Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 50 (2006) 1781–1798
Computer Networks
www.elsevier.com/locate/comnet

Key bundles and parcels: Secure communication in many groups

Eunjin Jung *, Alex X. Liu, Mohamed G. Gouda

Department of Computer Sciences, The University of Texas at Austin, 1 University Station C0500, Austin, TX 78712, United States

Received 30 March 2005; received in revised form 22 July 2005; accepted 28 July 2005
Available online 31 August 2005

Responsible Editor: D. Frincke

Abstract

We consider a system where each user is in one or more elementary groups. In this system, arbitrary groups of users can be specified using the operations of union, intersection, and complement over the elementary groups in the system. Each elementary group in the system is provided with a security key that is known only to the users in the elementary group and to the system server. Thus, for any user u to securely multicast a data item d to every user in an arbitrary group G, u first forwards d to the system server which encrypts it using the keys of the elementary groups that comprise G before multicasting the encrypted d to every user in G. Every elementary group is also provided with a key tree to ensure that the cost of changing the key of the elementary group, when a user leaves the group, is small. In [E. Jung, A.X. Liu, M.G. Gouda, Key bundles and parcels: secure communication in many groups, in: LNCS 2816, Group Communications and Charges, 2003], we introduced two methods for packing the key trees of elementary groups into key bundles and into key parcels. We also showed that packing into key bundles has the advantage of reducing the number of encryptions needed to multicast a data item to the complement of an elementary group, while packing into key parcels has the advantage of reducing the total number of keys in the system. In this paper, we present more details of key bundles and parcels: the algorithms that construct key bundles and parcels, and more simulation results comparing key bundles and parcels. We also discuss how to reconfigure key bundles and parcels when the user joins or leaves different elementary groups and how to balance the load among multiple servers.

Keywords: Security; Network security; Group communication; Multicast; Key tree

* Corresponding author. Tel.: +1 512 471 9532; fax: +1 512 471 8885.
E-mail addresses: ejung@cs.utexas.edu (E. Jung), alex@cs.utexas.edu (A.X. Liu), gouda@cs.utexas.edu (M.G. Gouda).

1389-1286/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2005.07.013

1. Introduction

We consider a system that consists of n users denoted u_i, $0 \leqslant i < n$. The system users share one security key, called the system key. Each user u_i can use the system key to encrypt any data item before sending it to any subset of the system users, and can use it to decrypt any data item after receiving it from any other system user. (Examples of such systems are secure multicast systems [1–4], secure peer-to-peer systems [5], and secure wireless networks [6].)

When a user u_i leaves the system, the system key needs to be changed so that u_i can no longer decrypt the encrypted data item exchanged within the system. This requires to add a server S to the system and to provide each system user u_j with an individual key K_j that only user u_j and server S know. When a user u_i leaves the system, server S changes the system key and sends the new key to each user u_j, other than u_i, encrypted using its individual key K_j. The cost of this rekeying scheme, measured by the number of needed encryptions, is $O(n)$, where n is the number of users in the system.

Clearly, this solution does not scale when the number of users become large. More efficient rekeying schemes have been proposed in [7–14]. A particular efficient rekeying scheme [3,4,12] is shown to cost merely $O(\log n)$ encryptions. This scheme is extended in [15–17], and is shown to be optimal in [18].

This scheme is based on a distributed data structure called a key tree. A *key tree* is a directed, incoming, rooted, balanced tree where each node represents a key. The root of the tree represents the system key and each leaf node represents the individual key of a system user. The number of leaf nodes is n, which is the number of users in the system. Each user knows all the keys on the directed path from its individual key to the root of the tree, and the server knows all the keys in the key tree. Thus, in a binary key tree, each user knows $\lceil \log_2 n \rceil + 1$ keys, and the server knows $(2n - 1)$ keys.

An example of a key tree for a system of eight users is depicted in Fig. 1(a). The root of the key tree represents the system key $K_{01234567}$ that is known to all users in the system. Each user also knows all the keys on the directed path from its individual key to the root of the key tree. For example, user u_7 knows all the keys K_7, K_{67}, K_{4567}, and $K_{01234567}$.

Fig. 1(a) and (b) illustrates the protocol for updating the system key when user u_7 leaves the system. In this case, the system server S is required to change the keys $K_{01234567}$, K_{4567}, and K_{67} that user u_7 knows. To update these keys, S selects new keys $K_{0123456(7)}$, $K_{456(7)}$, and $K_{6(7)}$, encrypts them, and sends them to the users that need to know them. To ensure that u_7 cannot get a copy of the new keys, S needs to encrypt the new keys using keys that u_7 does not know. Therefore, S encrypts the new $K_{0123456(7)}$ with the old K_{0123},

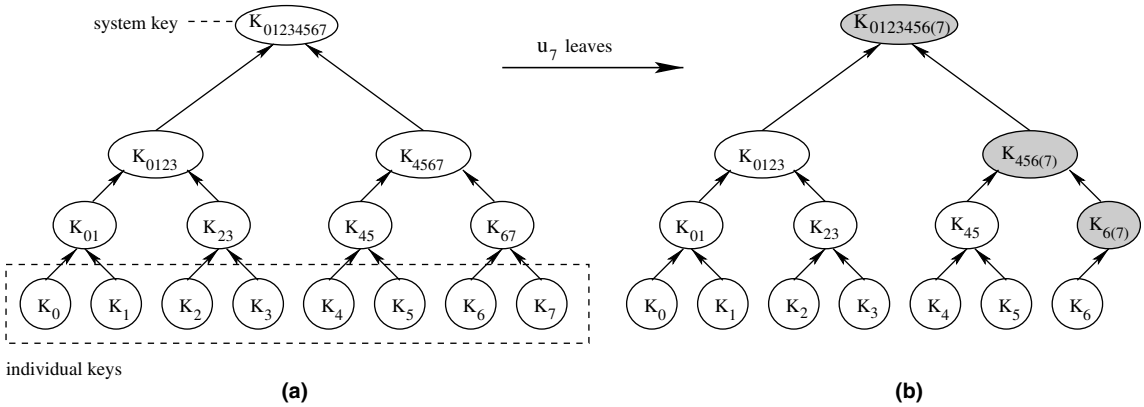

Fig. 1. A binary key tree before and after u_7 leaves.

encrypts the new $K_{0123456(7)}$ and the new $K_{456(7)}$ with the old K_{45}, encrypts the new $K_{0123456(7)}$, the new $K_{456(7)}$, and the new $K_{6(7)}$ with K_6. Then, S multicasts the encrypted keys to the corresponding holders of these keys. The protocol can be specified as follows:

$$S \rightarrow u_0,\ldots,u_6: \ \{u_0,u_1,u_2,u_3\},\ K_{0123}\langle K_{0123456(7)}|\mathrm{chk}\rangle,$$
$$S \rightarrow u_0,\ldots,u_6: \ \{u_4,u_5\},\ K_{45}\langle K_{0123456(7)}|K_{456(7)}|\mathrm{chk}\rangle,$$
$$S \rightarrow u_0,\ldots,u_6: \ \{u_6\},\ K_6\langle K_{0123456(7)}|K_{456(7)}|K_{6(7)}|\mathrm{chk}\rangle.$$

This protocol consists of three steps. In each step, server S broadcasts a message consisting of two fields to every user in the system. The first field defines the set of the intended ultimate destinations of the message. The second field is an encryption, using an old key, of the concatenation of the new key(s) and the checksum chk computed over the new key(s). Note that although the broadcast message is sent to every user in the system, only users in the specified destination set have the key used in encrypting the message and so only they can decrypt the message.

The above system architecture is based on the assumption that the system users constitute a single group. In our previous work and this paper, we extend this architecture to the case where the system users form many groups. In [19], we introduced two methods for packing the key trees of elementary groups into key bundles and into key parcels. We also showed that packing into key bundles has the advantage of reducing the number of encryptions needed to multicast a data item to the complement of an elementary group, while packing into key parcels has the advantage of reducing the total number of keys in the system. In this paper, we present more details of key bundles and parcels: the algorithms that construct key bundles and parcels, and more simulation results comparing key bundles and parcels. We also discuss how to reconfigure key bundles and parcels when the user joins or leaves different elementary groups and how to balance the load among multiple servers.

2. Groups and group algebra

Assume that the system has m, $m \geqslant 1$, elementary groups: each elementary group is a distinct subset of the system users and one elementary group has all the system users. Every elementary group has a unique identifier G_j, $0 \leqslant j \leqslant m-1$. The identifier for the elementary group that has all users is G_0. As an example, Fig. 2 illustrates a system that has eight users u_0 through u_7 and five elementary groups G_0, G_1, G_2, G_3, and G_4.

The system needs to be designed such that any user u_i can securely multicast data items to all users in any elementary group G_j. Moreover, any user u_i can securely multicast data items to all users in any group, where a group is defined recursively according to the following four rules:

1. Any of the elementary groups $G_0,\ldots,G_{m-1}$ is a group.
2. The union of any two groups is a group.
3. The intersection of any two groups is a group.
4. The complement of any group is a group. (Note that the complement of any group G is the set of all users in G_0 that are not in G.)

Thus, the set of groups is closed under the three operations of union, intersection, and complement.

Each group can be defined by a group formula that includes the following symbols.

- G_0 through G_{m-1},
- $\vee$ for union,
- $\wedge$ for intersection,
- $\neg$ for complement.

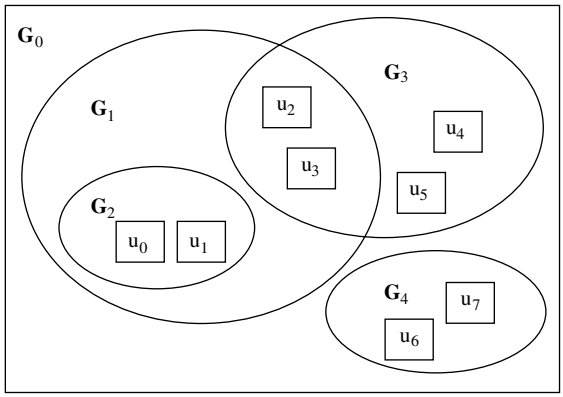

Fig. 2. A sample system.

Group formulae can be manipulated using the well-known laws of algebra: associativity, commutativity, distribution, De Morgan's, and so on. For example, the group formula

$$G_1 \vee \neg(\neg G_2 \wedge G_1)$$

can be manipulated as follows:

$$\begin{aligned} & G_1 \vee \neg(\neg G_2 \wedge G_1) \\ & = \{\text{by De Morgan's}\}\ G_1 \vee (\neg\neg G_2 \vee \neg G_1) \\ & = \{\text{by associativity of } \vee\}\ G_1 \vee \neg\neg G_2 \vee \neg G_1 \\ & = \{\text{by definition of complement}\}\ G_1 \vee G_2 \vee \neg G_1 \\ & = \{\text{by commutativity of } \vee\}\ G_1 \vee \neg G_1 \vee G_2 \\ & = \{\text{by definition of complement}\}\ G_0 \vee G_2 \\ & = \{\text{by definition of } \vee\}\ G_0. \end{aligned}$$

From this formula manipulation, it follows that the group defined by the formula $G_1 \vee \neg(\neg G_2 \wedge G_1)$ is the set of all system users. Thus, for a user u_i to securely multicast a data item d to every user in the group $G_1 \vee \neg(\neg G_2 \wedge G_1)$, it is sufficient for u_i to securely broadcast d to every user in the system.

In the rest of this paper, we consider solutions for the following problem. How to design the system so that any system user u_i can securely multicast data items to any group G in the system. Any reasonable solution for this problem needs to take into account that the users can leave any elementary group in the system or leave the system altogether, and these activities may require to change the security keys associated with the elementary groups from which users leave. In particular, the solution should utilize key trees, discussed in Section 1, that can reduce the cost of changing the security keys from $O(n)$ to $O(\log n)$, where n is the total number of users in the system. This problem is also discussed in [20].

The above problem has many applications. As a first example, consider a cable TV broadcasting system that has four packages: Basic, Standard, Cinema, and Sports. When the cable TV provider wants only users who subscribe both Standard and Sports packages to be able to watch Super Bowl game, then the provider securely multicasts the game to all users in the group, *Standard* $\wedge$ *Sports*.

As a second example, consider a student registration system in some university. This system has m elementary groups G_0 through G_{m-1}, where each G_i is a list of the students registered in one course section. A professor who is teaching three sections G_5, G_6, G_7 of the same course, may wish to securely multicast any information related to the course to all the students in the group $G_5 \vee G_6 \vee G_7$.

As a third example, we consider the following military scenario. There are three overlapping regiments, G_1, G_2 and G_3, in a battle field. The general commander of these regiments gets the information that some soldiers in regiment G_3 cannot be trusted. In this case, he might want to inform the soldiers in G_1 and G_2, but not in G_3, that the messages from G_3 should not be trusted. Therefore, he can multicast this message to all the soldiers in the regiment $(G1 \vee G2) \wedge \neg G3$.

3. Key bundles

The above problem suggests the following simple solution (which we show below that it is ineffective). First, assign to each elementary group G_j a security key to be shared by all the users of G_j. Second, assign to the complement $\neg G_j$ of each elementary group G_j a security key to be shared by every member of this complement. Third, provide a key tree for each elementary group and another key tree for its complement. Note that the two key trees provided for an elementary group and its complement span all the users in the system. Thus, these two trees can be combined into one *complete* key tree that spans all system users in the system. Fig. 3 shows the four complete key trees that are provided for the four elementary groups and their complements in the system in Fig. 2.

From Fig. 3(a), the key for the elementary group G_1 is K_{0123} and the key for its complement is K_{4567}. From Fig. 3(b), the key for the elementary group G_2 is K_{01} and the key for its complement is K_{234567}. From Fig. 3(c), the key for the elementary group G_3 is K_{2345}, and the key fro its complement is K_{0167}. From Fig. 3(d), the key for the elementary group G_4 is K_{67}, and the key for its complement is K_{012345}.

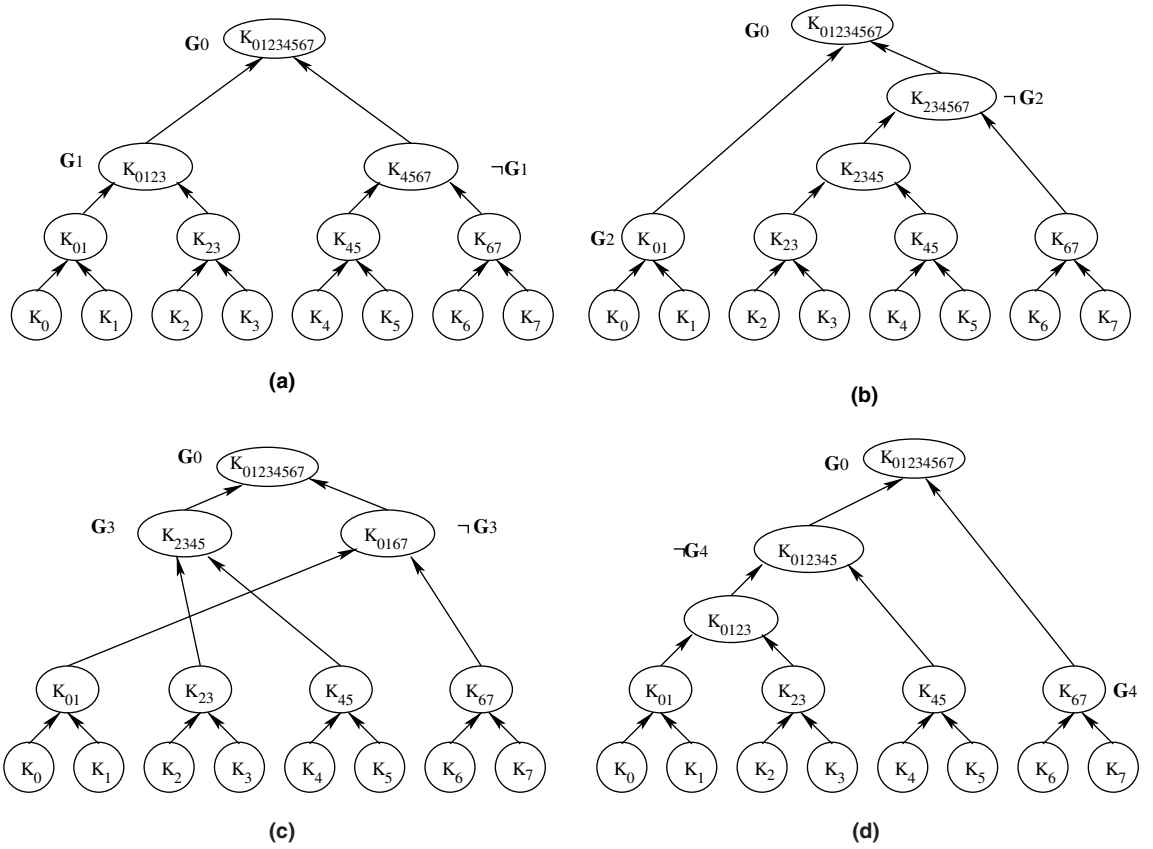

Fig. 3. The complete key trees for the elementary groups and their complements: (a) Groups $G_0, G_1, \neg G_1$; (b) Groups $G_0, G_2, \neg G_2$; (c) Groups $G_0, G_3, \neg G_3$; (d) Groups $G_0, G_4, \neg G_4$.

Note that these complete trees have the same key for group G_0, and the same individual key for each user. Nevertheless, the total number of distinct keys in these complete trees is 19, which is relatively large for this rather simple system. In general, this method requires O(mn) keys, where m is the number of elementary groups and n is the number of users in the system.

To reduce the total number of needed keys, several elementary groups can be added to the same complete key tree, provided that these elementary groups are "non-conflicting". This idea suggests the following three definitions of non-conflicting elementary groups, bundles, and bundle covers.

Two elementary groups are *non-conflicting* if and only if either their intersection is empty or one of them is a subset of the other. In the system example in Fig. 2, the three elementary groups G_0, G_1 and G_2 are non-conflicting since G_1 is a subset of G_0, and G_2 is a subset of G_1. On the other hand, the two elementary groups G_1 and G_3 are conflicting, because they share two users u_2 and u_3 and neither group is a subset of the other.

A *bundle* of a system is a maximal set of non-conflicting elementary groups of the system. In the system example in Fig. 2, the four elementary groups G_0, G_1, G_2, G_4 constitute one bundle B_0, and the four elementary groups G_0, G_2, G_3, G_4 constitute a second bundle B_1.

A *bundle cover* of a system is a set $\{B_0, \ldots, B_{m-1}\}$ of system bundles such that the following two conditions hold:

1. *Completeness*: Each elementary group of the system appears in some bundle B_i in the bundle cover.

2. *Compactness*: Each bundle B_i has at least one elementary group that does not appear in any other bundle B_j in the bundle cover.

Note that the set $\{B_0, B_1\}$, where $B_0 = \{G_0, G_1, G_2, G_4\}$ and $B_1 = \{G_0, G_2, G_3, G_4\}$, is a bundle cover for the system in Fig. 2.

The security keys for the elementary groups in a bundle can be arranged in a complete key tree. For example, Fig. 4(a) shows the complete key tree for B_0. In this tree, the key for group G_0 is $K_{01234567}$, the key for group G_1 is K_{0123}, the key for G_2 is K_{01}, and the key for G_4 is K_{67}. Note that users u_4 and u_5 in G_0 do not belong to any other elementary group in the bundle, and so they are viewed as forming a complement group C_0 whose key is K_{45}. We refer to a complete key tree that corresponds to a bundle as a *key bundle*.

Fig. 4(b) shows the complete key bundle for B_1. Note that in this bundle every user in G_0 is also in another elementary group. Thus, the resulting complete key tree does not have a complement group as in the former key tree in Fig. 4(a).

Comparing the two key bundles in Figs. 4(a) and (b), one observes that each of the elementary groups G_0, G_2, and G_4 appear in both key bundles because none of them conflict with any elementary group or any group in the system. One also observes that each of these groups has the same group key in both key bundles, and that the individual key of each user is the same in both key bundles. Note that these key bundles have only 15 distinct keys compared with the 19 distinct keys in the four complete trees in Fig. 3. This represents more than 20% reduction in the total number of keys in the server.

The system server S knows the two key bundles in Fig. 4, and each user u_i knows only the keys that exist on the paths from its individual key K_i to the key of group G_0. Thus, each user u_i needs to collaborate with the system server S in order to securely multicast data items to any elementary group or any group that can be defined by intersection, union, and complement of elementary groups. This point is illustrated by the following four examples.

For the first example, assume that user u_0 wants to securely multicast a data item d to every user in group G_4. In this case, user u_0 can execute the following protocol:

$$u_0 \to S: \quad K_0\langle d|G_4|\mathrm{chk}\rangle,$$
$$S \to u_0, \ldots, u_7: \quad G_4, \quad K_{67}\langle d|u_0|\mathrm{chk}\rangle.$$

This protocol consists of two steps. In the first step, user u_0 sends a message $K_0\langle d|G_4|\mathrm{chk}\rangle$ to server S. This message consists of three concatenated fields, namely the data item d, its intended destination G_4, and the checksum chk; the message is encrypted by the individual key K_0 of user u_0. In the second step, server S multicasts the message G_4, $K_{67}\langle d|u_0|\mathrm{chk}\rangle$ where the second field consists of the data item d, the message source u_0, and the checksum chk and is encrypted with the group key of G_4.

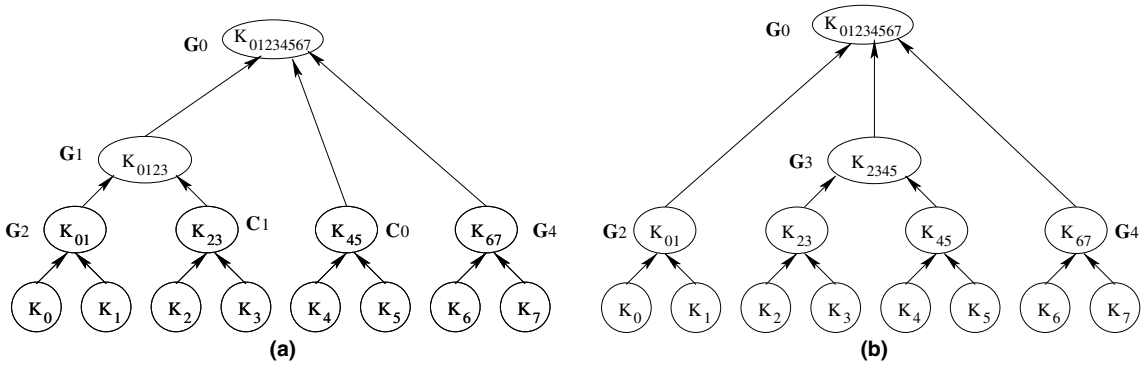

Fig. 4. The complete key trees for the two bundles B_0 and B_1: (a) $B_0 = \{G_0, G_1, G_2, G_4\}$ and (b) $B_1 = \{G_0, G_2, G_3, G_4\}$.

For the second example, assume user u_1 wants to securely multicast a data item d to the users in either group G_1 or G_3, namely the users in the union of G_1 and G_3. In this case, user u_1 can execute the following protocol:

$$u_1 \to S: \quad K_1\langle d|G_1 \vee G_3|\mathrm{chk}\rangle.$$
$$S \to u_0,\ldots,u_7: \quad G_1 \vee G_3, \quad K_{0123}\langle d|u_1|\mathrm{chk}\rangle, \quad K_{2345}\langle d|u_1|\mathrm{chk}\rangle.$$

In the second step of this protocol, server S multicasts the message $G_1 \vee G_3$, $K_{0123}\langle d|u_1|\mathrm{chk}\rangle$, $K_{2345}\langle d|u_1|\mathrm{chk}\rangle$ to the two groups G_1 and G_3. The users in group G_1 can get d by using the group key K_{0123} to decrypt $K_{0123}\langle d|u_1|\mathrm{chk}\rangle$ and the users in group G_3 can get d by using the group key K_{2345} to decrypt $K_{2345}\langle d|u_1|\mathrm{chk}\rangle$. Note that if it is u_2 who wants to send d to $G_1 \vee G_3$, then since u_2 belongs to both G_1 and G_3, u_2 already knows both K_{0123} and K_{2345}. Therefore, u_2 can send the encrypted d directly to the users in G_1 and G_3 as follows:

$$u_2 \to u_0,\ldots,u_7: \quad G_1 \vee G_3, \quad K_{0123}\langle d|u_2|\mathrm{chk}\rangle, \quad K_{2345}\langle d|u_2|\mathrm{chk}\rangle.$$

For the third example, assume that user u_4 wants to send a data item d to all the users in the intersection of G_1 and G_3. In this case, user u_4 can execute the following protocol.

$$u_4 \to S: \quad K_4\langle d|G_1 \wedge G_3|\mathrm{chk}\rangle.$$
$$S \to u_0,\ldots,u_7: \quad G_1 \wedge G_3, \quad K_{0123}\langle K_{2345}\langle d|u_4|\mathrm{chk}\rangle\rangle.$$

In the second step of this protocol, server S multicasts a message $G_1 \wedge G_3$, $K_{0123}\langle K_{2345}\langle d|u_4|\mathrm{chk}\rangle\rangle$ to the group $G_1 \wedge G_3$. Here the concatenation of d, u_4 and chk is encrypted by both the group key of G_1, which is K_{0123}, and the group key of G_3, which is K_{2345}. The encrypted message can only be decrypted by the users that are in both G_1 and G_3 because only these users know the two group keys K_{0123} and K_{2345}.

For the fourth example, assume that user u_5 wants to send a data item d to all the users in the complement of group G_1. In this case, user u_5 executes the following protocol:

$$u_5 \to S: \quad K_5\langle d|\neg G_1|\mathrm{chk}\rangle.$$
$$S \to u_0,\ldots,u_7: \quad C_0 \vee G_4, \quad K_{45}\langle d|u_5|\mathrm{chk}\rangle, K_{67}\langle d|u_5|\mathrm{chk}\rangle.$$

After server S receives this message, it translates $\neg G_1$ to $C_0 \vee G_4$ then multicasts the message $C_0 \vee G_4$, $K_{45}\langle d|u_5|\mathrm{chk}\rangle$, $K_{67}\langle d|u_5|\mathrm{chk}\rangle$. The users in group C_0 can get d using the group key K_{45}, and the users in group G_4 can get d using the group key K_{67}.

4. Construction of key bundles

In this section, we describe a procedure that can be used by the server of a system to construct and maintain key bundles for that system. This procedure consists of two algorithms. The first algorithm, presented in Section 4.1, constructs a bundle cover for the given system. The second algorithm, presented in Section 4.2, computes a complete key tree (i.e. a key bundle) for each bundle in the bundle cover constructed by the first algorithm.

4.1. Algorithm for bundle cover construction

This algorithm takes any given system with m elementary groups $G_0,\ldots,G_{m-1}$ and constructs a bundle cover $\{B_0,\ldots,B_{r-1}\}$ for the given system, where $r \leqslant m$.

In this algorithm, the given system is represented by a $m \times m$ boolean matrix C, where each element $C[i][j]$ is defined as follows:

$$C[i][j] = \begin{cases} \textbf{false} & \text{if } G_i \cap G_j = \emptyset \text{ or } G_i \subset G_j \text{ or } G_j \subset G_i, \\ \textbf{true} & \text{otherwise.} \end{cases}$$

The algorithm starts with m empty bundles, $B_0,\ldots,B_{m-1}$. Then the algorithm proceeds to add elementary groups of the given system to the bundles, one by one, until each elementary group is added to at least one bundle. Finally, the algorithm keeps the bundles $B_0,\ldots,B_{r-1}$ that have elementary groups and discards the remaining empty bundles $B_r,\ldots,B_{m-1}$.

The algorithm uses an array added $[0\ldots m-1]$ of m integers, where added[j] records the number of bundles that the elementary group G_j is added to. We call added[j] the counter of the elementary group G_j. Initially, the counter of every elementary group is zero. Whenever an elementary group G_j is

added to a bundle, its counter added[j] is incremented by one.

The bundle construction algorithm is specified in Algorithm 1. Note that Lines 7 and 12 in this algorithm call a boolean function NOCONFLICT(B_r, G_y). This function is specified in Algorithm 2.

Algorithm 1. Bundle cover construction algorithm

```
1:  r := 0;
2:  for x = 0 to m − 1
3:      if added[x] > 0 then goto line 2;
4:      B_r := B_r ∪ {G_x};
5:      added[x] := 1;
6:      for y = (x + 1) to m − 1
7:        if added[y] = 0 and NOCONFLICT
          (B_r, G_y)
8:        then B_r := B_r ∪ {G_y};
9:             added[y] := 1
10:     endfor;
11:     for y = 0 to (m − 1)
12:       if NOCONFLICT(B_r, G_y)
13:       then B_r := B_r ∪ {G_y};
14:            added[y] := added[y] + 1
15:     endfor;
16:     r := r + 1
17: endfor;
18: discard the empty bundles B_r, ..., B_{m−1};
19: discard each bundle in which the counter
    of each elementary group is greater than 1
```

Algorithm 2. Function NOCONFLICT

```
Function NOCONFLICT(B_r, G_y):boolean
  var flag: boolean
1: flag := true;
2: for every G_x in B_r
3:     if C[x][y] then
4:       flag := false;
5:       goto line 6;
6: endfor
7: return flag
```

As an example, if this algorithm is applied to the system in Fig. 2, the algorithm constructs the bundle cover $\{B_0, B_1\}$, where

$$B_0 = \{G_0, G_1, G_2, G_4\},$$
$$B_1 = \{G_0, G_2, G_3, G_4\}.$$

Consider a system with six elementary groups $G_0, \ldots, G_5$. Assume that G_0 conflicts with G_1, G_2.

As this algorithm computes a bundle cover with $O(m^3)$ time complexity, note that computing a bundle cover with minimum number of bundles is NP-Complete. The proof is in Theorem 1.

Theorem 1. *Given a system with elementary groups, computing a bundle cover of the system with the minimum number of bundles is NP-Complete.*

Proof. The proof consists of two parts. The first part shows that there is a polynomial algorithm that verifies whether a set of bundles is a bundle cover or not. The second part shows that a well-known NP-Complete problem, Vertex Coloring, can be reduced into computing a bundle cover with minimum number of bundles in polynomial time. The Vertex Coloring problem is defined as follows: given a graph $G = (V, E)$, each vertex is assigned with a color such that no adjacent vertices are in the same color. Finding an assignment with minimum number of colors is NP-Complete.

Part 1: Given a set of bundles, we can verify whether the set is a bundle cover or not in polynomial time. For each elementary group, find a bundle it belongs to. If there is any elementary group that does not belong to any bundle, the set of bundles is not a bundle cover. This step takes $O(mr)$, where m is the number of elementary groups, and r is the number of bundles. To be a bundle cover, each bundle has to have at least one elementary group that conflicts with some elementary group in every other bundle. Compute the conflict matrix $C[i][j]$ as described in Section 4, and compute the conflicts between bundles according to the matrix. We can verify all these conflicts in $O(m^2 + r^2)$ steps.

Part 2: Given a graph $G = (V, E)$, construct a system S_G as follows: for each node u in V, add an elementary group G_u to S_G. For each edge (u, v) in E, add a user $user_{uv}$ to both elementary groups G_u and G_v in S_G so that these elementary groups conflict with each other. Assume that we have the

bundle cover $B_0, B_1, \ldots, B_{r-1}$ of the system S_G with the minimum number of bundles.

First, we show that we can color the vertices with r colors. Let r colors to be $C_0, C_1, \ldots, C_{r-1}$. For each elementary group G_u in B_i, color the vertex u in G with color C_i. (For an elementary group G_u that is in many bundles, choose the smallest i of B_i can color with C_i. In fact, we can choose any i and the proof still holds.) The system S_G is constructed so that for any two adjacent nodes u and v in G, the corresponding elementary groups G_u and G_v conflict in S_G, so they cannot be in the same bundle. Therefore, no two adjacent nodes in G can be colored with the same color.

Second, we will show that r is the minimum number of colors we need to use for coloring G, by contradiction. Assume the minimum number of colors needed to color vertices in G be k, where $k < r$. Let the k colors used in vertex coloring be $C_0, C_1, \ldots, C_{k-1}$. For each color C_i, create a bundle B'_i, and for each node u with the color C_i, assign the elementary group G_u to B'_i. Add all the non-conflicting elementary groups to each bundle B'_i for maximality. The set of bundles B'_i is a bundle cover, because each elementary group G_u is in some B'_i (each node u in G is colored with some color C_i), and each bundle B'_i has an elementary group that is not in any other bundle (there is at least one node u that is colored with C_i). Now we have a bundle cover with k bundles, where $k < r$. This contradicts to the assumption that the bundle cover $B_0, B_1, \ldots, B_{r-1}$ is with the minimum number of bundles.

Therefore, finding a bundle cover with minimum number of bundles is NP-Complete. □

4.2. Algorithm for key bundle construction

Next we describe an algorithm that takes as input any bundle B in the bundle cover, constructed by the above algorithm, and computes a complete key tree for B. The following definition of a child is needed in stating our algorithm.

Let B be a bundle and let G and G' be two distinct elementary groups in B. Then, G' is a *child* of G if and only if $G' \subset G$ and B has no elementary group G'' such that $G' \subset G'' \subset G$.

The algorithm for constructing a complete key tree T for a given bundle B is shown below.

The child relation between elementary groups is computed once for all group pairs, using a new conflict matrix F. F is defined as follows:

$$F[i][j] = \begin{cases} \textbf{true} & \text{if } G_i \subset G_j, \\ \textbf{false} & \text{otherwise.} \end{cases}$$

The new conflict matrix F takes $O(mn)$ to compute, and it takes $O(m^3)$ to compute *isChild*, the child relationship matrix between all pairs of elementary groups.

Algorithm 3. Key Bundle Construction Algorithm

```
1: for each elementary group G_x in B
2:     add a node K_x to T;
3: for each elementary group G_x in B
4:     for each elementary group G_y in B
5:         if is Child(x, y)
6:             then add an edge (G_x, G_y) to T;
7: for each elementary group G_x in B
8:     if there is an edge (G_y, G_x) in T
9:      then
10:        add all users in G_x that are not in any
           child elementary group of G_x to C_y;
11:        add a node KC_y to T;
12:        add an edge (KC_y, K_x) to T;
13:     else
14:        add a key tree rooted at K_x whose
           leaves are labeled with
           the individual keys of the users in the
           elementary group G_x;
15: for each complement node KC_x in T
16:        add a key tree rooted at KC_x whose
           leaves are labeled with
            the individual keys of the users in
            the elementary group C_x;
```

$$isChild[i][j] = \begin{cases} \textbf{true} & \text{if } G_i \subset G_j, \text{ and no } G_x \text{ satisfies} \\ & \quad G_i \subset G_x \subset G_j, \\ \textbf{false} & \text{otherwise.} \end{cases}$$

The complexity of this algorithm is $O(m^2 + mn)$. The worst case is when this bundle includes all m elementary groups. The first loop in lines 1–2 takes $O(m)$ as there can be as many as m elementary groups in a bundle. The second loop in lines 3–6

takes $O(m^2)$. In the third loop in lines 7–14, there could be at most n users in G_x in the key tree rooted at K_x. Therefore, the third loop takes $O(mn)$. The last loop again takes $O(mn)$, since there could be at most $n-1$ users in the complement group C_x. In total, the complexity of this algorithm is $O(m^2+mn)$.

As an example, if this algorithm is applied to the system in Fig. 2, the algorithm constructs the key bundle shown in Fig. 4.

5. Key parcels

A bundle is defined as a maximal set of non-conflicting elementary groups in the system. From this definition the elementary group G_0 is in every bundle since it does not conflict with any other elementary group in the system. Thus, every key bundle is a complete key tree.

This feature of bundle maximality has one advantage and one disadvantage. The advantage is that the complement of any elementary group in a bundle B_j can be expressed as the union of some other elementary groups in B_j. Thus, securely multicasting a data item to the complement of any elementary group can be carried out efficiently. (In the simulation results below shown in Fig. 9 in Section 7, we show that the average number of encryptions required for a complement of an elementary group in a system with key bundles is much less than that with key parcels.) On the other hand, the disadvantage is that the number of keys needed in each key bundle is relatively large, and so the total number of keys in the server is relatively large. (In the simulation results below shown in Fig. 7 in Section 7, we show that the number of keys in a system with key bundles is far greater than that with key parcels.)

The disadvantage of bundle maximality outweighs its advantage in systems where users never need to securely multicast data items to the complements of elementary groups. Therefore, in these systems, we use "parcels", which are not maximal, instead of bundles, which are maximal. The definitions of parcels and parcel covers are given next.

A *parcel* of a system is a set of non-conflicting elementary groups of the system.

A *parcel cover* of a system is a sequence of parcels $(P_0,\ldots,P_{s-1})$ such that the following two conditions hold:

1. *Completeness*: Each elementary group of the system appears in some parcel P_i in the parcel cover.
2. *Compactness*: Each elementary group in each parcel P_i conflicts with at least one elementary group in each of the preceding parcels $P_0,\ldots,P_{i-1}$ in the parcel cover.

As an example, a parcel cover for the system in Fig. 2 is (P_0, P_1), where $P_0=\{G_0, G_1, G_2, G_4\}$ and $P_1=\{G_3\}$. Fig. 5 is a parcel cover (P_0, P_1) for the system in Fig. 2.

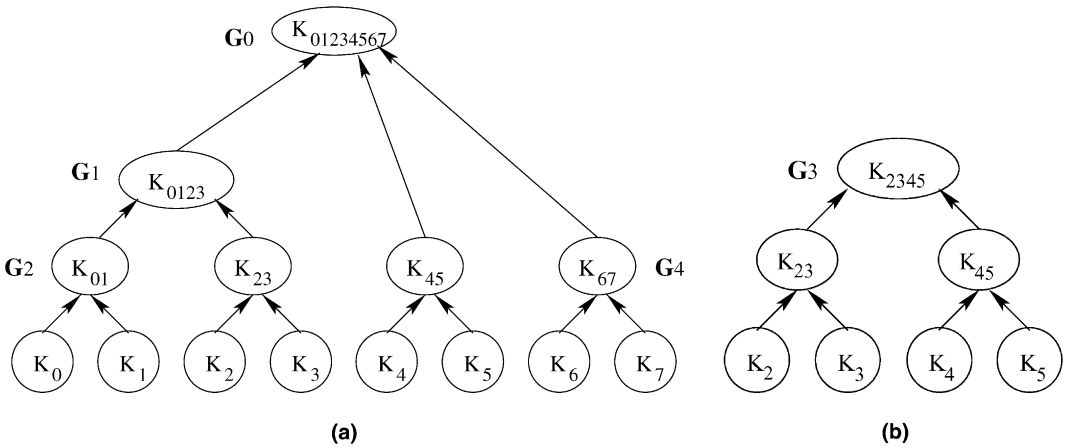

Fig. 5. The complete key trees for the parcel cover: (a) $P_0=\{G_0, G_1, G_2, G_4\}$ and (b) $P_1=\{G_3\}$.

The security keys for the elementary groups in a parcel can be arranged in a key tree that is not necessarily a complete tree. Fig. 5(a) shows the key tree for parcel P_0 consisting of the elementary groups G_0, G_1, G_2, and G_4. Fig. 5(b) shows the key tree for parcel P_1 consisting of the elementary group G_3. Note that the key tree for parcel P_1 is not a complete tree. We refer to a key tree that corresponds to a parcel as a *key parcel*.

6. Construction of key parcels

In this section, we describe a procedure that can be used by the server of a system to construct and maintain key parcels for that system. This procedure consists of two algorithms.

The first algorithm constructs a parcel cover for any given system. This algorithm takes any given system with m elementary groups $G_0, \ldots, G_{m-1}$ and constructs a parcel cover $(P_0, \ldots, P_{s-1})$ for the given system, $s \leqslant m$. This parcel cover construction algorithm uses the same data structures and the same NOCONFLICT function used in the bundle cover construction algorithm in Algorithm 2. The parcel cover construction algorithm is shown in Algorithm 4.

Note that the array added used in Algorithm 4 is actually a boolean array. The value of added[x] for each elementary group G_x is either 0 or 1, where 0 means it is not assigned to any parcel yet and 1 means it is assigned to some parcel.

Algorithm 4 Parcel Cover Construction algorithm

```
1:  s := 0;
2:  for x = 0 to m − 1
3:      if added[x] > 0 then goto line 2;
4:      P_s := P_s ∪ {G_x};
5:      added[x] := 1;
6:      for y = (x + 1) to m − 1
7:          if added[y] = 0 and NOCONFLICT
            (P_s, G_y)
8:          then P_s := P_s ∪ {G_y};
9:              added[y] := 1
10:     endfor;
11:     s := s + 1
12: endfor;
13: discard the empty parcels P_s, ..., P_{m−1}
```

The complexity of this parcel cover construction algorithm is $O(m^3)$, but in general the complexity of computing a parcel cover for a given system with the minimum number of parcels is NP-Complete. The proof is given in Theorem 2.

Theorem 2. *Given a system with elementary groups, computing a parcel cover of the system with the minimum number of parcels is NP-Complete.*

Proof. The proof consists of two parts. The first part shows that there is a polynomial algorithm that verifies whether a set of parcels is a parcel cover or not. The second part shows that a well-known NP-Complete problem, Vertex Coloring, can be reduced into computing a parcel cover with minimum number of parcels in polynomial time. The Vertex Coloring problem is defined as follows: given a graph $G = (V, E)$, each vertex is assigned with a color such that no adjacent vertices are in the same color. Finding an assignment with minimum number of colors is NP-Complete.

Part 1: Given a set of parcels, we can verify whether the set is a parcel cover or not in polynomial time. For each elementary group, find a parcel it belongs to. If there is any elementary group that does not belong to any parcel, the set of parcels is not a parcel cover. This step takes $O(ms)$, where m is the number of elementary groups, and s is the number of parcels. To be a parcel cover, each parcel has to have at least one elementary group that conflicts with some elementary group in every other parcel. Compute the conflict matrix $C[i][j]$ as described in Section 6, and compute the conflicts between parcels according to the matrix. We can verify all these conflicts in $O(m^2 + s^2)$ steps.

Part 2: Given a graph $G = (V, E)$, construct a system S_G as follows: for each node u in V, add an elementary group G_u to S_G. For each edge (u, v) in E, add a user $user_{uv}$ to both elementary groups G_u and G_v in S_G so that these elementary groups conflict with each other. Assume that we have the parcel cover $P_0, P_1, \ldots, P_{s-1}$ of the system S_G with minimum number of parcels.

First, we show that we can color the vertices with s colors. Let s colors to be $C_0, C_1, \ldots, C_{s-1}$. For each elementary group G_u in P_i, color the vertex u in G with color C_i. The system S_G is

constructed so that for any two adjacent nodes u and v in G, the corresponding elementary groups G_u and G_v conflict in S_G, so they cannot be in the same parcel. Therefore, no two adjacent nodes in G can be colored with the same color.

Second, we will show that s is the minimum number of colors we need to use for coloring G, by contradiction. Assume the minimum number of colors needed to color vertices in G be k, where $k < s$. Let the k colors used in vertex coloring be $C_0, C_1, \ldots, C_{k-1}$. For each color C_i, create a parcel P'_i, and for each node u with the color C_i, assign the elementary group G_u to P'_i. By the definition of vertex coloring, no two adjacent nodes are colored with the same color. This set of parcels P'_i is a parcel cover: Each elementary group G_u belongs to at least to one parcel, since each vertex u in G is colored with some color. Also, each parcel has at least one elementary group that was not in the preceding parcels, since for each color C_i, at least one vertex is colored with C_i and with no other color. Now we have a set of parcels with k parcels, where $k < s$. This contradicts to the assumption that the parcel cover $P_0, P_1, \ldots, P_{s-1}$ is with the minimum number of parcels.

Therefore, finding a parcel cover with minimum number of parcels is NP-Complete. □

The second algorithm computes a key tree (i.e. a key parcel) for each parcel in the parcel cover constructed by the first algorithm. This algorithm is exactly the same as the one presented in Section 4.2.

7. Simulation results

In this section, we present the results of simulations that we carried out to demonstrate the feasibility of key bundles and key parcels. The simulation is written in Java. For each system, we decide the number of users, the number of elementary groups, and the average number of elementary groups that each user joins. For the given average number of elementary groups c, we generate a random number k_u from Poisson distribution for each user u, and user u joins k_u elementary groups. The reason that we use Poisson distribution is to simulate the effect of a few users that join significantly more elementary groups than the average. Each elementary group that u joins is chosen randomly among all the elementary groups from uniform distribution. In our simulation shown in Figs. 6–9, we used a class of synthetic systems with the following properties:

1. The number of users in each system varies from 1000 to 10000.
2. Each system has 500 elementary groups.
3. In each system, a user joins 2 elementary groups on average.

Each system is simulated 100 times. The averages of the following four items are computed over

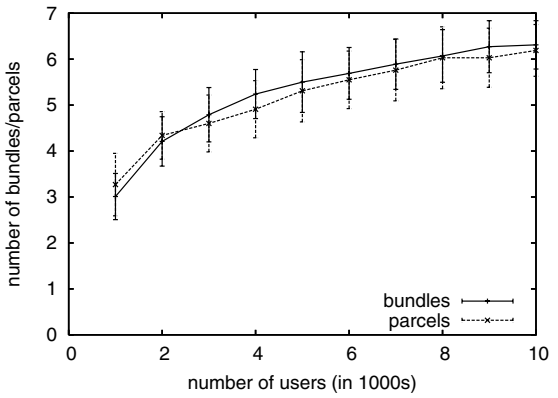

Fig. 6. Number of bundles or parcels.

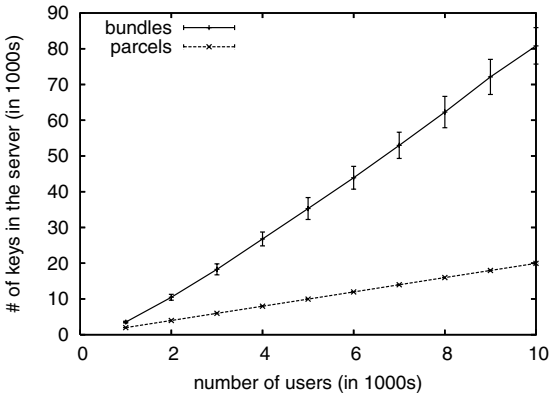

Fig. 7. Number of keys in the server.

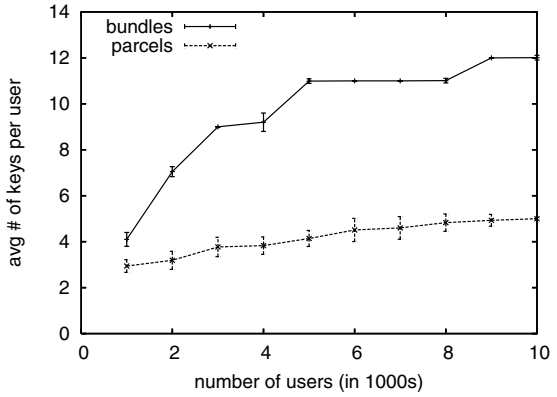

Fig. 8. Number of keys per user.

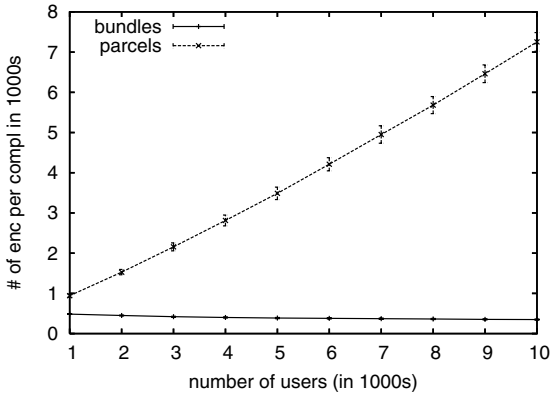

Fig. 9. Number of encryptions per complement.

the 100 simulation runs for each system: the number of bundles or parcels in the system cover, the total number of keys in the server, the number of keys per user, and the number of encryptions needed to multicast a data item to the users in the complement of an elementary group. The results of these simulations are shown in Figs. 6–9.

As shown in Fig. 6, the number of bundles in a bundle cover more or less equals the number of parcels in a parcel cover. Note that this number increases logarithmically as the number of users in the simulated system grows.

As shown in Fig. 7, the number of keys in systems that use key bundles and the number of keys in systems that use key parcels grow linearly as the number of users in the system increases. However, the number of keys in the case of key bundles grows much faster in the case of key parcels. This is because each key bundle is a complete key tree while each key parcel is not necessarily complete.

As shown in Fig. 8, the number of keys that each user needs to store increases as a logarithm function with the number of users in the system.

Fig. 9 shows how many encryptions are needed per complement of an elementary group when the Key Bundle or the Key Parcel approach is used. The Key Bundle approach shows better performance than the Key Parcel approach, and the difference becomes greater as the number of users increases. The number of encryptions in the Key Bundle approach decreases as the number of users increases, while that in the Key Parcel approach increases linearly as the number of users increases. The following two paragraphs explain why.

When the Key Bundle approach is used, the average number of encryptions performed for a complement of an elementary group decreases as the number of users increases (the actual number is from around 500 to 400). As the number of users increases, the probability of two groups' conflicting increases. Therefore, the average number of groups can be put in a bundle decreases. Since we use the keys of other groups in the same bundle that contains an elementary group to encrypt for the complement of the elementary group (for example, we use K_{01} and K_{67} for $\neg G_3$ in Fig. 4), the number of encryptions decreases as the number of users increases.

On the other hand, the number of encryptions for a complement performed in the Key Parcel approach linearly increases as the number of users grows. It is because that in the Key Parcel approach, we use the individual keys of users that are not in any elementary group in the same parcel. For example, to securely multicast to $\neg G_3$ in Fig. 5, we need to use the individual keys of K_0, K_1, K_6, K_7. As the number of users increases, the probability of two groups' conflicting increases, so the average number of groups in a parcel decreases. Therefore, as the number of users increases, we need to use more individual keys to encrypt for a complement.

We also ran simulations with the following parameters.

1. The number of users in each system is 5000.
2. Each system has 500 elementary groups.
3. In each system, the average number of elementary groups each user joins varies from 2 to 2.8.

Each system is simulated 100 times. The averages of the following two items are computed over the 100 simulation runs for each system: the number of bundles or parcels in the system cover and the total number of keys in the server. The results of these simulations are shown in Figs. 10 and 11.

As shown in Fig. 10, the number of bundles in a bundle cover more or less equals the number of parcels in a parcel cover. As the average number of elementary groups each user joins increases, more conflicts between elementary groups exist, so the number of bundles or parcels increases.

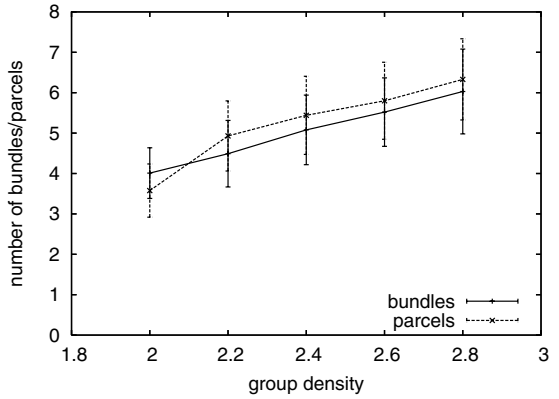

Fig. 10. Number of bundles or parcels.

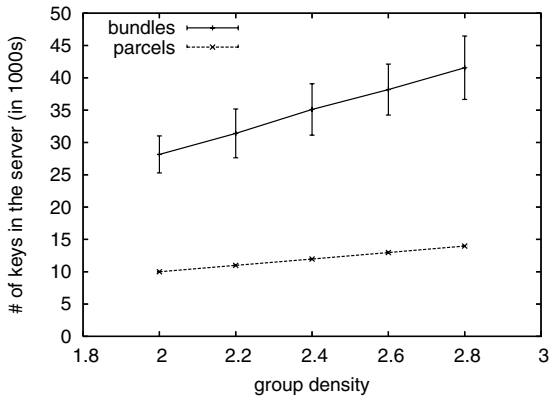

Fig. 11. Number of keys in the server.

As shown in Fig. 11, the number of keys in servers that use key bundles and the number of keys in servers that use key parcels grow linearly as the average number of elementary groups each user joins in the system increases. However, the number of keys in the case of key bundles grows much faster in the case of key parcels. This is the same tendency with Fig. 7 where the number of users in the system varies.

8. Bundle/parcel cover reconfiguration

We have shown how to construct a bundle/parcel cover for a system with elementary groups. As we discussed in Section 2, users in this system can join or leave any elementary group in the system, or leave the system altogether, and these activities may require to change the security keys associated with the elementary groups which users join or leave. In this section, we discuss how to handle these changes. Handling join and leave operations does not differ either in a system based on key bundles or in a system based on key parcels, so the following discussion is based on key parcels.

When a user u joins an elementary group G in a parcel P, this join operation may render G to conflict with other elementary groups in P. Also, when a user u leaves an elementary group G in a parcel P, this leave operation may render G to conflict with its child group, or not to conflict with some elementary groups in other parcels. We do not expect that the parcel cover of the system will be reconfigured for a certain period, say a year or six months, according to the nature of the application. During this period, the changes in conflicts will be handled tentatively.

Join operations can be handled by assigning additional individual keys to the joining users. If user u_i, who is already in a group G_j, joins another group G'_j in the same parcel P, then the two groups G_j and G'_j become conflicting and should be assigned to different parcels. Instead of recomputing a new parcel cover for the system, when user u_i, who is already in group G_j, joins group G'_j, another

user u'_i joins G'_j instead of u_i. In this case a new individual key is assigned to user u'_i (although the two users u_i and u'_i in fact correspond to the same physical user). The net effect of this solution is that the two groups G_j and G'_j are still not conflicting and can remain in the same parcel P. The conflict will be resolved when the current period expires and the parcel cover is recomputed.

For the leave operations, we assume that it is okay for leaving users to be able to understand the communications until the current period expires and the parcel cover is recomputed. In fact, this is what happens in many subscription-based services. Even if users specify their intentions to leave any service or the whole system altogether in the middle of pre-defined period, they can still receive the service until the current subscription period expires.

9. Server distribution

So far, we have presented the server as if it were a single entity in the system. In fact, the server does not have to be a single entity. In this section, we present a way to distribute the functionality of the server among several servers.

As explained earlier, the server has two tasks:

1. *Key Management*: The server generates new keys when a user joins or leaves a group, and distributes the newly generated keys to the appropriate set of users.
2. *Secure Multicast*: If a user does not know appropriate group keys to secure its message, it has to encrypt the message using its individual key and forward it to the server. The server then decrypts the message using the individual key of the user, encrypts the message using the appropriate group keys, and multicasts the message to the ultimate destination.

We deploy two types of servers to perform these two tasks: key servers to perform key management and multicast servers to perform secure multicast. In the following we present an architecture where multiple key and multicast servers are deployed in a system that use key parcels. (A similar architecture can be presented where multiple key and multicast servers are deployed in a system that use key bundles).

9.1. Architecture

Consider a system that has been partitioned into a parcel cover $(P_0, \ldots, P_{s-1})$, as described in Section 6. In this case, s key servers, $KS_0, \ldots, KS_{s-1}$, are deployed such that each key server KS_k maintains the keys in parcel P_k.

When a user u_i leaves a group G_j in parcel P_k then only key server KS_k needs to update a small number of keys in its parcel of P_k and to inform the other users in G_j. (The updated keys are multicast to the other users in G_j according to the key tree algorithm [4].) Similarly, when a user u_i joins a group G_j in parcel P_k then only the key server KS_k needs to update a number of keys in its parcel and to inform the other users in G_j.

The multicast servers know individual keys of all users and group keys of all groups in the system. Thus whenever key server KS_k updates the keys of its parcel P_k, KS_k securely multicast the updated group keys in its parcel to all the multicast servers in the system.

9.2. Convergence

At an ideal state of this system, each key K in each key parcel is known to user u_i if and only if either K is the individual key of u_i or u_i is a descendant of node K in the key parcel. Such a system is guaranteed to converge to an ideal state provided its initial state is ideal.

When a user u_i leaves a group G_j in a parcel P_k, the key server KS_k needs to update the keys that are on the path from the individual key K_i to the group key GK_j in the parcel. Now some keys of the parcel have been updated and the updated keys are not known to all the users that need to know them, so this state is not ideal. Key server KS_k uses the key distribution protocol in [4] to distribute the updated keys. The key distribution protocol ensures that only the users that are descendants of a key K can acquire the updated key K'. After the key distribution is completed, the system is in an ideal state.

Similar argument can be given for the case when user u_i joins a group G_j in a parcel P_k. User u_i has individual key K_i shared with the servers. The system is not in an ideal state until user u_i acquires all the keys on the path from K_i to GK_j in the key parcel. Key server KS_k uses the key distribution protocol in [4] to transmit keys, and after the key transmission is finished the system is in an ideal state.

So far, we have only discussed how to use key servers to store each key parcel. These key servers can be implemented using the protocol in [4], but also can be implemented using the technique in [13,21].

10. Conclusion

We consider a system where each user is in one or more elementary groups. In this system, arbitrary groups of users can be specified using the operations of union, intersection, and complement over the elementary groups in the system. Each elementary group in the system is provided with a security key that is known only to the users in the elementary group and to the system server. Thus, for any user u to securely multicast a data item d to every user in an arbitrary group G, u first forwards d to the system server which encrypts it using the keys of the elementary groups that comprise G before multicasting the encrypted d to every user in G. Every elementary group is also provided with a key tree to ensure that the cost of changing the key of the elementary group, when a user leaves the group, is small. We describe two methods for packing the key trees of elementary groups into key bundles and into key parcels.

Packing into key bundles has the advantage of reducing number of encryptions needed to multicast a data item to the complement of an elementary group. Packing into key parcels has the advantage of reducing the total number of keys in the system. We apply these two methods to a class of synthetic systems: each system has from 1000 to 10 000 users and 500 elementary groups, and a user in each system is in 2 elementary groups on average. Simulations of these systems show that our proposal to pack key trees into key bundles and key parcels provides a reasonable performance to the whole system. The number of keys stored per user in the case of key bundles is 12 for 10 000 user system, while that in the case of key parcels is 5. Instead, the number of encryptions needed for a complement in the case of key bundles is far less than that in the case of key parcels as shown in Fig. 9.

Key bundles and key parcels are two extremes in the spectrum of solutions supporting secure group communication in many groups. Key bundles put emphasis on supporting complement of an elementary group with relatively large number of keys in the system, while key parcels save the number of keys in the system but in result cannot support complement of an elementary group without considerably far more number of encryptions. As a future work, we would like to find a hybrid between these two methods, which needs less number of keys in the system than in the case of key bundles and at the same time supports complement of an elementary group with less number of encryptions than in the case of key parcels.

We are also interested in conducting a case study of these methods in a real world application. The case study includes to define appropriate scopes of elementary groups and to maintain key bundles or key parcels accordingly. As a typical application of secure group communication, a peer-to-peer knowledge sharing system can take advantage of these methods. An elementary group will represent an access control list for a specific domain of knowledge. Using key bundles or key parcels, users may securely communicate with any set of users according to the needs at the given time. For example, any two taskforces in a company may merge temporarily for a current task. They do not have to change their security keys, nor need to initialize a new security domain. Still, the users in two taskforces can securely communicate with one another.

As described in Section 3, if the sender of message m does not know the keys required to encrypt m appropriately, the system server has to encrypt m and multicast. This requirement for the server's help may cause performance bottleneck at the server. To reduce the workload of the system server, we have discussed in Section 9 how multiple serv-

ers may be placed and coordinated to work in a distributed manner. Moreover, rekeying does not need to occur for every single join/leave. In [16], batch rekeying showed favorable performance. Also exposure-oriented rekeying in [22] can be used to trigger rekeying process.

References

[1] L. Gong, Enclaves: enabling secure collaboration over the Internet, IEEE Journal of Selected Areas in Communications 15 (3) (1997) 567–575.

[2] S. Mittra, Iolus: a framework for scalable secure multicasting, in: Proceedings of the ACM SIGCOMM'97, ACM Press, 1997.

[3] D.M. Wallner, E.J. Harder, R.C. Agee, Key management for multicast: Issues and architectures, RFC 2627, 1999.

[4] C.K. Wong, M. Gouda, S.S. Lam, Secure group communications using key graphs, IEEE/ACM Transactions on Networking (TON) 8 (1) (2000) 16–30.

[5] M. Steiner, G. Tsudik, M. Waidner, Key agreement in dynamic peer groups, IEEE Transactions on Parallel and Distributed Systems 11 (8) (2000) 769–780.

[6] L. Gong, N. Shacham, Multicast security and its extension to a mobile environment, Wireless Networks 1 (3) (1995) 281–295.

[7] A. Ballardie, Scalable multicast key distribution, RFC 1949, 1996.

[8] I. Chang, R. Engel, D.D. Kandlur, D.E. Pendarakis, D. Saha, Key management for secure internet multicast using boolean function minimization techniques, in: IEEE INFOCOM 1999, vol. 2, 1999, pp. 689–698.

[9] D. Naor, M. Naor, J.B. Lotspiech, Revocation and tracing schemes for stateless receivers, in: CRYPTO, 2001.

[10] O. Rodeh, K. Birman, D. Dolev, The architecture and performance of security protocols in the ensemble group communication system: using diamonds to guard the castle, ACM Transactions on Information and System Security (TISSEC) 4 (3) (2001) 289–319.

[11] S. Setia, S. Koussih, S. Jajodia, E. Harder, Kronos: a scalable group re-keying approach for secure multicast, in: IEEE Symposium on Security and Privacy, 2000.

[12] 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.

[13] P.P. Lee, J.C.S. Lui, D.K. Yau, Distributed collaborative key agreement protocols for dynamic peer groups, in: ICNP, 2002.

[14] S.S. Kulkarni, B. Bruhadeshwar, Reducing the cost of the critical path in secure multicast for dynamic groups, in: Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW'02), 2002.

[15] M.G. Gouda, C.-T. Huang, E. Elnozahy, Key trees and the security of the interval multicast, in: 22nd International Conference on Distributed Computing Systems (ICDCS'02), 2002, pp. 467–468.

[16] X.S. Li, Y.R. Yang, M.G. Gouda, S.S. Lam, Batch rekeying for secure group communications, in: The Tenth International World Wide Web Conference on World Wide Web, ACM Press, 2001.

[17] Y.R. Yang, X.S. Li, X.B. Zhang, S.S. Lam, Reliable group rekeying: a performance analysis, in: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM Press, 2001.

[18] J. Snoeyink, S. Suri, G. Varghese, A lower bound for multicast key distribution, in: IEEE INFOCOM 2001, 2001.

[19] E. Jung, A.X. Liu, M.G. Gouda, Key bundles and parcels: secure communication in many groups, in: LNCS 2816, Group Communications and Charges, 2003.

[20] M.T. Goodrich, J.Z. Sun, R. Tamassia, Efficient tree-based revocation in groups of low-state devices, in: CRYPTO'04, 2004.

[21] O. Rodeh, K. Birman, D. Dolev, Optimized group rekey for group communication systems, in: Symposium on Network and Distributed System Security, 2000.

[22] Q. Zhang, K.L. Calvert, On rekey policies for secure group applications, in: ICCCN, 2003.

Eunjin Jung is a Ph.D candidate in the Department of Computer Sciences in the University of Texas at Austin. She received the BS degree in Computer Science from Seoul National University in 1999, and the M.S. degree in Computer Sciences from the University of Texas at Austin in 2002. Her research interests are in security in distributed systems. Currently she is working on security issues in certificate systems.

Alex X. Liu received the B.S. degree in computer sciences from Jilin University, China, in 1996. He received the M.S. degree in computer sciences from the University of Texas at Austin in 2002, where he is currently working toward a Ph.D. degree in computer sciences. He won the 2004 IEEE&IFIP William C. Carter Award, the 2004 National Outstanding Oversea Students Award sponsored by the Ministry of Education of China, and the 2005 George H. Mitchell Award for Excellence in Graduate Research in The University

of Texas at Austin. His research interests include network security, computer security, networks protocols, and distributed systems.

Mohamed G. Gouda was born in Egypt. His first B.Sc. was in Engineering and his second was in Mathematics; both are from Cairo University. Later, he obtained M.A. in Mathematics from York University and Masters and Ph.D. in Computer Science from the University of Waterloo. He worked for the Honeywell Corporate Technology Center in Minneapolis 1977–1980. In 1980, he joined the University of Texas at Austin where he currently holds the Mike A. Myers Centennial Professorship in Computer Sciences. He spent one summer at Bell labs in Murray Hill, one summer at MCC in Austin, and one winter at the Eindhoven Technical University in the Netherlands. His research areas are distributed and concurrent computing and network protocols. In these areas, he has been working on abstraction, formality, correctness, nondeterminism, atomicity, reliability, security, convergence, and stabilization. He has published over sixty journal papers, and over eighty conference and wokshop papers. He supervised Nineteen Ph.D. Dissertations. Four of his former Ph.D. students are now Full Professors at the University of Colorado at Colorado Springs, the University of California at Santa Barbara, the University of North Carolina at Chapel Hill, and the Ohio State University. Another six of his former Ph.D. students are now Associate and Assistant Professors in the United States of America. Prof. Gouda was the founding Editor-in-Chief of the Springer-Verlag journal Distributed Computing 1985–1989. He served on the editorial board of Information Sciences 1996–1999, and he is currently on the editorial boards of Distributed Computing and the Journal of High Speed Networks. He was the program committee chairman of ACM SIGCOMM Symposium in 1989. He was the first program committee chairman of IEEE International Conference on Network Protocols in 1993. He was the first program committee chairman of IEEE Symposium on Advances in Computers and Communications, which was held in Egypt in 1995. He was the program committee chairman of IEEE International Conference on Distributed Computing Systems in 1999. He is on the steering committee of the IEEE International Conference on Network Protocols and on the steering committee of the Symposium on Self-Stabilizing Systems, and was a member of the Austin Tuesday Afternoon Club from 1984 till 2001. Prof. Gouda is the author of the textbook "Elements of Network Protocol Design", published by John-Wiley & Sons in 1998. This is the first ever textbook where network protocols are presented in an abstract and formal setting. He coauthored, with Tommy M. McGuire, the monograph "The Austin Protocol Compiler", published by Springer in 2005. He also coauthored, with Chin-Tser Huang, the monograph "Hop Integrity in the Internet", to be published by Springer in 2006. Prof. Gouda is the 1993 winner of the Kuwait Award in Basic Sciences. He was the recipient of an IBM Faculty Partnership Award for the academic year 2000–2001 and again for the academic year 2001–2002 and became a Fellow of the IBM Center for Advanced Studies in Austin in 2002. He won the 2001 IEEE Communication Society William R. Bennet Best Paper Award for his paper "Secure Group Communications Using Key Graphs", coauthored with C.K. Wong and S.S. Lam and published in the February 2000 issue of the IEEE/ACM Transactions on Networking (Volume 8, Number 1, Pages 16–30). In 2004, his paper "Diverse Firewall Design", coauthored with Alex X. Liu and published in the proceedings of the International Conference on Dependable Systems and Networks, won the William C. Carter Award. More detailed information about the career of Prof. Gouda can be found on his website http://www.cs.utexas.edu/users/gouda