Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 2497–2509
computer communications
www.elsevier.com/locate/comcom

Efficient group key management for multi-privileged groups

Guojun Wang [a,d], Jie Ouyang [a], Hsiao-Hwa Chen [b,*], Minyi Guo [c,d]

[a] School of Information Science and Engineering, Central South University, Changsha, Hunan Province 410083, China
[b] Department of Engineering Science, National Cheng Kung University, Tainan City 701, Taiwan
[c] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200030, China
[d] School of Computer Science and Engineering, University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan

Available online 5 May 2007

Abstract

Multi-privileged group communications containing multiple data streams have been studied in the traditional wired network environment and the Internet. With the rapid development of mobile and wireless networks and in particular mobile ad-hoc networks (MANETs), the traditional Internet has been integrated with mobile and wireless networks to form the mobile Internet. The multi-privileged group communications can be applied to the mobile Internet. Group users can subscribe to different data streams according to their interest and have multiple access privileges with the support of multi-privileged group communications. Security is relatively easy to be guaranteed in traditional groups where all group members have the same privilege. On the other hand, security has been a challenging issue and is very difficult to handle in multi-privileged groups. In this paper, we first introduce some existing rekeying schemes for secure multi-privileged group communications and analyze their advantages and disadvantages. Then, we propose an efficient group key management scheme called ID-based Hierarchical Key Graph Scheme (IDHKGS) for secure multi-privileged group communications. The proposed scheme employs a key graph, on which each node is assigned a unique ID according to access relations between nodes. When a user joins/leaves the group or changes its access privileges, other users in the group can deduce the new keys using one-way function by themselves according to the ID of joining/leaving/changing node on the graph, and thus the proposed scheme can greatly reduce the rekeying overhead.

Keywords: Multi-privileged group communications; Rekeying; One-way function; Key graph

1. Introduction

With the rapid development of mobile and wireless networks and the mobile ad-hoc networks (MANETs), the traditional Internet has been integrated with mobile and wireless networks to form so-called the mobile Internet. With the rapid development of the mobile Internet and increase of network bandwidth, more and more applications have been found for the mobile Internet. These network applications are based upon unicast or multicast communications. Unicast employs a client-server model, where the server handles the requests of all users and delivers suitable packets to the users via a dedicated point-to-point channel. However, unicast is inefficient if all users request the same data stream. On the contrary, multicast is an efficient method for delivery of data from a source to multiple recipients, such as teleconference, information service and live sports. If compared with unicast, multicast can reduce sender transmission overhead and network bandwidth requirements.

In order to guarantee the security of group communications, it must be ensured that a user that is not a member of a group can not access any communications resource belonging to the group. In order to achieve this requirement, an encryption key, also known as the Session Key (SK) is shared by all legitimate group members [1]. In addition, in order to ensure forward secrecy and backward secrecy [2], the SK should be changed after every join

* Corresponding author. Tel.: +886 910790746; fax: +886 6 2766549.
E-mail addresses: csgjwang@mail.csu.edu.cn (G. Wang), shorange@gmail.com (J. Ouyang), hshwchen@ieee.org (H.-H. Chen), guo-my@cs.sjtu.edu.cn (M. Guo).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.04.019

and leave so that a former group member has no access to current communications and a new member has no access to previous communications.

In traditional group communication schemes [3–8], all members in a group have same level of access privilege. In these schemes, if members hold the decryption key, they can access all the content; otherwise, they can not read anything. In order to improve the scalability of these schemes, reducing the communication overhead and the rekeying overhead are the major design concerns.

However, many group applications contain multiple related data streams and the members have different access privileges. For example:

- Multimedia applications distributing contents in multi-layer coding format. In video broadcasting, users with normal TV receivers can receive the contents with the normal format only, while users with HDTV receivers can receive the contents with both the normal format and the extra information needed to achieve HDTV resolution [9].
- In e-newspaper broadcasting, there are multiple data streams to broadcast the contents of top news, weather forecasts, financial news, stock quotes, and sports news. The service provider also classifies users into several membership groups, such as gold, silver sports, silver finance and basic. In such an application, different membership groups can access different contents [10].

Group key management in multi-privileged group communications is crucial and complicated due to the following two factors. First, the service generally provides multiple data streams and encrypts different data streams using separate SKs [11]. Users can subscribe to one or multiple data streams and should have the corresponding SKs for the purpose of security. The challenge is how to manage these keys while ensuring that no users can access the key and data beyond their privileges. Second, not only the users can join or leave the group at will, but also the users can change their access privileges according to their interest at any time. Hence, a group key management scheme should be flexible enough to accommodate users' join/leave/change requirements. These challenging issues raise the critical problem of how we can efficiently manage the keys when users join/leave/change their access privileges. In this paper, we will present a new multi-privileged group rekeying scheme. The proposed scheme employs a key graph to manage SKs and exploits a one-way function to update the keys [12] in order to reduce the rekeying messages in the join/leave and change operations.

The rest of the paper is outlined as follows. In Section 2, we describe the service model and logical key hierarchy. In Section 3, we introduce some existing rekeying schemes for multi-privileged group communications. In Section 4, we propose a novel rekeying scheme. Finally, we conclude this paper in Section 5.

2. Preliminaries

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, and then describe the basic idea of key tree in group communications containing single data stream.

2.1. System descriptions

2.1.1. One-dimensional data stream

Let $\{r_1, r_2, \ldots\}$ denote the set of resources in a group communication system. In such a system, each resource corresponds to a data stream.

A Data Group (DG) consists of a set of users that can access to a particular resource. Obviously, the DGs can have overlapped membership because users may subscribe to multiple resources. The DGs are denoted by D_1, $D_2, \ldots, D_M$, where M is the total number of the DGs. A Service Group (SG) consists of a set of users who can access the exactly same set of resources. The users in each SG have same access privilege. The SGs have non-overlapped membership. The SGs are denoted by S_1, $S_2, \ldots, S_I$, where I is the total number of SGs. In order to describe the access relations in group communications, t^i_m is defined as:

$$t^i_m = \begin{cases} 1, & \text{the users in SG } S_i \text{ subscribs to resource } r_m \\ 0, & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, I$, and $m = 1, \ldots, M$. In addition, S_0 is defined as a virtual SG, which represents users who do not participate in any group communications.

The following Example 1 and Example 2 are two typical applications of multimedia group communications [9].

Example 1. Multimedia applications that distribute contents in multi-layer format. The access relations are illustrated in Table 1.

Example 2. Multicast programs containing several related services, as shown in Table 2.

2.1.2. Multi-dimensional data stream

An MPEG-4 FGS video frame, supporting T PSNR service levels and M bitrate service levels, is divided into $T \times M$ different two-dimensional units [13]. A single tile JPEG 2000 frame can support four-dimensional scalability

Table 1
Multi-layer service groups and their access relations

Access relation	D_1 (r_1:base layer)	D_2 (r_2: enhancement layer 1)	D_3 (r_3: enhancement layer 2)
$S_{\{001\}}$	√		
$S_{\{011\}}$	√	√	
$S_{\{111\}}$	√	√	√

Table 2
Cellular phone service groups and their access relations

Access relation	D_1 (r_1: news)	D_2 (r_2: stock quote)	D_3 (r_3: traffic/ weather)
$S_{\{001\}}$	√		
$S_{\{010\}}$		√	
$S_{\{100\}}$			√
$S_{\{011\}}$	√	√	
$S_{\{101\}}$	√		√
$S_{\{110\}}$		√	√
$S_{\{111\}}$	√	√	√

Table 3
Data streams in video

	Lowest quality	Middle quality	Full quality
Resolution level 0	r_{00}	r_{01}	r_{02}
Resolution level 1	r_{10}	r_{11}	r_{12}

innately: resolution, quality, component and precinct. That is, data streams are scalable in multiple dimensions.

Suppose there is a scalable video with 2 resolution levels and 3 quality layers. The group has 6 data streams, as shown in Table 3.

Users that subscribe to r_{ij} can access a scalable unit of resolution i and quality layer j. Similarly, a Service Group (SG) defines a set of users who receive the same set of data streams. When an SG has a privilege to access the video stream at resolution 0 with full quality, the users in this SG can receive r_{00}, r_{01} and r_{02}.

Each data stream needs one unique SK to encrypt the data. In order to achieve access control, the users in each DG share an SK. If a user subscribes to multiple data streams, it needs an SK for each data stream. When a user joins or leaves the group, the SKs the user holds must be changed. However, when a user switches between SGs, it is unnecessary to change SKs for data streams to which the user is still subscribing.

2.2. Logical key hierarchy

Logical Key Hierarchy (LKH) [14] scheme provides an efficient and secure mechanism to manage the keys and to coordinate the key update. The LKH employs a hierarchical tree whose root node is associated with a group key and whose leaf nodes are individual keys of all users in the group. The intermediate nodes correspond to Key Encryption Key (KEK). Each user in the group holds a set of keys on the path from its leaf to the root. Consider a multicast group with six users. The KDC constructs a hierarchy of keys as shown in Fig. 1. The root node k_{1-6} is group key, and the user u_2 owns k_2, k_{1-2}, k_{1-4} and k_{1-6}.

Reference [15] presents an efficient scheme to update the key tree when users dynamically join or leave. Each key contains a unique key ID, a version field and a revision field, as shown in Fig. 2. When a user wants to join

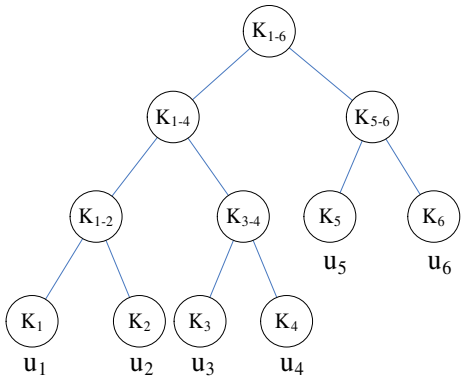


Fig. 1. Logical key hierarchy.

Fig. 2. Structure of a key.

the group, the KDC assigns a leaf node to represent the new user, and increases the revision numbers of all keys on the path from the leaf node to the root by passing the keys through a one-way function. When a user notices the revision change in ordinate data packet, the user updates the keys with new revision number from old key using the one-way function. In this case, the KDC needs only to send one rekeying message to the new user. In addition, when a user wants to leave the group, the KDC updates the keys that are held by the leaving user. The number of rekeying messages for a user leaving increases linearly with the logarithm of group size.

2.3. Requirements of the rekeying schemes for multi-privileged group communications

Obviously, it is impossible to directly apply the logical key hierarchy to multi-privileged group communications. In order to achieve hierarchical access control, a simple method is to construct a separate key tree for each DG. The leaves are associated with the users in each DG. The method is very simple and it is easy to manage the keys and the communications. But the method can bring redundancy because DGs have overlapped membership and doesn't scale well when the number of data streams increases.

Therefore, the rekeying schemes for multi-privileged group communications should provide security, flexibility and scalability.

- *Security.* Each user may subscribe to one or multiple resources. The rekeying schemes must prevent the user from accessing any data before he joins or after he leaves the group. In addition, the rekeying schemes must ensure that the user can't access the data that he doesn't subscribe to.

- *Flexibility.* Besides joining or leaving the group, the users may change their access privileges, which can be considered that the users switch between different SGs. Because of the dynamics of users, the rekeying schemes need to support users' join/leave/ switch at any time.
- *Scalability.* When a new SG joins the group communications or an SG in a group decomposes, it should not lead to the reconstruction of the structure for group key management. In other words, it should support the dynamic service group formation and decomposition.

3. The existing group key management schemes

In order to eliminate the redundancy because of overlapped membership among DGs, Sun and Liu [9] proposes a Multi-Group Key Management Scheme (MGKMS). The MGKMS scheme employs an integrated key graph to manage the keys when a user joins/leaves/switches. The key graph is constructed as follows:

(1) The KDC constructs an SG-subtree for each SG. The root of the SG_i-subtree is the SG key K_i^S. The leaves are the users in the SG S_i.
(2) The KDC constructs a DG-subtree for each DG. The root of the DG_m-subtree is the DG key K_m^D. The leaves are the SG keys in which the users can access the resource r_m.
(3) The KDC generates the key graph by connecting the leaves of the DG-subtrees and the roots of the SG-subtrees.

The procedure of constructing the integrated key graph is illustrated in Fig. 3. Suppose each SG has 4 users. Each user in S_i holds a set of keys on the paths from the leaf to the root of the DG-subtree of $\{D_m, \forall m : t_m^i = 1\}$.

Here, a switching user changes the SG from S_i to S_j. ϕ_i denotes a set of keys that the user holds in S_i, and ϕ_j denotes a set of keys that the user holds in S_j. The rekeying algorithm consists of two steps as follows:

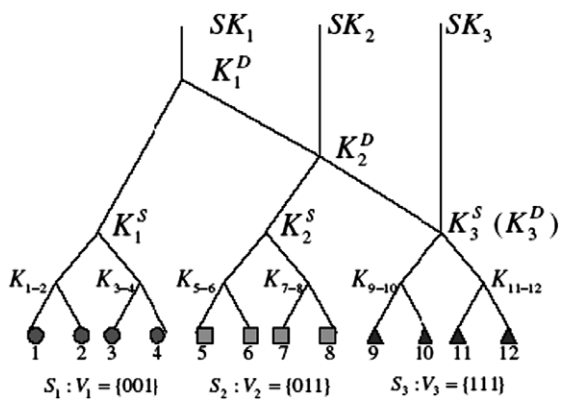


Fig. 3. Multi-group key management graph construction.

(1) The KDC updates the keys in $\bar{\phi}_i \cap \phi_j$ through a one-way function and increases the revision numbers of these keys. When users notice that the revision numbers of keys they hold increase, they compute the new keys using the same one-way function.
(2) The KDC updates the keys in$\phi_i \cap \bar{\phi}_j$, increases their version numbers and sends the new keys encrypted with their children keys to the users.

The MGKMS scheme can achieve the forward and backward secrecy when users join/leave the group or switch between different SGs. Compared with the tree-based group key management scheme in multicast communications containing single data stream, it can greatly reduce the storage, computation and communication overheads. However, if there are complicated relations between SGs and DGs, the merging key graph step will also be complicated. In addition, the scheme can't flexibly deal with formation and decomposition of SGs.

In many applications, users and data streams both form a partially ordered hierarchy, but the above MGKMS scheme only considers the former. Therefore, a Hierarchical Access Control Key Management Scheme (HACKMS) [10] is proposed, which considers both partially ordered users and partially ordered data streams. The HACKMS scheme presents an algorithm to construct a key graph based the Directed Acyclic Graph (DAG) of SGs and the DAG of resource groups [16]. The algorithm traverses the unified DAG in breath-first search and constructs the key graph from bottom to top. The HACKMS scheme considers the partially ordered relationship among data streams, and the number of auxiliary keys of DGs and SKs is less than that of the MGKMS scheme. So, compared with the MGKMS scheme, it reduces the storage and rekeying overheads at key server and users. But the construction of key graph is a little bit complicated, because it must first form a unified DAG for SGs and resource groups. In addition, if the partially ordered relationship of data streams is changed, the key graph must be reconstructed.

The above schemes only support one-dimensional data streams. However, in some applications, there are multi-dimensional data streams [17]. Therefore, Dynamic Access Control Scheme (DACS) [18] is proposed, which supports not only one-dimensional data streams, but also multi-dimensional data streams. It don't need to construct DG-subtree, the root of each SG-subtree is associated with the SKs of the scalable streams that the users in this SG can access.This scheme isn't only suitable for the multi-dimensional data streams but also flexible for the dynamic service group formation and decomposition. It scales well when the new SG is formed. In addition, the storage overhead and the rekeying overhead are less than those in the MGKMS scheme because of no auxiliary keys in DG-subtrees.

In Distributed Key Management Scheme (DKMS) [19], every SG maintains an SG server to be used to manage all the users in the SG. The DKMS proposed a structure that includes two parts: DG part and SG part. The DG part

consists of all SG servers and is used to manage those servers. The SG part includes an SG server and all users in this SG. Compared with the MGKMS scheme, both the storage overhead and the rekeying overhead can be reduced. And this scheme supports the service group formation and decomposition. However, compared with the DACS scheme, forming a new SG in DKMS is more complicated because the group key of SGSG shared by all SG servers needs to be updated.

4. Our proposed scheme

We propose an ID-based Hierarchical Key Graph Scheme (IDHKGS) to manage multi-privileged group communications. The proposed scheme employs a key graph [9] and each node is assigned an ID to uniquely identify a key.

The key graph contains two types of nodes: *u*-nodes which contain individual keys and *k*-nodes which contain SG keys, DG keys and auxiliary keys. The proposed scheme differs from the MGKMS scheme in two aspects, i.e., the identification of a key and the rekeying operation. In the proposed scheme, as long as a user knows the IDs of another user's *u*-node in the group, it can deduce the IDs of *k*-nodes on the paths from the *u*-node to the SK nodes which contain SKs. In addition, we update the keys using a one-way function for the old users to compute the new keys by themselves when a user joins/leaves the group or switches between different SGs.

4.1. Identification of a key

In our proposed scheme, the key graph contains two parts, the SG part and the DG part. The SG part is composed of all SG-subtrees, and the DG part is composed of all SK nodes and the *k*-nodes between the SG *k*-nodes and the SK nodes on the key graph. The SGs are denoted by $S_2, S_3, \ldots, S_i, \ldots$, where i is a prime number. The server assigns two integers as the ID of each node on the key graph. In each SG-subtree, a node is identified by the SG $i(i = 2, 3, 5, \ldots)$ to which the node belongs and by the position $m(m \geqslant 0)$ which is numbered from the root of its SG-subtree in a top–down and left–right order. The node $\langle i, 0\rangle$ is the root of the S_i-subtree. We observe that the IDs of a node and its parent node have the following simple relationship: $k_{\langle i,\lfloor (m-1)/2 \rfloor\rangle}$ is the parent node of $k_{\langle i,m\rangle}$.

In the DG part, if a node has two children nodes $\langle j_1, n_1\rangle$ and $\langle j_2, n_2\rangle$, the node is identified by j that is the least common multiple of j_1 and j_2 and by $n(n = \max(j_1, j_2))$. If a node only has one child node $\langle j_1, n_1\rangle$, such as the SK node, the node is identified by j_1 and by -1. Fig. 4 illustrates the IDs of nodes in Fig. 3 of Section 3.

A user in SG S_i holds a set of keys on the paths from the leaf to the root of the DG-subtree of $\{D_m, \forall m : t_m^i = 1\}$. When a user in a group knows the ID of u_5's *u*-node, $\langle 3, 3\rangle$, then this user can deduce that user u_5 holds $k_{\langle 3,1\rangle}$, $k_{\langle 3,0\rangle}$, $k_{\langle 15,5\rangle}$, $k_{\langle 15,-1\rangle}$, $k_{\langle 30,15\rangle}$, $k_{\langle 30,-1\rangle}$.

In order to maintain forward secrecy and backward secrecy, a rekeying operation is executed when a user joins/leaves a group or switches between SGs.

4.2. Rekeying algorithm

In order to maintain the forward secrecy and backward secrecy, a rekeying operation is executed when a user joins/leaves a group or switches between SGs.

4.2.1. Single user join

When a user joins or switches between SGs, the IDs of some *u*-nodes in the SG part will be changed and users

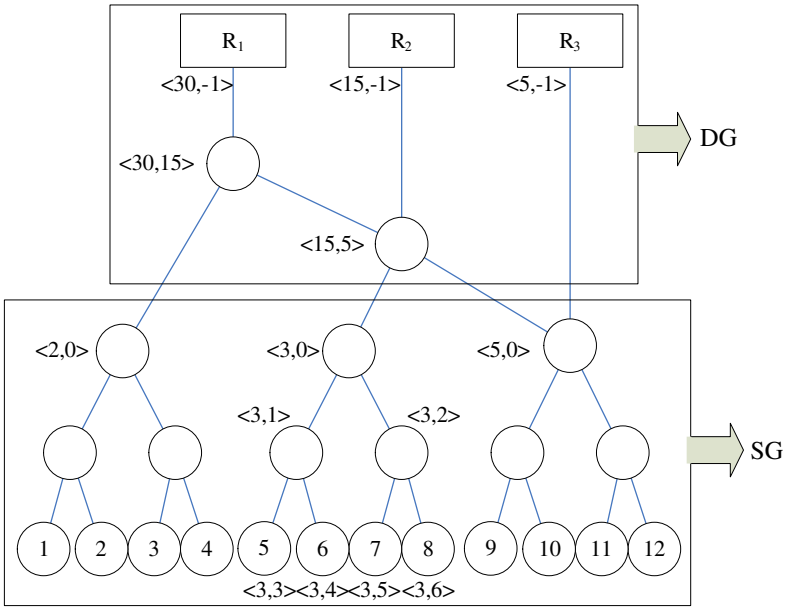


Fig. 4. Illustration of key identification.

should know the up-to-date IDs of their u-nodes. In [20], the authors prove that a user can deduce its current ID by knowing its old ID and the maximum ID of the current k-nodes. Then, a user in SG can compute its current ID using the same method in our proposed scheme.

Suppose that a user u requests to join S_i. The server inserts it at the end of one of the shortest paths of the S_i-subtree, assigns $\langle i, m\rangle$ to the new u-node, broadcasts the ID of the new u-node, $\langle i, m\rangle_J$ (where J is named after the join operation of the joining u-node), and the maximum ID of the current k-nodes in S_i, $\langle i, n_k\rangle$. When a user receives the broadcast messages, the users in the group can deduce the new keys using a one-way function so that $k' = f(k)$ (where k' denotes the updated version of key k). If $k_{\langle i,n\rangle}$ is a newly created one, the new key is $k'_{\langle i,n\rangle} = f(k_{\langle i,l\rangle} \oplus k_{\langle i,0\rangle})$, where $k_{\langle i,l\rangle}$ is the key of the spitted u-node. The rekeying algorithm for single user join is illustrated in Fig. 5.

We explain the join operation of user u_{13}, as shown in Fig. 6. When a user u_{13} joins the group, a new u-node is created to hold u_{13}'s individual key. The server broadcasts $\langle 3, 8\rangle_J$ (the ID of the new user u_{13}'s u-node) and $\langle 3, 3\rangle$ (the maximum ID of the k-node in S_3 after u_{13} joins the group).

According to the IDs, the users in the group can deduce that $k_{\langle 3,1\rangle}$, $k_{\langle 3,0\rangle}$, $k_{\langle 15,5\rangle}$, $k_{\langle 15,-1\rangle}$, $k_{\langle 30,15\rangle}$ and $k_{\langle 30,-1\rangle}$ need to be updated and$k_{\langle 3,3\rangle}$ is the newly created one. Then, the users compute the new key values using a one-way function by themselves. The new keys are:

$$
\begin{aligned}
&k'_{\langle 3,1\rangle} = f(k_{\langle 3,1\rangle}), && k'_{\langle 3,0\rangle} = f(k_{\langle 3,0\rangle}),\\
&k'_{\langle 15,5\rangle} = f(k_{\langle 15,5\rangle}), && k'_{\langle 15,-1\rangle} = f(k_{\langle 15,-1\rangle}),\\
&k'_{\langle 30,15\rangle} = f(k_{\langle 30,15\rangle}), && k'_{\langle 30,-1\rangle} = f(k_{\langle 30,-1\rangle}).\\
&k'_{\langle 3,3\rangle} = f(k_{\langle 3,7\rangle} \oplus k_{\langle 3,0\rangle}).
\end{aligned}
$$

Finally, the server only needs to encrypt all new keys for the newly joining user.

$$
s \rightarrow u_{13} : \{k'_{\langle 3,3\rangle}, k'_{\langle 3,1\rangle}, k'_{\langle 3,0\rangle}, k'_{\langle 15,5\rangle}, k'_{\langle 15,-1\rangle}, k'_{\langle 30,15\rangle}, k'_{\langle 30,-1\rangle}\}_{k_{\langle 3,8\rangle}}
$$

```
Input: <i, m>J, <i, nk>;
Output: Updated keys.
if (h == i) && (nk != k) && the user holds k<i, k+1> {
    compute the new ID of u-node;
    k'<i, k+1> = f(k<i, m-1> ⊕ k<i, 0>);
    k = nk;
    m = ⌊(m−1)/2⌋;
}
while (m>0) {
    m = ⌊(m−1)/2⌋;
    if (the user holds k<i, m>)
        k'<i, m> = f(k<i, m>);
}
Check k<j, n> the user holds where j isn't a prime number;
if (j mod i ==0)
    k'<j, n> = f(k<j, n>);
```

Fig. 5. Rekeying algorithm for single user join.

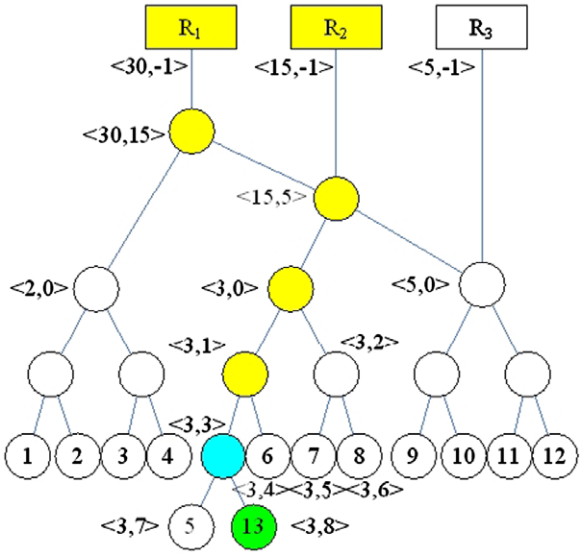


Fig. 6. Key graph after u_{13} joins.

4.2.2. Single user leave

Suppose that a user u requests to leave S_i. Then all the keys the user u holds must be updated. The server broadcasts the ID of the leaving u-node that is $\langle i, n\rangle_L$ (where L is named after the leave operation of the leaving u-node). The users in the group deduce the new keys through a one-way function such that $k' = f(k \oplus k_1)$ where k_1 is one of the auxiliary keys that is not on the leave paths. The server only needs to encrypt and send these new keys to the users who can not deduce them. The rekeying algorithm for single user leave is illustrated in Fig. 7.

We explain the leave operation of user u_8, as shown in Fig. 8. The server removes the u-node of u_8. The server broadcasts $\langle 3, 6\rangle_L$ (the ID of the leaving user u_8's u-node). The users in S_3 can deduce that $k_{\langle 3,2\rangle}$ and $k_{\langle 3,0\rangle}$ need to be

```
Input: <i, m>L;
Output: Updated keys.
While (m>0){
    t = ⌊(m−1)/2⌋;
    if (m mod 2 ==1) && the user holds k<i, t*2+2>
        k'<i, t> = f(k<i, t> ⊕ k<i, t*2+2>);
    if (m mod 2 != 1) && the user holds k<i, t*2+1>
        k'<i, t> = f(k<i, t> ⊕ k<i, t*2+1>);
    m = t;
}
Check k-nodes in the DG part that the user holds, such as k<j, n>, where j
isn't a prime number;
if ( j mod i ==0) && (n mod i != 0)
    if n is a prime number && the user holds k<n, 0>
        k'<j, n> = f(k<j, n> ⊕ k<n, 0>);
    if n isn't a prime number && the user holds k<n, l> where l!=-1,0
        k'<j, n> = f(k <j, n> ⊕ k<n, l>);
if ( j mod i ==0) && (j/n mod i != 0)
    if j/n is a prime number && the user holds k<j/n, 0>
        k'<j, n> = f(k<j, n> ⊕ k<j/n, 0>);
    if j/n isn't a prime number && the user holds k< j/n, l> where l!=-1,0
        k'<j, n> = f(k<j, n> ⊕ k<j/n, l>);
```

Fig. 7. Rekeying algorithm for single user leave.

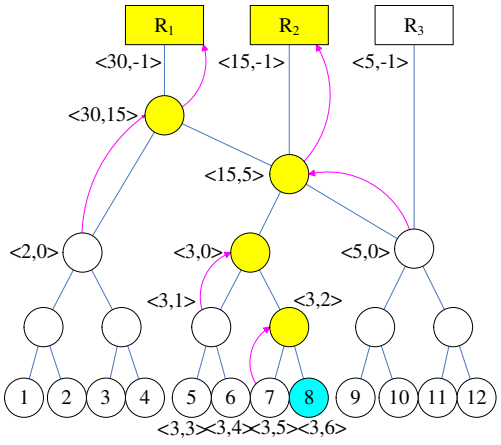


Fig. 8. Key graph after u_8 leaves.

updated and $k_{\langle 3,5\rangle}$ and $k_{\langle 3,1\rangle}$ are chosen to compute $k_{\langle 3,2\rangle}$ and $k_{\langle 3,0\rangle}$, respectively.

$$k'_{\langle 3,2\rangle} = f(k_{\langle 3,2\rangle} \oplus k_{\langle 3,5\rangle}), k'_{\langle 3,0\rangle} = f(k_{\langle 3,0\rangle} \oplus k_{\langle 3,1\rangle}).$$

In the DG part, the users deduce that $k_{\langle 15,5\rangle}$, $k_{\langle 30,15\rangle}$ $k_{\langle 15,-1\rangle}$ and $k_{\langle 30,-1\rangle}$ need to be updated and the new keys except the SKs are:

$$k'_{\langle 15,5\rangle} = f(k_{\langle 15,5\rangle} \oplus k_{\langle 5,0\rangle}), k'_{\langle 30,15\rangle} = f(k_{\langle 30,15\rangle} \oplus k_{\langle 2,0\rangle}).$$

The server needs to encrypt and send the new keys to the users that can not deduce them.

$$s \rightarrow u_7 : \{k'_{\langle 3,0\rangle}\}_{k'_{\langle 3,2\rangle}}.$$

$$s \rightarrow u_5 - u_7 : \{k'_{\langle 15,5\rangle}\}_{k'_{\langle 3,0\rangle}},$$

$$s \rightarrow u_5 - u_7, u_9 - u_{12} : \{k'_{\langle 30,15\rangle}\}_{k'_{\langle 15,5\rangle}}.$$

In addition, the server computes the new SKs,

$$k'_{\langle 30,-1\rangle} = f(k_{\langle 30,-1\rangle} \oplus k'_{\langle 30,15\rangle}), k'_{\langle 15,-1\rangle} = f(k_{\langle 15,-1\rangle} \oplus k'_{\langle 15,5\rangle}),$$

and it encrypts and sends them to the users.

$$s \rightarrow u_1 - u_7, u_9 - u_{12} : \{k'_{\langle 30,-1\rangle}\}_{k'_{\langle 30,15\rangle}},$$

$$s \rightarrow u_5 - u_7, u_9 - u_{12} : \{k'_{\langle 15,-1\rangle}\}_{k'_{\langle 15,5\rangle}}.$$

4.2.3. Single user switch

In multi-privileged group communications, the users can flexibly change their access privileges according to their interest at any time. That is, the users are able to switch between different SGs.

Suppose that a user wants to switch from S_i to S_j, which can be considered as that the user first leaves S_i and then joins S_j. The server broadcasts the IDs of leaving/joining node and the maximum ID of the current k-nodes in S_j, $\langle i, n\rangle_{SL}$, $\langle j, m\rangle_{SJ}$ and $\langle j, n_k\rangle$, where SL is named after the leave operation of the switching user u-node in S_i and SJ is named after the join operation of the switching user u-node in S_j. The users deduce the new keys using one-way function that are similar to the join operation and the leave

```
Input: <i, n>SL, <j, m>SJ, <j, nk>;
Output: Updated keys.
if (h == j) && (nk != k) && the user holds k<j, k+1> {
      compute the new ID of u-node;
      k'<j, k+1> = f(k<j, k+1> ⊕ k<j, 0>);
      k = nk;
      m = ⌊(m − 1)/2⌋;
}
while (m>0 || n>0) {
    t = ⌊(m − 1)/2⌋;
    if (n mod 2 ==1) && the user holds k<i, t*2+2>
        k'<i, t> = f(k<i, t> ⊕ k<i, t*2+2>);
    if (n mod 2 != 1) && the user holds k<i, t*2+1>
        k'<i, t> = f(k<i, t> ⊕ k<i, t*2+1>);
    n = t;
    m = ⌊(m − 1)/2⌋;
    if user holds k<j, m>
        k'<j, m> = f(k<j, m>);
}
Check k-nodes in the DG part that the user holds, such as k<x, y>, where x
isn't a prime number;
if (x mod i ==0) && (x mod j !=0){
    if (y mod i != 0)
        if y is a prime number && the user holds k<y, 0>
            k'<x, y> = f(k<x, y> ⊕ k<y, 0>);
        if y isn't a prime number && the user holds k<y, l> where l!=-1
            k'<x, y> = f(k <x, y> ⊕ k<y, l>);
    if (x/y mod i != 0)
        if x/y is a prime number && the user holds k<x/y, 0>
            k'<x, y> = f(k<x, y> ⊕ k<x/y, 0>);
        if x/y isn't a prime number && the user holds k<x/y, l> where l!=-1
            k'<x, y> = f(k<x, y> ⊕ k<x/y, l>);
}
if (x mod i!=0) && (x mod j ==0)    k'<x, y> = f(k<x, y>);
```

Fig. 9. Rekeying algorithm for single user switch between SGs.

operation. The rekeying algorithm for single user switch between SGs is illustrated in Fig. 9.

We explain the single user switch operation in Fig. 10. A user u_8 wants to switch from S_2 to S_1. A new u-node inS_1 is created to hold u_8's individual key. The server broadcasts the ID of the switching user's old u-node and new u-node and the maximum ID of the k-node in S_1 after u_8 switches to S_1, $\langle 3, 6\rangle_{SL}$, $\langle 2, 6\rangle_{SJ}$ and $\langle 2, 3\rangle$. The users deduce that-$k_{\langle 2,1\rangle}$, $k_{\langle 2,0\rangle}$,$k_{\langle 3,2\rangle}$, $k_{\langle 3,0\rangle}$ need to be updated and $k_{\langle 2,1\rangle}$ is a newly created one. The new keys are computed as follows:

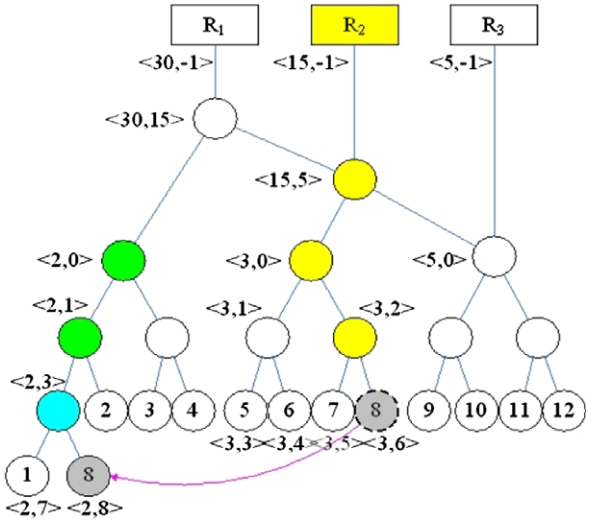


Fig. 10. Key graph after u_8 switches from S_2 to S_1.

$$k'_{\langle 2,1\rangle} = f(k_{\langle 2,1\rangle}), \quad k'_{\langle 2,0\rangle} = f(k_{\langle 2,0\rangle}),$$
$$k'_{\langle 2,3\rangle} = f(k_{\langle 2,7\rangle} \oplus k_{\langle 2,0\rangle}), \quad k'_{\langle 3,2\rangle} = f(k_{\langle 3,2\rangle} \oplus k_{\langle 3,5\rangle}),$$
$$k'_{\langle 3,0\rangle} = f(k_{\langle 3,0\rangle} \oplus k_{\langle 3,1\rangle}).$$

In the DG part, the users deduce that $k_{\langle 15,5\rangle}$ and $k_{\langle 15,-1\rangle}$ need to be updated,

$$k'_{\langle 15,5\rangle} = f(k_{\langle 15,5\rangle} \oplus k_{\langle 5,0\rangle}).$$

The server encrypts $k'_{\langle 3,0\rangle}$ and $k'_{\langle 15,5\rangle}$, and sends them to the users who can not deduce them.

$$s \to u_7 : \{k'_{\langle 3,0\rangle}\}_{k'_{\langle 3,2\rangle}}, s \to u_5 - u_7 : \{k'_{\langle 15,5\rangle}\}_{k'_{\langle 3,0\rangle}}.$$

Finally, the server computes the new SKs,

$$k'_{\langle 15,-1\rangle} = f(k_{\langle 15,-1\rangle} \oplus k'_{\langle 15,5\rangle}),$$

and sends them to the corresponding users.

$$s \to u_5 - u_7, u_9 - u_{12} : \{k'_{\langle 15,-1\rangle}\}_{k'_{\langle 15,5\rangle}}.$$

4.2.4. Batch update operation

If users join, leave or switch frequently, the individual rekeying operations, that is, rekeying after each join, leave or switch request, has very large rekeying overhead. In periodic batch rekeying [21], the server collects all join, leave and switch requests. At the end of each rekeying period of time, the server processes all requests, generates new keys and sends them to the corresponding users. In our proposed scheme, when the SG-subtrees that the users want to join or switch to become full binary trees, the server splits nodes after the rightmost k-node at the highest level to accommodate the extra joins. Firstly, the server labels the k-nodes, and the process consists of 3 steps as follows:

(1) The server removes the u-nodes of leaving users and switching users in the SGs from which the users switch, labels all k-nodes on the leave paths as LEAVE.
(2) The server labels the newly created k-nodes as NEW, labels all k-nodes on the join paths as JOIN.
(3) To the switching users, the server labels the k-nodes which the users hold originally but do not hold after the switch operation as LEAVE, labels the k-nodes which the users do not hold originally but hold after the switch operation as JOIN.

After the key graph is labeled, the server needs to broadcast the IDs of all joining users, leaving users and switching users, $\langle a_i, b_i\rangle_L$, $\langle c_j, d_j\rangle_J$, $\langle e_p, f_p\rangle_{SL_p}$, $\langle g_p, h_p\rangle_{SJ_p}$, and the IDs of the k-nodes in some SGs that the users want to join and switch to. According to the broadcast IDs, the users label the k-nodes which they hold as JOIN, LEAVE or NEW. Then, the users can deduce the new key for all labeled k-nodes according to following three cases.

Case 1: As shown in Fig. 11, if the k-node is labeled as LEAVE, whether or not it is labeled as JOIN, the users compute the new key value as follows.

1) i is a prime number. The operation is similar to the leave operation. The new key is $k'_{\langle i,j\rangle} = f(k_{\langle i,j\rangle} \oplus k_{\langle i,l\rangle})$ when $k_{\langle i,l\rangle}$ is not labeled. If both two children nodes are labeled, the server computes the new key, $k'_{\langle i,j\rangle} = f(k_{\langle i,j\rangle} \oplus k'_{\langle i,j*2+1\rangle})$, encrypts and sends it to the users.
2) i is not a prime number. $k_{\langle i,j\rangle}$ has two children nodes, and $k_{\langle i/j,l_2\rangle}$. If all a_i and e_p are not common factors of j, the new key is $k'_{\langle i,j\rangle} = f(k_{\langle i,j\rangle} \oplus k_{\langle j,l_1\rangle})$ ($l_1 = 0$ when j is a prime number, otherwise l_1!=-1,0). If all a_i and e_p are not common factors of i/j, the new key is $k'_{\langle i,j\rangle} = f(k_{\langle i,j\rangle} \oplus k_{\langle i/j,l_2\rangle})$ ($l_2 = 0$ when i/j is a prime number, otherwise $l_2! = -1, 0$). When the two children nodes are labeled, the server chooses a new child node key to compute $k'_{\langle i,j\rangle}$.

Case 2: As shown in Fig. 12, the new key is $k'_{\langle i,j\rangle} = f(k_{\langle i,j\rangle})$.

Case 3: As shown in Fig. 13, the k-node is newly created, the new key is $k'_{\langle i,j\rangle} = f(k_{\langle i,j*2+1\rangle} \oplus k_{\langle i,0\rangle})$.

Fig. 14 shows an example of the batch update operation. During a rekeying period of time, u_1 leaves the group, u_{13} joins S_2, and u_5 switches from S_2 to S_3. The server removes the u-node $u_{\langle 2,3\rangle}$, broadcasts their IDs and the maximum IDs of k-nodes in some SGs which some users want to join and switch to, $\langle 2, 3\rangle_L$, $\langle 3, 3\rangle_{SL_1}$, $\langle 5, 8\rangle_{SJ_1}$, $\langle 3, 3\rangle_J$, and $\langle 3, 2\rangle$, $\langle 5, 3\rangle$. The server and the users label the k-nodes. The users can compute the new keys according to the above method.

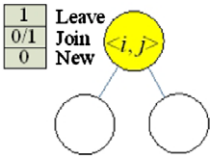


Fig. 11. $k_{\langle i,j\rangle}$ is labeled as LEAVE.

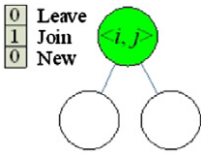


Fig. 12. $k_{\langle i,j\rangle}$ is labeled as JOIN.

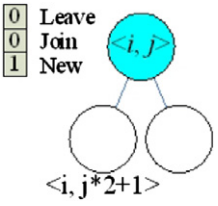


Fig. 13. $k_{\langle i,j\rangle}$ is labeled as NEW.

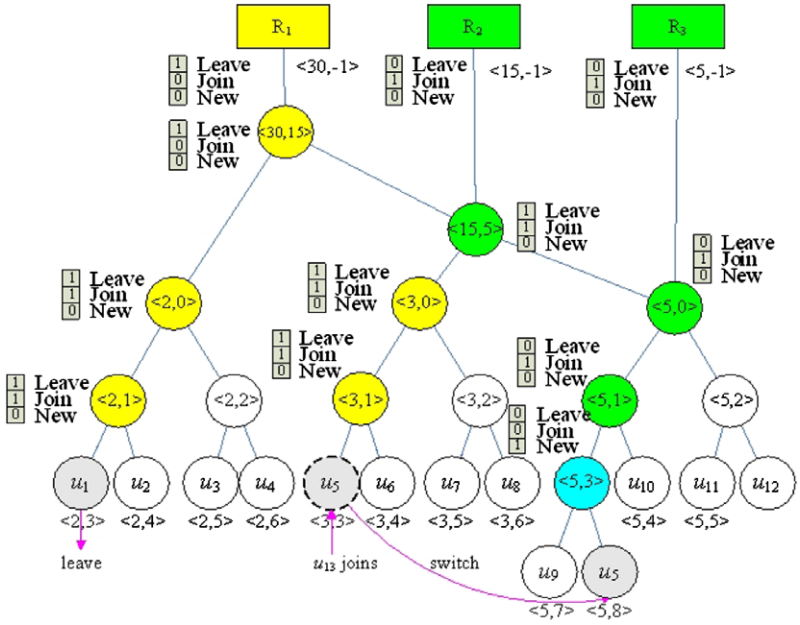


Fig. 14. A batch update example.

$$k'_{\langle 5,1\rangle} = f(k_{\langle 5,1\rangle}),\ k'_{\langle 5,0\rangle} = f(k_{\langle 5,0\rangle}),\ k'_{\langle 5,-1\rangle} = f(k_{\langle 5,-1\rangle}),$$
$$k'_{\langle 15,5\rangle} = f(k_{\langle 15,5\rangle}),\ k'_{\langle 15,-1\rangle} = f(k_{\langle 15,-1\rangle}),$$
$$k'_{\langle 5,3\rangle} = f(k_{\langle 5,7\rangle} \oplus\ k_{\langle 5,0\rangle}),$$
$$k'_{\langle 3,1\rangle} = f(k_{\langle 3,1\rangle} \oplus\ k_{\langle 3,4\rangle}),\ k'_{\langle 3,0\rangle} = f(k_{\langle 3,0\rangle} \oplus k_{\langle 3,2\rangle}),$$
$$k'_{\langle 2,1\rangle} = f(k_{\langle 2,1\rangle} \oplus k_{\langle 2,4\rangle}), k'_{\langle 2,0\rangle} = f(k_{\langle 2,0\rangle} \oplus k_{\langle 2,2\rangle}),$$
$$k'_{\langle 30,15\rangle} = f(k_{\langle 30,15\rangle} \oplus k'_{\langle 2,0\rangle}).$$

To the SK, $k_{\langle 30,-1\rangle}$, the server computes the new key,

$$k'_{\langle 30,-1\rangle} = f(k_{\langle 30,-1\rangle} \oplus k'_{\langle 30,15\rangle}).$$

Finally, the server encrypts the new keys that some users can not compute by themselves and sends them to these users.

$$s \to u_6 : \{k'_{\langle 3,0\rangle}\}_{k'_{\langle 3,1\rangle}}, s \to u_2 : \{k'_{\langle 2,0\rangle}, k'_{\langle 30,15\rangle}\}_{k'_{\langle 2,1\rangle}},$$
$$s \to u_5 - u_{13} : \{k'_{\langle 30,15\rangle}\}_{k'_{\langle 15,5\rangle}}, s \to u_2 - u_{13} : \{k'_{\langle 30,-1\rangle}\}_{k'_{\langle 30,15\rangle}}.$$

5. Theoretical analysis and simulation studies

5.1. Security analysis

In our proposed scheme, the attacker model is divided into two kinds of adversaries. One kind of adversary does not belong to a group, and the other is a member of a group.

For the former, suppose that the adversaries can eavesdrop all traffic but they do not hold any keys in the key graph since they do not belong to the group. So, they can not get the SKs and decrypt the data messages.

For the latter, when the adversaries join the group and hold all keys on their paths, they might collect previous key update messages. However, they cannot deduce the previous SKs. This is so because, although the adversaries get the current SKs and key update messages, it is computationally infeasible to compute the previous SKs. When the adversaries leave the group, the server updates the SKs and the key graph. Notice that the new SK is deduced as $k' = f(k \oplus k'_1)$, with $k'_1 = f(k_1 \oplus k_2)$, where k'_1 is the new child key of SK and k_2 is one of children keys which is not on the leave path. The adversaries cannot compute the new SKs because they do not hold k_2.

In conclusion, our proposed scheme ensures that users who do not belong to a particular group cannot decrypt data messages and thus both forward secrecy and backward secrecy are maintained.

5.2. Performance analysis

Because the IDHKGS scheme and the MGKMS scheme employ the same key graph, the storage overhead of the server in the IDHKGS scheme is equal to that in the MGKMS scheme, and the storage overhead of each user in the MGKMS scheme is 1 less than that in the IDHKGS scheme since a user in the IDHKGS scheme needs to hold the maximum ID of k-nodes in SG to which the user belongs, whereas a user in the MGKMS scheme does not need it. Therefore, we only analyze the rekeying overhead of single join/leave/switch.

In this section, we mainly compare the time and communication requirements of our new scheme and the MGKMS scheme. Table 4 summarizes our comparisons, focusing on the following measures: the multicast size, the unicast size, the KDC computation, and all members computation.

Our analysis assumes that the trees used by IDHKGS and MGKMS are binary. We define that C_E, C_r and C_f

Table 4
Performance comparison for single join/leave/switch

		MGKMS	IDHKGS
Single join	Multicast size	0	0
	Unicast size	$(N_J + 1)K$	$N_J K$
	KDC comp.	$N_J C_f + 2(C_r + C_E)$	$N_J C_f + C_r + C_E$
	Mem. comp.	$2N_J C_f + 2C_E$	$2N_J C_f + C_E$
Single leave	Multicast size	$2KN_L$	KN_L
	Unicast size	0	0
	KDC comp.	$N_L(2C_E + C_r)$	$N_L(C_E + C_f + C_r)$
	Mem. comp.	$2N_L C_E$	$N_L(C_E + C_f)$
Single switch	Multicast	$2KN_{SL}$	KN_{SL}
	Unicast	$(N_{SJ}+1)K$	$N_{SJ}K$
	KDC comp.	$N_{SJ}C_f + C_r(2+N_{SL})+2C_E(1+N_{SL})$	$2N_{SJ}C_f + C_r(1+N_{SL}) + C_E(1+N_{SL})$
	Mem. comp.	$2N_{SJ}C_f 82C_E(1+N_{SL})$	$3N_{SJ}C_f + C_E(1 + N_{SL})$

are the computational cost of an encryption/decryption algorithm, the computational cost of generating one key from a cryptographically secure random source, and the computational cost of one evaluation of the one-way function f, respectively. In addition, K is the size of one key in bits.

When a user wants to join the group S_i or a user in S_i wants to leave the group, the numbers of old k-nodes that are needed to be updated are N_J and N_L.

$$N_J = h(S_i) + \sum_{m=1}^{M} t_m^i h(D_m),$$

$$N_L = h(S_i) + \sum_{m=1}^{M} t_m^i h(D_m),$$

where M is the total number of the DGs, h(S_i)and h(D_m), are the heights of S_i-subtree and D_m-subtree, respectively. Notice that if the position that the new user joins is the same as the position of the leaving user in S_i-subtree, $N_J = N_L$, otherwise, $N_J \neq N_L$.

When a user in S_i wants to switch to S_j, $(i \neq j)$, the number of old k-nodes on the joining path and leaving path are N_{SJ} and N_{SL}.

$$N_{SJ} = h(S_i) + \sum_{m=1}^{M} \max((t_m^i - t_m^j), 0) \cdot h(D_m),$$

$$N_{SL} = h(S_j) + \sum_{m=1}^{M} \max((t_m^j - t_m^i), 0) \cdot h(D_m).$$

After a joining operation, in the two schemes, the KDC doesn't need to multicast any rekeying message because the old users in the group can deduce the new keys by themselves and the KDC does not need to multicast rekeying overhead. However, the unicast size, the KDC computation and all members computation of the IDHKGS scheme are slightly smaller than that of the MGKMS scheme. The reason is that the rekeying message for the newly created key is not needed in the IDHKGS scheme.

While both MGKMS and IDHKGS require a number of bits to be multicast to rekey after leaving or switching operation, IDHKGS roughly halves the multicast length over MGKMS because a child node of the updated k-node that is not on the leave path is chosen for the new key derivation in the IDHKGS scheme. Therefore, in the IDHKGS scheme, the KDC only needs to multicast new keys to the users that can't deduce by themselves, and achieves the goal of reducing the multicast size.

From Table 4, another advantage of IDHKGS over MGKMS is that the KDC requires fewer random keys in the IDHKGS. In particular, to evict one member, in the IDHKGS scheme the KDC must generate N_L new random key. By contrast, in the MGKMS scheme the KDC must generate $2KN_L$ new keys. If generating random keys in some practical application is relatively slow, the IDHKGS scheme could save more time than the MGKMS scheme.

In addition, for both the KDC computation and all members computation, the C_E in the IDHKGS scheme is smaller than that in the MGKMS scheme but the IDHKGS scheme requires more C_f because the users in the IDHKGS scheme can deduce more new keys by themselves and the number of rekeying messages that the KDC needs to send is reduced. As the computational cost of the encryption function is larger than that of the one-way function, therefore, the IDHKGS scheme can reduce the computations of KDC and members compared with the MGKMS scheme.

5.3. Simulation studies

We compare the IDHKGS scheme with the MGKMS scheme in simulations using C++. The simulation scenario is within a group of three data streams, in which N users are randomly distributed on the three SGs. The users in SG S_i have access to the DG D_1, D_2,..., D_i. To simplify the model, batch rekeying is applied and the key trees are binary. During fixed time interval t, users join different SGs with the same probability α, the users in each SG leave the group with the same probability β, and the users in the group switch to the neighboring SGs with the same probability γ, that is, the users in the S_1 and S_3 can only switch to S_2, the users in the S_2 can switch to S_1 and S_3. After t, the

KDC updates the keys and sends the rekeying messages together. The parameters in the model are shown in Table 5.

Fig. 15 shows the rekeying overhead at the KDC with the increase of N using the IDHKGS scheme and the MGKMS scheme respectively. When the total number of users in the group is increased from 100 to 3000, the rekeying overhead at the KDC of both schemes tends to increase because more users join/leave the group or switch to different SGs. It is clear that when N is very small, the rekeying overhead of two schemes is similar. However, the IDHKGS scheme reduces the rekeying overhead by more than 50% with the increase of N.

Table 5
The simulation parameters

Rekeying interval (t)	1000 (ms)
Probability of joining (α)	0.001
Probability of leaving (β)	0.01
Probability of switching (γ)	0.001

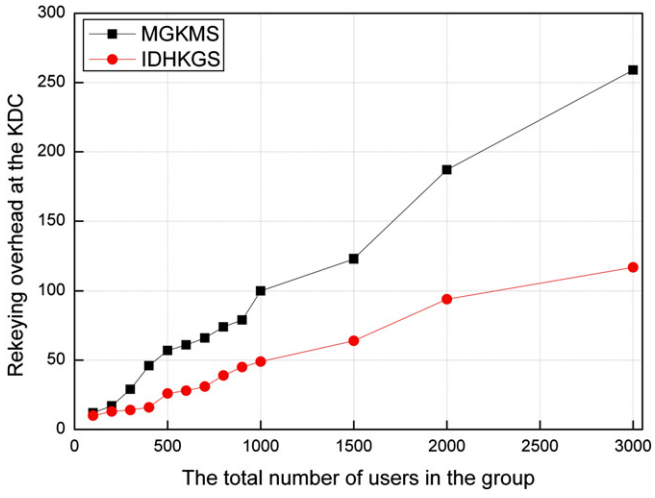


Fig. 15. Rekeying overhead at the KDC (M = 3).

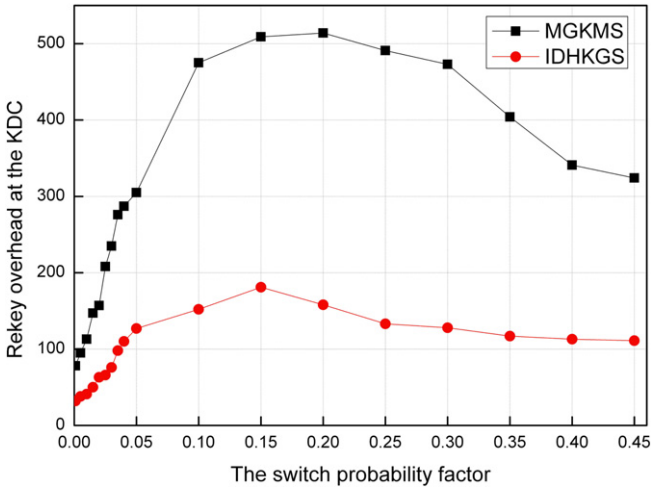


Fig. 16. Rekeying overhead at the KDC with different switch probability ($N = 15000$).

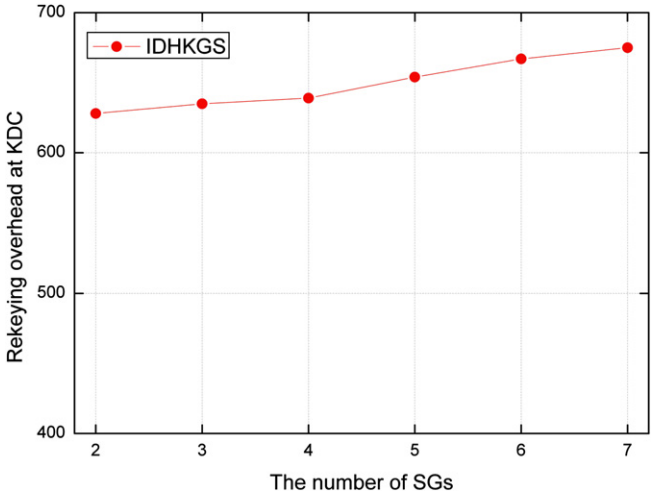


Fig. 17. Rekeying overhead at the KDC with different numbers of SGs.

Fig. 16 shows the comparison of rekeying overhead at the KDC with the change of switch probability when the total number of users in the group is 1000. It can be seen that the rekeying overhead at the KDC of both schemes tends to increase first and then decrease. That is because users are much more likely to switch SGs than to stay in the current SG when the switch probability increases. Many users leave an SG with batch rekeying requires less rekeying messages than several users' leaving. Whenγ is smaller than 2β, the rekeying overhead of the IDHKGS scheme increase slower than the MGKMS scheme, the rekeying overhead will be reduced by 70% in the best situation. Whenγ is larger than 2β, the rekeying overhead of the IDHKGS scheme does not reduce as fast as the MGKMS, but the former also reduces the rekeying overhead by more than 50%.

Fig. 17 shows the rekeying overhead at the KDC with the number of SGs increase. In the simulation, $N = 15000$, $t = 100$ s, =0.02/M. The rekeying overhead of MGKMS scheme does not change much in this situation. It is clear that when M increases, the rekeying overhead of the IDHKGS scheme also increase slowly. Therefore, in the IDHKGS scheme, the scale of the key graph mainly depends on the whole group size, not the number of SGs or data streams.

6. Conclusions

In this paper, we have investigated the issues of group key management in support of multi-privileged group communications and proposed an ID-based hierarchical key graph scheme for secure multi-privileged group communications in the mobile Internet, where the mobile ad-hoc networks (MANETs) are considered as an essential component in such integrated network environment. According to the relationship between children nodes and parent node as well as the relationship between data streams and SGs, each node on key graph is assigned a unique ID, in order for the node to deduce the IDs of his parent node and ancestor nodes. The server only needs to broadcast the IDs of joining user and

leaving user, the old users in the group know which nodes they hold should be updated. The new key value is computed by a one-way function, the server does not need to send rekeying messages when a user joins. When a user leaves, part of users also can compute the new keys by themselves. No matter when joining/leaving or switching, the users in the group can deduce some updated keys. Theoretical analysis and simulation studies show that our proposed scheme needs only half of the rekeying overhead of the MGKMS scheme, and the proposed scheme outperforms the MKMG scheme in all cases.

Currently the proposed scheme uses only binary tree and can not flexibly deal with formation and decomposition of service groups. We plan to extend its usage of k-ary trees and to propose solutions to the dynamic formation and decomposition of service groups.

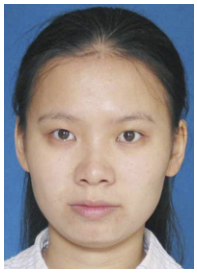
Acknowledgements

This work is supported in part by the National High-Tech Research and Development Plan of China (863 Plan) under Grant No. 2006AA01Z202, the National Natural Science Foundation of China under Grant No. 60533040, the Research Grants NSC 95-2221-E-110-062 and NSC 95-2221-E-110-063 from National Science Council, Taiwan, and the Program for New Century Excellent Talents in University under Grant No. NCET-06-0686.

References

[1] W. Trappe, J. Song, R. Poovendran, K.J.R. Liu, Key distribution for secure multimedia multicasts via data embedding, Proceedings of IEEE ICASSP'01. 3 (2001) 1449–1452.
[2] S. Rafaeli, D. Hutchison, A survey of key management for secure group communication, ACM Computing Surveys 35 (3) (2003) 309–329.
[3] D. McGrew, A. Sherman, Key establishment in large dynamic groups using one-way function trees, Technical Report 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, (1998).
[4] G.H. Chiou, W.T. Chen, Secure broadcasting using the secure lock, IEEE Transactions on Software Engineering 15 (8) (1989) 929–934.
[5] S. Banerjee, B. Bhattacharjee, Scalable secure group communication over IP multicast, IEEE Journal on Selected Areas in Communications Special Issue on Network Support for Group Communication. 20 (8) (2002) 1511–1527.
[6] D.M. Wallner, E.J. Harder, R.C. Agee, Key management for multicast: issues and architectures, Internet Draft Report, Filename: draft-wallner-key-arch -01.txt. 1998.
[7] S. Mittra, Iolus: A framework for scalable secure multicasting, Computer Communication Review, 27 (4) (1997) 277–288, New York, ACM Press.
[8] A. Perrig, D. Song, D. Tygar, ELK, a new protocol for efficient large-group key distribution, Proceedings of IEEE Symposium on Security and Privacy (2001) 247–262.
[9] Y. Sun, K.J.R. Liu, Scalable hierarchical access control in secure group communications, Proceedings of IEEE INFOCOM 2004 2 (2004) 1296–1306.
[10] Q. Zhang, Y. Wang, A centralized key management scheme for hierarchical access control, Proceedings of Global Telecommunications Conference 4 (2004) 2067–2071.
[11] A.M. Eskicioglu, S. Dexter, E.J. Delp, Protection of multicast scalable video by secret sharing: simulation results, Proceedings of SPIE Security and Watermarking of Multimedia Contents, Santa Clara, CA. (2003) 505–515.
[12] J.C. Lin, P.F. Lai, H.C. Lee, Efficient group key management protocol with one-way key derivation, Proceedings of IEEE conference on Local Computer Networks 30th Anniversary (2005) 336–343.
[13] C. Yuan, B. Zhu, M. Su, X. Wang, S. Li, Y. Zhong, Layered access control for MPEG-4 FGS video, Proceedings of IEEE International Conference on Image Processing 2003 1 (2003), I- 517-20.
[14] C.K. Wong, M. Gouda, S.S. Lam, Secure group communications using key graphs, Proceedings of ACM SIGCOMM98. (1998) 68–79.
[15] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, B. Plattner, The versakey framework:versatile group key management, IEEE Journal on Selected Areas in Communications 17 (9) (1999) 1614–1631.
[16] J.C. Birget, X. Zou, G. Noubir, B. Ramamurthy, Hierarchical-based access control in distributed environments, Proceedings of International Conference on Communications 1 (2001) 229–233.
[17] R. Deng, Y. Wu, D. Ma, Securing JPEG2000 Code-Streams, Proceedings of International Workshop on Advanced Developments in Software and Systems Security (2003).
[18] D. Ma, Y. Wu, R. Deng, T. Li, Dynamic access control for multi-privileged group communications, Proceedings of 6th International Conference on Information and Communications Security, Lecture Notes in Computer Science (LNCS), 3269 (2004) 508–519.
[19] R. Li, J. Li, H. Kameda, Distributed hierarchical access control for secure group communications, Proceedings of ICCNMC, LNCS 3619 (2005) 539–548.
[20] X.B. Zhang, S.S. Lam, D.Y. Lee, Y.R. Yang, Protocol design for scalable and reliable group rekeying, IEEE/ACM Transactions on Networking. 11 (6) (2003) 908–922.
[21] Y.R. Yang, X.S. Li, X.B. Zhang, S.S. Lam, Reliable group rekeying: a performance analysis, Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications (2001) 27–28.

Guojun Wang received his BSc degree in Geophysics, MSc degree in Computer Science, and PhD degree in Computer Science, from the Central South University, in 1992, 1996, 2002, respectively. He is currently a Professor (September 2005) and a PhD supervisor in the Department of Computer Science, the Central South University, Changsha, PR China, 410083. He is also the vice head of the Department and the director of the Mobile Computing Lab in the Department. He was a Research Fellow (March 2003–March 2005) in the Department of Computing, the Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. And he was a Visiting Researcher (September 2006–February 2007) in the School of Computer Science and Engineering, University of Aizu, Japan. His research interests include computer networks, fault tolerant and dependable computing, and software engineering. He has published more than 120 technical papers in the above areas. He has served as a reviewer for many international journals such as IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Wireless Communications, IEEE Transactions on Vehicular Technology, IEEE Wireless Communications Magazine, IEEE Network, IEICE Transactions on Information and Systems, Elsevier Computer Communications, Elsevier Information Sciences, Elsevier Pervasive and Mobile Computing Journal, Wireless Communications and Mobile Computing (John Wiley & Sons). He has also served as a program committee member for many international conferences such as GLOBECOM, ICC, WCNC, IWCMC, AINA, EUC, and ISPA. He is a senior member of the China Computer Federation (2005), the chair of the YOCSEF Changsha of the China Computer Federation (2007), a vice chair of the YOCSEF Changsha of the China Computer Federation (2006), a YOCSEF member of the China Computer Federation (2001), and a member of the Hunan Provincial Association of Computers (1994).



Jie Ouyang received her BSc and MSc degrees in Computer Science from the Hunan Normal University and the Central South University in 2004 and 2007, respectively. Her research interests include designing secure protocols in mobile Internet, mobile ad-hoc networks, and wireless sensor networks.

Hsiao-Hwa Chen was the founding Director of Institute of Communications Engineering, National Sun Yat-Sen University, Taiwan. He has been a visiting Professor to Department of Electrical Engineering, University of Kaiserslautern, Germany, in 1999, the Institute of Applied Physics, Tsukuba University, Japan, in 2000, Institute of Experimental Mathematics, University of Essen, Germany in 2002 (under DFG Fellowship), Department of Information Engineering, The Chinese University of Hong Kong, 2003, and Department of Electronics Engineering, The City University of Hong Kong, 2006. His current research interests include wireless networking, MIMO systems, and next generation CDMA technologies for future wireless communications. He has authored or co-authored over 200 technical papers in major international journals and conferences, and five books in the areas of communications, including "Next Generation Wireless Systems and Networks" (512 pages) and "The Next Generation CDMA Technologies" (463 pages), both published by John Wiley & Sons. He served or is serving as International Steering Committee member, Symposium Chair and General Co-Chair of more than 50 international conferences, including IEEE VTC, IEEE ICC, IEEE Globecom, and, IEEE WCNC, etc. He served or is serving as the Editor, Associate Editor, Editorial Board member or Guest Editor of many international journals, including IEEE Communications Magazine, IEEE JSAC, IEEE Vehicular Technology Magazine, IEEE Wireless Communications Magazine, IEEE Network Magazine, IEEE Transactions on Wireless Communications, IEEE Communications Letters, Wiley's Wireless Communications and Mobile Computing (WCMC) Journal, Wiley's International Journal of Communication Systems, etc. He is an adjunct Professor of Zhejiang University, China, and Shanghai Jiao Tung University, China. Currently, Professor Chen is serving as the Chair of Radio Communications Committee, IEEE Communications Society.

Minyi Guo received his Ph.D. degree in computer science from University of Tsukuba, Japan. Before 2000, Dr. Guo had been a research scientist of NEC Corp., Japan. He is now a chair professor at the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. He was also a visiting professor of Georgia State University, USA, Hong Kong Polytechnic University, University of Hong Kong, National Sun Yet-Sen University in Taiwan, China, University of Waterloo, Canada and University of New South Wales, Australia. He is also a professor of The University of Aizu, Japan. Dr. Guo has published more than 120 research papers in international journals and conferences. Dr. Guo has served as general chair, program committee or organizing committee chair for many international conferences. He is the founder of International Conference on Parallel and Distributed Processing and Applications (ISPA), and International Conference on Embedded and Ubiquitous Computing (EUC). He is the editor-in-chief of the Journal of Embedded Systems. He is also in editorial board of Journal of Pervasive Computing and Communications, International Journal of High Performance Computing and Networking, Journal of Embedded Computing, Journal of Parallel and Distributed Scientific and Engineering Computing, and International Journal of Computer and Applications. Dr. Guo's research interests include parallel and distributed processing, parallelizing compilers, Pervasive Computing, Embedded Software Optimization, molecular computing and software engineering. He is a senior member of IEEE, a member of the ACM, IPSJ and IEICE.