A Middleware Service for Secure Group Communication in Mobile Ad Hoc Networks

Stephen S. Yau and Xinyu Zhang
Computer Science and Engineering Department
Arizona State University
Tempe, AZ 85287, USA
{yau, zhxinyu}@asu.edu

Abstract

Secure group communication in mobile ad hoc networks is often dynamic and impromptu, and thus requires efficient and automated secure group management and seamless combination of secure groups with distributed applications running upon them. Existing approaches to secure group communication cannot satisfy these requirements. In this paper, an automated secure group management approach is presented. Based on this approach, a middleware service for secure group communication is developed to facilitate development and execution of distributed applications using secure group communication in mobile ad hoc networks. This middleware service is implemented in a context sensitive middleware RCSM.

Keywords: *Secure group communication, mobile ad hoc networks, middleware service, context-sensitive middleware, distributed applications, ubiquitous computing.*

1 Introduction

Secure group communication (SGC) is required by many applications, such as net meeting and on-line bidding. A secure group [1] consists of a number of group members. A secret group key is shared by all group members for secure communication among them. One member can broadcast a message to other members in the same secure group. There may be group members joining/leaving the group frequently. When such a group change occurs, the secret group key needs to be renewed (rekeying) to prevent previous group members from accessing new messages and new group members from accessing past messages.

Due to rapid progress in low-cost, short-range, and low-power wireless ubiquitous computing devices, as well as appropriate middleware techniques [2, 3], mobile ad hoc networks have become more realistic in various computer applications. However, existing approaches to SGC [4-9] cannot be applied in such environment.

Compared with a secure group over an infrastructure-based (wired or station-based wireless) computer network, a secure group consisting of a set of ubiquitous computing devices in a mobile ad hoc network has two characteristics: (1) Group members have limited capability: CPU, memory, power, and network bandwidth. (2) Mobility and limited capability makes secure group communication potentially dynamic and impromptu with short life cycle.

Because of these two characteristics, mobile ad hoc networks pose more requirements on secure group communication:

(1) Efficient and automated secure group management is required for fast group set up and for different secure groups to share secret keys.

(2) Authentication of group members should be performed in a distributed fashion.

(3) Collaboration of security group management and applications using secure group communication needs to be addressed because of the volatility of secure groups.

A mobile ad hoc network usually consists of heterogeneous platforms with different techniques and standards. Under these circumstances, middleware can be effectively used to provide interoperability, development, and run-time support for distributed applications in heterogeneous ad hoc network environments.

To address the above issues, we will present a middleware service to manage secure groups and to facilitate development and execution of applications with secure group communication. Using the middleware service, distributed applications can send/receive secure messages among members of a secure group. Prior to application execution, members of secure groups need to be identified. When distributed applications are executed, secure groups are managed automatically according to identified members beforehand. This middleware service can easily be incorporated in the reconfigurable context-sensitive middleware RCSM [2].

2 Current State of Art

Early research on secure group communication mainly focused on network protocol and group key generation and distribution techniques [4,5]. These approaches do not scale well with group size. Later work deals scalability of secure groups [6-8]. The approach provided in [6] has rekeying cost O(n) because it needs to check all group members to generate rekeying message. Approaches in [7,8] have rekeying cost O(log(n)), where n is the group size. Griffin, et al [9] presented an approach to reduce rekeying cost when a mobile member moves from one area to another. It is not associated with any specific secure group rekeying approach.

Our Reconfigurable Context-Sensitive Middleware (RCSM) [2] provides some middleware services for distributed applications in mobile ad hoc networks to use various data about the surrounding environment to control their interactions. Situation-Aware Interface Definition Language (SA-IDL) is developed to describe situations and situation-aware components [3]. However, secure group communication has so far not been considered in RCSM.

3 Our Approach

Our middleware service for secure group communication (SGC) in mobile ad hoc networks involves *devices*, *secret keys*, *secure groups*, and *distributed applications.* A device is a computing device in the network. A secret key is shared by multiple devices. A device can encrypt a message with a secret key. Other devices in the network can decrypt the cipher text using the same secret key. Within the network, various secure groups are formed and dismissed dynamically. Within each secure group, various devices may join/leave it dynamically. A device can be in several secure groups simultaneously. Our middleware service deals with SGC within a secure group only. If two or more secure groups need inter-group secure group communication, i.e., any member in these groups needs to receive secure broadcast from the members in these secure groups, all members from these groups can form a large secure group containing all these groups using the same group key.

Figure 1 shows our Secure Group Communication Service (SGCS) in the context-sensitive middleware on a ubiquitous computing device. The R-GIOP and system context reflector provide situation-sensitive interconnection for ubiquitous computing devices through embedded OS. *SA-IDL* is an interface definition language, which can describe situation-sensitive attributes [3]. *Distributed applications* involve communication among the members in each secure group. SGCS consists of three types of components: *SGCS Daemon, Group Services ,and SGCS API.*

SGCS Daemon is used to create and manage Group Services. A *Group Service* provides core functionality for a secure group on a device. On a device, there is a Group Service for each secure group that the device is in. A Group Service will be created if a new secure group is detected, and will be deleted if the secure group is dismissed.

SGCS API is used for distributed applications to use secure group functionality.

Authentication relations are used to define secure groups prior to secure group set up.

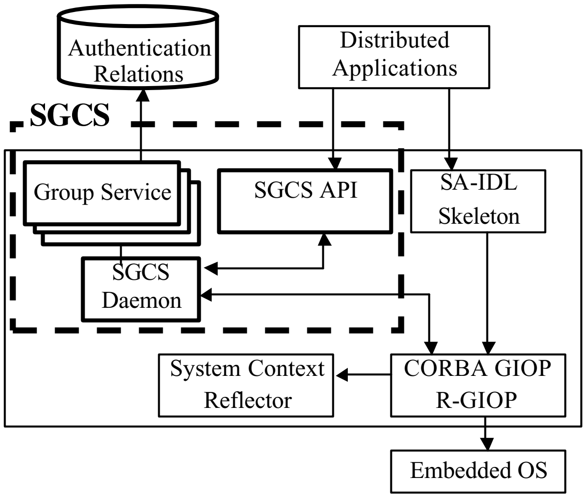

Figure 1: SGCS in context-sensitive middleware.

The SGCS is used to facilitating development and execution of distributed applications with secure group communication among the members of each secure group. The process for developing and executing each of these distributed applications using SCGS is summarized as follows:

1) Develop a distributed application with secure group communication.
2) Identify the members of each secure group that is required by the distributed application. Information of secure groups is stored as authentication relations.
3) When the distributed application is running, secure groups are managed by SGCS automatically to support secure communication among the group members of each secure group.

4 Development of A Distributed Application with Secure Group Communication

We have developed an approach to developing situation-aware application software in RCSM [3]. To incorporate secure group communication in a distributed application using SGCS, we need to do the following on the situation-aware application development process using RCSM [3]:

In the first step of the process, generating a situation-aware object interface using SA-IDL, we need to include situations that will trigger secure group management operations. A situation is a logical expression on a set of contexts relevant to the application software on the device over a period of time. Secure group management operations will be discussed in Section 6. Some situations and related secure group management operations that need to be included in situation-aware object interfaces are listed below:

- Situations when a device does not need to broadcast or receive secure messages in a secure group any more. These situations will trigger a "deleting device" operation.
- Situations when a secure group is no longer needed. These situations will trigger a "dismissing secure group" operation.

In the second step, generating a Situation-Aware Adaptive Object Container, nothing specific to SGCS needs to be done at this step.

In the third step, generating the code of the situation-aware objects, when the distributed application needs to broadcast a secure message to all other members in a secure group in which the device is in, a method *sgc_send* provided in SGCS API needs to be called. This method queries SGCS daemon for the group key of the secure group, encrypts the message with the group key, and broadcasts the message to other members in the secure group.

When the distributed application receives a secure message broadcasted from another device in the same group, a method *sgc_receive* provided in SGCS API needs to be called to decrypt the message using the group key.

5 Identify Secure Group Members

In the second step of our approach, the members of each secure group need to be identified. Before discussing this step, let us introduce the following notations:

- Uppercase letters denote sets. Specifically, D, K and P denote a device set, a key set, and an authentication relation set, respectively.
- Lowercase letters denote items. Specifically, d, x, and y denote devices, k a secret key, gkd a Group Key Distributor, and lkd a Local Key Distributor.
- i, j, a, and b are used as subscripts of items. For instance, d_i, d_j denotes two devices.
- {msg}k denotes a cipher-text which is the message " msg" encrypted with the key k.
- $d_1 \rightarrow d_2$:msg denotes that device d_1 sends a message msg to device d_2.

In our approach, a new group member can be authenticated by any existing group members in order for the new group member to join the secure group. We use authentication relation to define such a relation.

Definition 1: *An authentication relation* is a 4-tuple (g, d_a, d_b, k), where g is a secure group and k is a secret key so that if device d_a is a member of the secure group g, d_a can authenticate device d_b using the key k for d_b to join the group g. d_a is called the *authenticating device* of d_b for the secure group g.

In this paper, if there is only one group to discuss, the group can be omitted from an authentication relation.

Definition 2: A *device tree* is a set of devices organized as a tree, in which each node is a device and each edge is an authentication relation.

All authentication relations in a device tree form an authentication relation set. A device tree can be specified by its authentication relation set. By adding an edge, i.e., an authentication relation, two device trees can be merged into one device tree.

Definition 3: A *secure group* is a group of devices in a device tree defined by an authentication relation set P. The root device is called the *Group Key Distributor (GKD)*. Each child device of the root device is called a *Local Key Distributor (LKD)*. In the entire device tree, the sub-tree rooted by a LKD is called a *subgroup*.

The GKD generates the group key and distribute it to all group members. In a subgroup, the *LKD* manages group members in the subgroup and participates in group key distribution. Devices in a secure group are divided into two layers. All LKDs and the GKD form a special secure group within the entire secure group, which is called the *Management Layer*. The remaining devices are in the *Group Member Layer* of disjoint subgroups. Figure 2 shows the structure of a secure group.

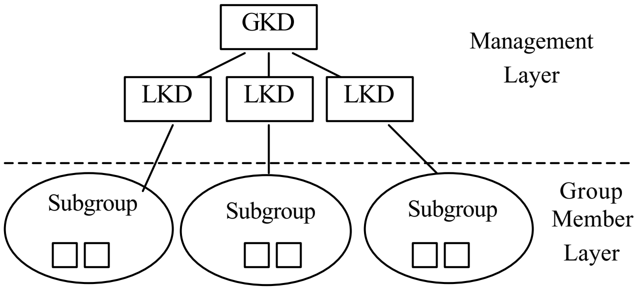

Figure 2: A secure group with multiple subgroups

A secure group is *complete* if all devices identified for the secure group form a device tree rooted by the GKD according to authentication relations for the secure group. All identified devices for a secure group can be authenticated by the GKD only if the secure group is complete. The following is the algorithm of Step 2) of our approach to make a secure group complete:

2.1) The process starts with an empty device set D and an empty authentication relation set P.

2.2) Identify devices which will be in the secure

group and add them into D.
2.3) Identify a device gkd ∈ D as the GKD.
2.4) Identify subgroups. For each subgroup, identify a device from D as its LKD.
2.5) For each LKD lkd, find a secret key k shared by lkd and gkd. If there is no such secret key, generate one. Add authentication relation (gkd, lkd, k) into P.
2.6) Assign each remaining device into a subgroup.
2.7) For each subgroup do the following:
    2.7.1) As each device is a single-node device tree, the subgroup consists of a device tree T containing a list of disjoint device trees.
    2.7.2) Find two devices d_1 and d_2 so that d_1 is the root of a device tree in T, d_2 is in another device tree in T. Find a secret key k shared by d_1 and d_2. If there is not such a key, generate one.
    2.7.3) Add (d_1, d_2, k) to P. Thus the two device trees containing d_1 d_2 merge into one device tree in T.
    2.7.4) If T contains only one device tree, the process for the subgroup ends. Otherwise go back to Step 2.7.2).
2.8) The process stops.

6 Automated Secure Group Management

Given a secure group {gkd, lkd_1, lkd_2, .., $lkd_{n,}$, d_1, d_2, .., $d_{m,}$ }, n≥1, m≥0, with an authentication relation set P, where gkd is the GKD and each lkd_i is a LKD, the secure group management is to generate and distribute secret keys needed for the secure group using existing keys in P when the secure group changes. The existing keys of a secure group are called the *base keys* of the group, and the additional keys generated by the devices in the secure group are called the *derived keys* of the group. A secure group will have the following derived keys:

- K_g – The secure group key shared by all devices in the secure group.
- k_{mlk} – The management layer key shared by {gkd, lkd_1, lkd_2, .., lkd_n}.
- k_{sub-i} – The local subgroup key of a subgroup i shared by lkd_i and all its subgroup members.
- k_{m-x} – The membership key of a device d_x shared by d_x and its LKD.
- k_{pc-y} – The parent-child key of device d_y, shared by d_y and its authenticating device.

The secure group management for a secure group is performed in the Group Service for the secure group. The Group Service receives and handles secure group operation requests continuously until the secure group has been dismissed. There is a list of predefined operations that can be served by Group Service. Each operation represents a generic secure group activity, including initiating a secure group, adding group members, and renewing the group key. A secure group management operation can be invoked by secure group communication request from the distributed application, by a detected situation or by another operation. Our approach uses the following seven pre-defined operations for secure group management: initiating group, adding members, deleting members, renewing local subgroup key, renewing group key, dismissing secure group, querying group key.

The following is the algorithm for Step 3) using secure group operations:

3.1) When the device is turned on, SGCS Daemon is started and executes the following until the device is turned off.
3.2) SGCS Daemon is waiting until it receives an operation request from a distributed application, a detected situation, or an operation.
3.3) Check if the Group Service of this secure group exists or not. If yes, go to 3.5). Otherwise, go to 3.4).
3.4) SGCS Daemon creates one Group Service for the secure group.
3.5) Check if the secure group has been initiated or not. If yes, go to 3.7). Otherwise, go to 3.6).
3.6) Invoke the " Initiating group" operation.
3.7) Check if the device has been added to the secure group or not. If yes, go to 3.9). Otherwise, go to 3.8).
3.8) Invoke the "Adding device" operation.
3.9) If the received operation is " Initiate group" operation, go to 3.2). Otherwise, go to 3.10).
3.10) Invoke the received operation.
3.11) Go to 3.2) to wait for another operation request.

The functions and algorithms of these pre-defined operations are described below:

- **Operation 1: Initiating group – Init()**

This operation is to set up the secure group and can be done by the following algorithm:

(1.1) Device gkd generates a secret key k_{mlk}.
(1.2) For each subgroup, gkd sends message { k_{mlk} }k_{lkd} to lkd, where lkd is the LKD of the subgroup and (gkd, lkd, k_{lkd}) ∈ P.
(1.3) For each subgroup, invoke the operation "Adding(lkd)".
(1.4) The operation stops.

- **Operation 2: Adding members -- Adding(d)**

This operation is to add a device tree rooted by device d to a subgroup so that the devices in the added tree have the most recent group key. Device lkd is the LKD of the subgroup. This operation can be performed by the following algorithm:

(2.1) Let device set C={d}.
(2.2) If C is empty, go to (2.9). Otherwise, find x∈C

```
        and let C=C-{x}.
    (2.3) If x has a membership key already, or x is not
        available currently, go to (2.2).
    (2.4) Find $(x', x, k_{x'-x}) \in P$.
    (2.5) x' authenticates x, and generates the
        membership key $k_{m-x}$, and the parent-child
        key of x' and x, $k_{pc-x}$.
    (2.6) $x' \rightarrow lkd: \{ k_{m-x}, x\}k_{m-x'}$.
    (2.7) $x' \rightarrow x : \{ k_{m-x}, lkd_i\}k_{x'-x}, \{ k_{pc-x}\}k_{x'-x}$ .
    (2.8) $C = C \cup \{ y \mid (x, y, k_{x-y}) \in P$ and y is not a
        group member$\}$. Go to (2.2).
    (2.9) Invoke " Renewing('add')".
    (2.10) The operation stops.
```

- **Operation 3: Deleting members -- Deleting(d)**

This operation is to delete a device tree rooted by device d from a subgroup. The operation can be done by the following algorithm:

```
    (3.1) Let C={d}.
    (3.2) Find $x \in C$. If C is empty, go to (3.5). Let
        C=C-{x}.
    (3.3) Delete the registration of x from the subgroup.
    (3.4) Let $C \cup \{ y \mid (x, y, k_{x-y}) \in P$ and y is in the
        subgroup$\}$. Go to (3.2).
    (3.5*) Notify other devices that d is deleted. Find
        devices $\{d_{d1}, d_{d2}, .., d_{dt}\}$, t=1, $d_{di}$ is the parent
        of $d_{di+1}$, $d_{dt}$ is the parent of deleted device d,
        $d_{d1}$ is available, but $d_{d2}$ ,.., $d_{dt}$ are not
        available currently. Notify $d_{d1}$ that devices
        $d_{d2}$, ..,$d_{dt}$ are suspended and device d is
        deleted.
    (3.6) Invoke " Renewing('delete')".
    (3.7) The operation stops.
    Note*: A device is suspended if it is temporarily not
    available. When it is available again, it will be
    updated for the secure group change.
```

- **Operation 4: Renewing local subgroup key – RenewingSKey(type), where the parameter *type* is either 'add' or 'delete'.**

This operation is to generate a new local subgroup key k_{sub}' and distribute k_{sub}' to all members in the local subgroup. The old local subgroup key for the subgroup is k_{sub}. This operation can be done by the following algorithm:

```
    (4.1) lkd generates $k_{sub}$'.
    (4.2) Let C={ lkd } where lkd is the LKD of the
        subgroup.
    (4.3) If C is empty, go to (4.6). Otherwise, Find $x \in C$
        and let C=C-{x}.
    (4.4) Let $C=C \cup \{ y \mid (x, y, k_{x-y}) \in P$ and y is a group
        member$\}$. For each device y,
          ▪ If y is an old member and the operation is
            "add" : $x \rightarrow y: \{ k_{sub-i}'\} k_{sub-i}$.
          ▪ Otherwise, if y is not suspended:
                $x \rightarrow y: \{ k_{sub-i}'\} k_{pc-y}$.
    (4.5) Go to (4.3).
    (4.6) Invoke " Rekeying" .
    (4.7) The operation stops.
```

- **Operation 5: Renewing group key – Rekeying()**

This operation is to generate a new group key K_g and distribute K_g to all members in the entire secure group: This can be done as follows:

```
    (5.1) GDK gkd generates $K_g$.
    (5.2) For each subgroup with the LKD lkd and
          subgroup key $k_{sub}$:
          $gkd \rightarrow lkd: \{k_g\}k_{mlk}$.
          $lkd \rightarrow$ subgroup members: $\{k_g\}k_{sub}$.
    (5.3) The operation stops.
```

- **Operation 6: Dismissing a secure group**

This operation is to dismiss a secure group and delete Group Service of the secure group on all devices that are in the secure group. This operation is simple and hence needs not to be discussed.

- **Operation 7: Querying group key**

This operation is to query the most recent group key from the Group Service of a secure group. This operation is quite simple and hence needs not to be discussed.

7 An Example of a Distributed Application with Secure Group Communication

We use the following example to illustrate our approach: There is a company with two departments *dept1* and *dept2*, which have 5 employees with devices d_1, d_2, d_3, d_4, and d_5, and 4 employees with devices d_6, d_7, d_8, and d_9, respectively. The device d_0 is used for management work. The company plans to have a meeting in which a device can broadcast secure messages to all other devices attending the meeting. Employees of *dept1* will attend the meeting from the beginning and employees of *dept2* will attend the meeting from the second session. This example is to develop and execute a net meeting software for secure group communication for the company for this meeting.

According to our approach,

Step 1) Develop the distributed application.

The distributed application is developed using RCSM middleware. When generating the situation-aware object interface, the situation "application starts" is identified to trigger the " initiating group" operation, and the situation "application ends" is identified to trigger the "dismissing group" request.

When generating the code of the application, the method *sgc_send* and *sgc_receive* are used for a device to broadcast secure messages to all other devices.

Step 2) Identify secure group members.

The secure group will have the following devices: { d_0, d_1, .., d_9}. Device d_0 will act as the GKD and device d_1 will act as the LKD.

After generating required secret keys and adding proper authentication relations, the secure group has the following authentication relations:

$(d_0, d_1, k_{0\text{-}1})$ $(d_1, d_2, k_{1\text{-}2})$ $(d_2, d_3, k_{2\text{-}3})$
$(d_2, d_4, k_{2\text{-}4})$ $(d_3, d_5, k_{3\text{-}5})$ $(d_1, d_6, k_{1\text{-}6})$
$(d_6, d_7, k_{6\text{-}7})$ $(d_6, d_8, k_{6\text{-}8})$ $(d_6, d_9, k_{6\text{-}9})$

Step 3) Secure group management.

When the distributed application starts to run, the " Initiating secure group" operation is invoked by the situation "application starts" . The following operations will be executed in the following sequence:

a) " Initiating secure group" . The management layer key is k_{mlk}:

1) $d_0 \rightarrow d_1$: $\{k_{mlk}\}\ k_{0\text{-}1}$

b) "Adding members":

1) $d_1 \rightarrow d_1$: $k_{m\text{-}2}$ (no encryption)
2) $d_1 \rightarrow d_2$: $\{k_{m\text{-}2}, k_{pc\text{-}2}\}k_{1\text{-}2}$
3) $d_2 \rightarrow d_1$: $\{k_{m\text{-}3}\}k_{m\text{-}2}$
4) $d_2 \rightarrow d_3$: $\{k_{m\text{-}3}, k_{pc\text{-}3}\}k_{2\text{-}3}$
5) $d_2 \rightarrow d_1$: $\{k_{m\text{-}4}\}k_{m\text{-}2}$
6) $d_2 \rightarrow d_4$: $\{k_{m\text{-}4}, k_{pc\text{-}4}\}k_{2\text{-}4}$
7) $d_3 \rightarrow d_1$: $\{k_{m\text{-}5}\}k_{m\text{-}3}$
8) $d_3 \rightarrow d_5$: $\{k_{m\text{-}5}, k_{pc\text{-}5}\}k_{3\text{-}5}$

c) " Renewing local subgroup key" . The new subgroup key is k_{sub}:

1) $d_1 \rightarrow d_2$: $\{ k_{sub}\}k_{m\text{-}2}$
2) $d_2 \rightarrow d_3$: $\{k_{sub}\}k_{pc\text{-}3}$
3) $d_2 \rightarrow d_4$: $\{k_{sub}\}k_{pc\text{-}4}$
4) $d_3 \rightarrow d_5$: $\{k_{sub}\}k_{pc\text{-}5}$

d) " Renewing group key" . The new group key is k_g.

1) $d_0 \rightarrow d_1$: $\{ k_g\}k_{mlk}$
2) $d_1 \rightarrow d_2, d_3, d_4, d_5$: $\{k_g\}k_{sub}$

When the second meeting session starts, devices d_6, d_7, d_8, and d_9 receive secure messages and the "adding members" operation is invoked. The following operations will be executed in the following sequence:

a) "Adding members":

1) $d_1 \rightarrow d_1$: $k_{m\text{-}6}$ (no encryption)
2) $d_1 \rightarrow d_6$: $\{k_{m\text{-}6}, k_{pc\text{-}6}\}k_{1\text{-}6}$
3) $d_6 \rightarrow d_1$: $\{k_{m\text{-}7}, k_{pc\text{-}7}\}k_{m\text{-}6}$
4) $d_6 \rightarrow d_7$: $\{k_{m\text{-}7}, k_{pc\text{-}7}\}k_{6\text{-}7}$
5) $d_6 \rightarrow d_1$: $\{k_{m\text{-}8}, k_{pc\text{-}8}\}k_{m\text{-}6}$
6) $d_6 \rightarrow d_8$: $\{k_{m\text{-}8}, k_{pc\text{-}8}\}k_{6\text{-}8}$
7) $d_6 \rightarrow d_1$: $\{k_{m\text{-}9}, k_{pc\text{-}9}\}k_{m\text{-}6}$
8) $d_6 \rightarrow d_9$: $\{k_{m\text{-}9}, k_{pc\text{-}9}\}k_{6\text{-}9}$

b) " Renewing local subgroup key" . The new local subgroup key is k_{sub}':

1) $d_1 \rightarrow d_2, d_3, d_4, d_5$: $\{k_{sub\text{-}1}'\}\ k_{sub\text{-}1}$
2) $d_1 \rightarrow d_6$: $\{k_{sub\text{-}1}'\}\ k_{m\text{-}6}$
3) $d_6 \rightarrow d_7$: $\{ k_{sub\text{-}1}'\}k_{pc\text{-}7}$
4) $d_6 \rightarrow d_8$: $\{ k_{sub\text{-}1}'\}k_{pc\text{-}8}$
5) $d_6 \rightarrow d_9$: $\{ k_{sub\text{-}1}'\}k_{pc\text{-}9}$

c) " Renewing group key" . The new group key is k_g':

1) $d_0 \rightarrow d_1$: $\{k_g'\}\ k_{mlk}$
2) $d_1 \rightarrow d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9$: $\{ k_g'\}\ k_{sub\text{-}1}'$

When the application ends, the operation "dismissing group" is invoked by the situation "application ends" and the secure group is dismissed.

8 Discussions

In this paper, we have presented a middleware service for secure group communication for distributed applications in mobile ad hoc networks. Our approach has flexible secure group management, and supports development of distributed applications with secure group communication in mobile ad hoc networks. We have developed a simulation system to evaluate the performance of our approach. The simulation results show that our approach is more suitable than other approaches [7, 8] in mobile ad hoc networks because it has fewer operations (encryption/decryption and communication) in a single device: 5 for adding a device and d+2 for deleting a device, where d is the degree of the device tree of the secure group. We are currently extending our approach to secure group communication covering different types of networks, such as mobile ad hoc networks and infrastructure-based networks, and combining the secure group communication service with other middleware security services to address general security requirements, such as secure group access control.

Acknowledgment

This research is supported in part by National Science Foundation under grant number ANI-0123980.

References

[1] J. McHugh, J. B. Michael, " Secure group management in large distributed systems: what is a group and what does it do?", *Proc. 1999 Workshop on New Security Paradigm,* Sept. 1999, pp. 80-85

[2] S. S. Yau, et al, " Reconfigurable context-sensitive middleware for pervasive computing", *IEEE Pervasive Computing,* July-Sept., 2002, Vol. 1, pp. 33–40.

[3] S. S. Yau, Y. Wang, and F. Karim, " Development of Situation-Aware Application Software for Ubiquitous Computing Environments", *Proc. COMPSAC 2002*, Aug 26-29, 2002, pp. 233-238

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

[5] R. H. Deng, L. Gong, A. A. Lazar, and W. Wang, "Authenticated key distribution and secure broadcast using no conventional encryption: A unified approach based on block codes," *Proc. IEEE Globecom'95*, Nov. 1995.

[6] C. Tseng, et al, "A Constant Size Rekeying Message Framework for Secure Multicasting", *Proc. LCN'01*, at www.computer.org

[7] S. Mittra, " Iolus: A Framework for Scalable Secure Multicasting", *Proc. ACM SIGCOMM '97 conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 277-288

[8] C. K. Wong, and S. S. Lam, " Secure Group Communications Using Key Graphs", *IEEE/ACM Trans on Networking*, Vol. 8, Feb. 2000, pp. 16-30

[9] B. DeCleene, et al, " Secure Group Communications for Wireless Networks", *MILCOM 2001*, at www.tascnets.com/newtascnets/Publications/Documents/Main2.html#SecureGroup