tsudik, and M. Waidner, "CLIQUES: a new approach to group key agreement," in *Proceedings of the 18th International Conference on Distributed Computing Systems*, May 1998, pp. 380 – 387.

[18] M. Steiner, G. Tsudik, . and M. Waidner, "Key agreement in dynamic peer groups," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 11, no. 8, pp. 769–780, Aug 2000.

[19] G. Tsudik Y. Kim, A. Perrig, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *Proceedings of the 7th ACM conference on Computer and communications security*, November 2000.

[20] L.R. Dondeti, S. Mukherjee, and A. Samal, "DISEC: a distributed framework for scalable secure many-to-many communication," in *Proceedings of Fifth IEEE Symposium on Computers and Communications*, 2000, pp. 693 –698.

[21] W. Trappe, Y. Wang, and K.J.R. Liu, "Establishment of conference keys in heterogeneous networks," in *proceedings of IEEE International Conference on Communications*, 2002, vol. 4, pp. 2201 –2205.

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

[23] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Transactions on Computer Systems*, vol. 1, pp. 239–247, March 1983.

[24] C. H. Lin, "Dynamic key management schemes for access control in a hierarchy," *Computer Communications*, vol. 20, no. 15, pp. 1381–1385, 1997.

[25] R. S. Sandhu, "Cryptographic implementation of a tree hierarchy for access control," *Information Processing Letters*, vol. 27, no. 2, pp. 95–98, February 1988.

[26] J.-C. Birget, X. Zou, G. Noubir, and B. Ramamurthy, "Hierarchy-based access control in distributed environments," in *Proceedings of IEEE International Conference on Communications*, June 2001, vol. 1, pp. 229 –233.

[27] Indrakshi Ray, Indrajit Ray, and Natu Narasimhamurthi, "A cryptographic solution to implement access control in a hierarchy and more," in *Proceedings of the seventh ACM symposium on Access control models and technologies*, 2002, pp. 65–73.

[28] M. Eltoweissy and J. Bansemer, "A framework for scalable multicast security with bell-lapedulla confidentiality model," *Journal of Internet Technology: Special Issue on Network Security*, July 2002.

[29] D. Bell and L. La Padula, "Secure computer systems: Mathematical foundations and model," in *MITRE Report, M74-244, MTR 2547 v2*, Nov. 1973.

[30] S. Holeman, G. Manimaran, and J. Davis, "Differentially secure multicasting and its implementation methods," in *Proceedings of International Conference on Computer Communications and Networks*, 2001, pp. 212–217.

[31] A.M. Eskicioglu, S. Dexter, and E.J. Delp, "Protection of multicast scalable video by secret sharing: Simulation results," in *Proceedings of SPIE Security and Watermarking of Multimedia Contents*, Santa Clara, CA, Jan. 2003.

[32] K. Almeroth and M. Ammar, "Collecting and modeling the join/leave behavior of multicast group members in the mbone," in *Proc. High Performance Distributed Computing (HPDC'96), Syracuse, New York*, 1996, pp. 209–216.

[33] K. Almeroth and M. Ammar, "Multicast group behavior in the internet's multicast backbone (MBone)," *IEEE Communications*, vol. 35, pp. 224–229, June 1999.

[34] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, Addison Wesley, 2nd edition, 1994.

[35] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam, "Reliable group rekeying: a performance analysis," *Proc. of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pp. 27 – 38, August 2001.

[36] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. on Information Theory*, vol. 22, pp. 644–654, 1976.

[37] B. Sun, W. Trappe, Y. Sun, and K.J.R. Liu, "A time-efficient contributory key agreement scheme for secure group communications," *Proc. of IEEE International Conference on Communication*, vol. 2, pp. 1159 –1163, 2002.