Scalable key management for secure multicast communication in the mobile environment

Jiannong Cao\textsuperscript{a,\ast}, Lin Liao\textsuperscript{a,b}, Guojun Wang\textsuperscript{a,b}

\textsuperscript{a} Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
\textsuperscript{b} School of Information Science and Engineering, The Central South University, Changsha, 410083, PR China

Received 1 May 2005; received in revised form 26 November 2005; accepted 30 November 2005
Available online 6 January 2006

Abstract

The proliferation of the Internet computing and mobile computing technologies gives rise to the growth of mobile Internet applications. There are increasing demands on secure multicast services in the mobile Internet environment. An important issue in secure group communication is key management, which is concerned with distributing and updating the keys for encrypting the multicast messages in a group of mobile users. The challenges in designing secure and scalable key management protocols are dynamic updates of the key caused by frequent moves, joining and leaving of group members and the large size of a group for mobile Internet applications. In this paper, we propose a scalable and hierarchical key management (SHKM) protocol in the mobile Internet. In order to address the scalability issue, SHKM divides the group of users into different subgroups, where each subgroup uses its own key. Different from existing decentralized schemes, the subgroups in the SHKM protocol are organized into a hierarchical structure with different priorities. Each pair of parent–child subgroups is given a related factor based on their subgroup keys. The trusted third-party authority is responsible for computing the parameters. Based on these and some public parameters, users belonging to a higher-priority subgroup have the right and are capable to deduce the keys of lower-priority subgroups but the reverse operation is not allowed. Because the subgroup itself can perform the derivation, the number of re-keying messages for updating the global key management system across the subgroup boundaries can be significantly reduced. We compare the proposed protocol with some existing protocols, and conclude that the number of re-keying messages in our proposed protocol is the least among all these protocols.

\ast Corresponding author. Tel.: +852 27667275; fax: +852 27740842.
E-mail addresses: csjcao@comp.polyu.edu.hk (J. Cao), cslliao@comp.polyu.edu.hk (L. Liao),
csgjwang@comp.polyu.edu.hk, csgjwang@mail.csu.edu.cn (G. Wang).

1574-1192/\$ - see front matter \textcopyright 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2005.11.003
1. Introduction

Multicast provides the support for the efficient delivery of data from one or multiple sources to a group of receivers. In the Internet, IP multicast saves a great deal of bandwidth by sending the source traffic on a multicast tree that spans all the group members. There are many multicast applications including video conferencing, interactive group games and video on demand (VOD). Recently, the proliferation of the Internet technology, wireless communication technologies, and mobile computing devices gives rise to the growth of mobile Internet applications. Consequently, how to design high-performance multicast communication protocols for the mobile Internet environment becomes a hot research topic. In this paper, we study the security issues in group communication and propose a scalable group key management protocol for the mobile Internet environment. Among the others, group key management is an important issue, which is concerned with distributing and updating the keys for encrypting the multicast messages in a group of mobile devices. The challenges in designing a secure and scalable key management protocol for mobile Internet are dynamic updates of the key caused by the frequent moving, joining and leaving of group members, and the large size of a group for mobile Internet applications.

Most multicast applications require an access control mechanism to guarantee that only the valid users can access the group communication content. In the context of secure multicast communication, cryptographic methodology is widely employed to ensure secure transmissions. The fundamental principle in secure group communication is to allow authorized entities to obtain the valid group key and participate in the communication using the key. The basic idea is to generate a cryptographic group key shared by all members within the group to encrypt the traffic data so that the users who obtain the correct group key have the capability to recover the original messages from the ciphered ones. Given that any node of the network can join and leave the group freely and also an intruder can send malicious messages to the group without being a valid member, it becomes critical and necessary to have some mechanisms to distribute and maintain the group key for dynamically changing group members.

Since the multicast group is open and dynamic, whenever a member joins or leaves, the group key should be renewed and the new group key should be securely distributed to current legitimate users. More specifically, when a host requires joining a group, the group key needs to be updated in order to ensure the backward confidentiality; otherwise, the previous communication traffic will be disclosed to the newcomer. Similarly, when a member leaves a group, the forward confidentiality is guaranteed by re-keying the remaining group so that the departed member is unable to eavesdrop on the future messages. The re-keying procedures include regeneration and distribution of the new group key. The so-called 1-affects-n phenomenon is a challenging issue in designing group key management protocols. If the group size keeps increasing, such a phenomenon will significantly degrade the system performance. Furthermore, in mobile Internet, the high
mobility of mobile hosts and limited bandwidth of wireless connections intensify the complexity of security in multicast group communication. Especially when the number of group members becomes larger and the coverage area becomes wider, the key distribution and re-keying process will accordingly incur a larger overhead.

Several group key management approaches are proposed to minimize the overall overhead in a scalable and secure manner. However, all the approaches have some shortcomings in handling group dynamism. The centralized schemes suffer from the 1-affects-\(n\) phenomenon especially if the group membership is highly dynamic. The root in the centralized scheme is easily saturated and the whole system can be attacked due to the single point of failure problem. On the other hand, the decentralized schemes suffer a lot from the decryption and re-encryption operation before forwarding the messages to the local users, especially if the group membership is relatively static. Taking the two concerns into consideration, in this paper, we propose a new scalable and hierarchical key management (SHKM) protocol for secure multicast communication in mobile Internet. SHKM addresses the scalability issue by separating the global group of mobile users into different subgroups, where each subgroup uses its own key, and takes the advantages of both the decentralized cluster techniques to reduce the effects of the 1-affects-\(n\) phenomenon and the hierarchical key assignment techniques to reduce the re-keying cost. In SHKM, cluster heads are assigned higher priorities than the local users and the priorities of the cluster heads are defined by the levels of positions where they join. According to the priorities, all clusters including attached local users are arranged into a tree hierarchy. Users belonging to a higher-priority subgroup have the right and are capable to deduce the key of a lower-priority subgroup but the reverse operation is not allowed. Because the subgroup itself can perform the derivation, the re-keying messages for updating the global key management system across the group boundaries can be significantly reduced.

We compare our protocol with some existing protocols. SHKM requires the least number of re-keying messages among these protocols, especially when dynamic changes occur frequently.

The remainder of the paper is organized as follows. In Section 2, we describe related works and classify and summarize existing solutions. Section 3 describes our system model. The design of the proposed scalable and hierarchical key management (SHKM) protocol is presented in Section 4. In Section 5, we give an analysis of the protocol for dynamic scenarios. Finally, Section 6 compares our protocol with some existing protocols and concludes the paper.

2. Related works

The problems of access control to the shared resources and key management for multiple users under frequent mobility and high failure probability in mobile Internet are much more complicated than those in traditional Internet. Many solutions have been proposed in the literature to address the problems with concerns of scalability.

We can classify the solutions for group key management into four categories: the simplest scheme, centralized scheme, decentralized scheme, and hierarchical scheme. In the simplest scheme, all the group members share a secret group key called traffic
encryption key (TEK) assigned by the group manager (GM). The GM encrypts the new TEK by each user’s individual key one by one, and then constructs such a re-keying message including all these encrypted items. The re-keying operation is performed by just multicasting the re-keying message and each user only requires one time decryption upon receiving it. Although the scheme has the advantage of simplicity, it suffers a lot from the scalability problem with long latency and GM saturation. In addition, frequent user changes will cause the system bottleneck problem.

In the centralized scheme, there is only one central group manager that performs secure access control without relying on any auxiliary entity. The group manager is the single entity who is responsible of computing and distributing the shared TEK. It distributes a separate individual key for every user in order to build a secret unicast channel to transmit the shared TEK. Typical protocols using the centralized scheme are logical key hierarchy (LKH) [1] and key graph [2]. In the centralized scheme, the group manager located at the root organizes and maintains all the keying materials in a balanced tree whose leaves are the group members’ individual keys. The internal nodes of the tree correspond to auxiliary keys called key encryption keys (KEKs) and the key of the root is the shared TEK. Each user stores all the keys along the path from itself to the root. Different from the simplest scheme, the group manager uses the hierarchy of KEKs as the distribution carrier. With a group of \( n \) users, the communication complexity is \( O(\log n) \) and the storage complexity of each group member is \( O(\log n) \), while for the simplest scheme the communication complexity is \( O(n) \).

The tree hierarchy has the inherent disadvantage of being vulnerable to the single point of failure problem. Many works have been proposed to improve the scheme in different aspects. In [3], Balenson proposed the technique of one-way functions tree (OFT), which reduces the number of required re-keying messages to a half. However, since each internal node key is a function of its children, the costs saved on sending re-keying messages are actually replaced by the increasing computation cost. In [4], Roberto proposed a new protocol called LKH++, which improves the performance by exploiting both the properties of one-way hash functions and the information that the users already share in the LKH model. Later, some schemes based on key trees are proposed, such as adaptive FEC and periodic batch re-keying to deal with frequent changes [5], hierarchical \( a \)-ary tree with clustering [6], and an \( e \)-key policy called exposure-oriented re-keying which accounts for the cost of no re-keying [7]. Since using a single TEK suffers from the \( 1 \)-affects-\( n \) phenomenon, all the schemes attempt to make the GM to minimize the storage requirements such as key encryptions keys (KEKs) and size of re-keying message to achieve the scalability. Extending to large-scale groups, such a centralized key distribution center (KDC) turns out to be somewhat burdensome and the single server turns out to be the main point of attack for intruders.

In the decentralized scheme, a given group is partitioned into subgroups to minimize the problem of concentrating on a single entity. Each subgroup is assigned a subgroup manager, which establishes the encrypting key for the subgroup. Membership changes in a subgroup can be treated locally and the subgroup manager multicasts the new secret subgroup key to the residual local members. According to whether there is the global TEK or not, this scheme can be classified into two sub-classes: weakly decentralized and strongly decentralized. A typical strongly decentralized scheme is Iolus [8], a novel
framework for scalable and secure multicast. Scalability is achieved by having each subgroup to operate relatively independently. A group security agent (GSA) is designed to manage each subgroup. The GSAs are grouped in a top-level structure controlled by the group security controller (GSC). Iolus uses independent keys for each subgroup and the absence of a general group key allows membership changes in a subgroup to be handled locally. The joining or leaving of a member in certain subgroups will not directly trigger the re-keying operation of other subgroups. Therefore, it can be easily extended for large-scale dynamic group communication. However, decryption and re-encryption of data packets are needed while transmitted among different subgroups. Another typical weakly decentralized scheme is intra-domain group key management (IDGKM) [9]. In this protocol, the subgroups called “leaf” regions in the architecture are connected together through the “trunk” region (backbone). There exists an initiator key distributor (IKD) that holds a copy of the multicast key $K_m$ and a copy of all the subgroup keys corresponding to the $N$ subgroups $AS_1, AS_2 \ldots AS_N$. Thereafter, the IKD is actually the organizer of all key distributors (KDs) as well as the initiator of the whole multicast instance. Due to the existence of a global multicast key, the re-keying of any $AS$ due to some members’ changes will lead to the update of the global multicast key, as well as the delivery of a new multicast key to other $AS$s.

Other strategies have been proposed to improve the overall performance of the system. Setia et al. proposed the Kronos approach, where membership changes are batched for a period of time and a single re-keying operation is made for the batch of membership changes [10]. In this protocol, the overall area key distributions (AKDs) are synchronized using a specific protocol. Hydra [11] is also based on decentralized architecture. However, some decentralized protocols bring new problems such as synchronization and conflicting resolution.

In the hierarchical scheme, the roles of the users are not equal but organized into a hierarchy with different priorities. Cryptographic access control is important to protect the shared resources in a hierarchical system. Hence, special key assignment is required for users of various priorities in a group so that different users can access different and limited resources. Akl and Taylor first proposed the hierarchical cryptographic key assignment protocol among a group of users [12] for assigning the cryptographic key to a set of partially ordered classes so that a cryptographic key of a high-privileged class can be used to derive the cryptographic key of lower-privileged classes. Many works have concentrated on achieving more secure performance and reducing time complexity of servers in order to solve the scalability problem [13–17]. The dynamic key management scheme for access control (DKSAC) protocol [13] uses a related parameter of partially ordered classes for the hierarchical key management in a user hierarchy. A time-bound cryptographic key assignment protocol [14] is proposed in which the cryptographic keys of a class can change during different time periods, and the key derivation is constrained not only by the class relation but also by the time period. To improve Akl and Taylor’s protocol, a protocol is proposed for cryptographic key assignment based on Newton interpolation method and a predefined one-way function [15], and a cipher system based upon the Diophantine equations is proposed which enables the keys to be easily generated [16]. A secret sharing scheme employs the concept of admission tickets to delegate the access right from ancestors to their descendants [17].
Most works on cryptographic access control focus on achieving security in a user hierarchy. In multicast communication, the protection of messages is necessary; therefore, access control can be used in multicast key management protocols if we allocate some delegated users with privileged priorities. In view of the fact that in multicast communication systems, the multicast data and group key messages should be disseminated from the multicast source to all the receivers in a secure way, similarly, all the forwarding nodes can also be organized in a hierarchical structure.

3. System model and assumptions

In this section, we present a generic architecture for secure multicast communications in mobile Internet. In mobile Internet, more issues need to be considered which are different from those in traditional Internet [18]. First, mobile hosts such as laptop computers, PDAs, and mobile phones have severe resource constraints in terms of energy and processing capabilities. Second, wireless communication involves high error rate and limited bandwidth problems. Third, the number of mobile hosts may be very huge and the mobile hosts may move very frequently. Considering these concerns, we make use of the clustering techniques to handle the mobility of group users. Under this circumstance, the events that one mobile user moves from one location to another will be processed like moving from one subgroup to another. Especially when the mobile users move around within the same subgroup, the cost of key management that influences the other subgroup area is very small.

We aim at minimizing the effect of the 1-affects-n phenomenon, in terms of the number of re-keying messages and the number of encryptions and decryptions. There are two kinds of messages transmitted during a multicast instance, namely, multicast data massages and multicast control messages. Messages for key management belong to the control message class.

Fig. 1 shows the system architecture for group key management. Node S denotes the multicast source node where the multicast instance starts, and the nodes in the rectangle denote the network entities involved in the multicast communication. These nodes are implemented through network entities typically found in the wired Internet, such as routers, switches and servers, together with their corresponding network protocols. They are also called forwarding nodes since they constitute the overlay network. The dotted lines connect the node S to the forwarding nodes, with the black point indicating the destination. Each forwarding node F denotes a subgroup as shown in the dotted circle.

The forwarding nodes can be assigned with different priorities according to the precedence relation in the multicast network as shown in Fig. 1. The node with the prior position naturally has the higher priority value than those positioned behind it. For example, F1 has a higher priority than F2 and F6, represented by F1 ≻ F2, F1 ≻ F6; F3 has a higher priority than F4, represented by F3 ≻ F4. In order to simplify the relationship, F1 is called the direct predecessor of F2 and F6, and F4 is called the direct successor of F3. If F4 has its successor F7 (if exists), then F3 is called the ancestor of F7. As to F2 and F6, we cannot decide their prior relationship between siblings. Therefore, only the direct relationship between forwarding nodes is defined, i.e. the parent–child relationship. Similarly, the subgroups related to each forwarding node become partially ordered groups.
In our system architecture, the backbone nodes of the network connect all user subgroups. As shown in Fig. 1, the forwarding node in the multicast network is actually the mirror image of the subgroup manager (SGM) in each subgroup. The SGM is the manager of the subgroup, which is responsible for the key establishment, key re-generation, key transmission and membership change control of its group members. Since the forwarding nodes are hierarchical, the whole group can be treated as many hierarchical subgroups with different priorities.

The following terms are defined with the system architecture.

- **Hierarchy**. The term hierarchy here has a different meaning from a traditional hierarchical system. Here, it is defined according to the precedence relation between the forwarding nodes in the backbone network. Along one path, the node that gets messages earlier will be assigned the higher priority than its successors. The hierarchical relation is logical.

- **Centralized authority (CA)**. CA is widely used in secure systems. It is a trusted third-party, which every user can trust. CA authenticates all its group members by firstly checking, registering personal documents, and issuing certificates. In a very large group, one single CA seems unlikely to be capable to handle all the things. From the point of fault tolerance, we propose to have a series of redundant CAs rather than a single CA as shown in Fig. 1 in our protocol.

- **Priority ordering >**. Assume there are \( m \) subgroups, namely \( S_1, S_2, \ldots, S_m \). From the above hierarchical criterion, the priority comparison can be represented as \( S_j > S_l > S_i \), while \( > \) denotes that its left-side subgroup has the higher privilege than its right-side subgroup. Obviously the subgroup with a position closer to the node \( S \) has the higher privilege than that with a position further in the whole hierarchy. A node has the right
to access the keys of all the successors, whose positions are after the node $S$ along one path from the node $S$.

4. Secure group key management protocol

In this section, we present the scalable and hierarchical key management (SHKM) protocol. SHKM consists of three algorithms: one for key generation, one for key derivation, and one for key modification.

4.1. Key generation

Strictly speaking, only the users including mobile hosts classified into different subgroups may join a group session because they stand for the true users of the group communication services. The forwarding nodes (FNs) and node $S$ in the network trunk are not the users of the group communication services. However, we still call the FNs users since they logically function as special users to assist the true users to get group communication services.

The main idea of our approach is that each subgroup can randomly choose its own traffic encryption key (TEK) and report it to the trusted CA, and after that CA will compute a parameter for any two predecessor–successor subgroups. The users belonging to higher-priority subgroups will have the right to deduce the TEK of lower-priority subgroups but the reverse operation is not allowed. Due to the fact that the derivation can be performed by the SGM, the number of re-keying messages which need to be transmitted can be significantly reduced.

After authenticating all the users, CA is responsible for computing some parameters, such as changing residue and related factor, used for key derivation. Changing residue is a mapping value of the TEK of one subgroup computed by CA using some secret parameters only known by CA. For any two predecessor–successor subgroups, related factor is computed based on their TEKs and some public parameters including changing residue. Related factor converts the abstract and secret relationship of two groups into a numerical and public value. Before CA computes the required parameter $r$, all subgroups need to send the chosen TEK (for example $k_i$ for subgroup $S_i$) secretly to CA. A cryptographic function $f_r(x, y)$ needs to be established in the cryptosystem so that $k_i$ can be derived in such an equation, where $x$ and $y$ denote two related TEKs. Because of some secret parameters in the function held by CA, CA is the only one which can compute the related factors. Before CA performs the computation, it needs initiation operations, such as generating a pair of keys (PK, SK) according to the RSA algorithm, obtaining all valid direct candidate predecessor–successor relationships, etc.

The initiation operations performed by the CA are carried out in the following steps:

**Step 1** Chooses two secret large prime numbers, $p$ and $q$ respectively.

**Step 2** Computes a secret number $n = p \times q$ as well as a function result $\varphi(n) = (p - 1)(q - 1)$. $\varphi(n)$ is the Euler function of $n$.

**Step 3** Chooses an integer $e$, which satisfies $1 < e < \varphi(n)$ and $\gcd(\varphi(n), e) = 1$, and $e$ is an integer with no common factor with $\varphi(n)$ and less than $\varphi(n)$.
Step 4 Computes an integer $d$, which satisfies $d \times e \equiv 1 \mod \varphi(n)$.

Step 5 Computes the public key $(PK) = \{e, n\}$, and the secret key $(SK) = \{d, n\}$ according to the famous RSA algorithm. Using the public key mechanism, the encryption and decryption functions are separate for security. PK is publicly known to the whole users and SK is known only by itself for decrypting secret messages.

After CA gets the pair of keys, it starts to configure the system by computing the required parameters. Let $P$ be a large prime number, and $Z$ be a primitive element of definite domain $GF(P)$. During the key generation period, each SGM chooses TEK for its subgroup, which is shared by all the members within its subgroup. Suppose the SGM for group $S_i$ randomly chooses one TEK $k_i$ for itself, and then conveys the chosen key to CA, encrypted by the PK of CA. Upon receiving the TEK of subgroup $S_i$, CA will compute some parameters for $S_i$ in order for $S_i$ to perform the key derivation operation. Only if the subgroup is authenticated and computed by CA, can the subgroup perform key derivation and be connected to other subgroups.

The key generation protocol is described as follows.

Step 1 Each SGM of subgroup $S_i$ chooses the TEK $k_i$ independently with, for example, 512 bits.

Step 2 Each subgroup encrypts the TEK by PK of CA and sends message $E_{PK}^k (k_i)$ to CA. Here $E(\ )$ means some kind of encryption function. CA uses the issued SK to decrypt each of them and authenticate them by their signature information.

Step 3 CA recovers the encrypted message $E_{PK}^k (k_i)$ using the SK to obtain secret $k_i$.

Step 4 For each $k_i$, CA computes a parameter called changing residue $e_i$ using the Eq. (1). CA is responsible for retaining a pair of parameters changing residue $e_i$ and $k_i$ for each subgroup. Because $n$ is known only by CA, $e_i$ is only held by CA. From the Rabin cryptography and the work in [19], the difficulty of finding the solutions of the congruence $k_i^2 \equiv e_i \mod n$ is equivalent to factorizing a product of two large prime numbers

$$e_i = k_i^2 \mod n. \quad (1)$$

Step 5 As to the priority relationship $S_j > S_i$ of two subgroups $S_j$ and $S_i$, CA computes the related factor $\gamma_{ji}$ for them using the following Eq. (2).

$$\gamma_{ji} = h (Z^k_j \oplus e_i \mod P) \oplus k_i. \quad (2)$$

In Eq. (2), $h(\ )$ represents a one-way hash function, $\oplus$ indicates the bitwise XOR operation. Since CA is aware of TEK of all local groups as well as the required primitive parameters $P$, $Z$ and $n$, only CA can compute the related factors $\gamma$ and the changing residues $e_i$. After the initial procedures, the value of $P$, $Z$, and $h(\ )$ are public to all subgroups. However, the value of $n$ is also a secret number. It is worth noting that not each pair of subgroups will give a related factor but the direct predecessor and successor subgroups are allowed to have one. Because the sibling subgroups cannot access each other, CA will not compute such a related factor for them and only legitimate related factors can exist in CA. For the reason that we configure a series of CAs, we believe that such a computation for each subgroup rather than each group will not be a great burden for CA.
4.2. Key derivation

After key generation, CA obtains the knowledge about the whole system and the quantified relationship between subgroups. By performing the access control in key management, users in a higher-priority subgroup have the right to access the TEK of a lower-priority group, but the reverse access is not allowed. For each relation $S_j \succ S_i$, users of the subgroup $S_j$ can deduce the TEK of subgroup $S_i$ by functioning $f_r(x, y)$ in a secure and efficient way. Here the SGM undertake the derivation work on behalf of subgroup $S_j$. The derivation procedure works as follows.

**Step 1** $S_j$ requests for necessary parameters from CA while requiring the TEK of $S_i$;

**Step 2** If CA detects that the request is out of its power range or the requester is not a group member, it will terminate the request;

**Step 3** If all the requirements are OK, CA sends a message including $\gamma_{ji}$ and $e_i$ to $S_j$ encrypted by $k_j$;

**Step 4** On receiving the parameters, $S_j$ starts to derive the $k_i$ according to Eq. (3).

$$k_i = h \left( Z^{k_j \oplus e_i} \mod P \right) \oplus \gamma_{ji}.$$  \hspace{1cm} (3)

The derived lower-priority subgroup TEK is resulted by a hash function value bitwise XOR the related factor. As the common parameters are public and related factor and changing residue are obtained from CA legitimately, the TEK of the lower-priority subgroup can be derived in this way. It is the setting of the intermediate parameter that makes the TEK of each subgroup transparent to each other.

4.3. Key modification

Since the TEK of each subgroup is chosen by the group itself, it is likely that the SGM would like to replace the key by another secure one when some scenarios occur such as the joining of a new group member or the departure of some group members. Assuming that $S_l$ has changed its TEK, all related parameters computed on the basis of it need to be recomputed. In terms of related factor, the neighbour subgroups of a subgroup pointing both from and towards to it will be involved by the modification. Similarly, the changing residue $e_i$ needs to be renewed by CA. Let us imagine that a group $S_l$ needs to change its TEK from $k_l$ to $k^*_l$; $S_j$ denotes the direct predecessor that has the power to deduce the group key of $S_l$; $S_i$ denotes the direct successor of $S_l$. Referring to the relationship $S_j \succ S_l \succ S_i$, CA needs to re-compute the following three required parameters by Eqs. (4)–(6), and storing the information renewed.

$$e^*_i = \left( k^*_l \right)^2 \mod n$$ \hspace{1cm} (4)

$$\gamma^*_{ji} = h \left( Z^{k_j \oplus e^*_i} \mod P \right) \oplus k^*_l$$ \hspace{1cm} (5)

$$\gamma^*_{li} = h \left( Z^{k^*_l \oplus e_i} \mod P \right) \oplus k_i.$$ \hspace{1cm} (6)

From the above analysis, we can see that when a subgroup changes its TEK, the predecessor and successor of the subgroup are not affected, which contributes much to the
scalability of multicast communication. What is needed to do is to update the quantified relationship by CA. Although the re-keying cost caused to subgroups is little, the changing subgroup has to distribute the new TEK to its members in a scalable way.

5. Protocol analysis

In the previous section, we described the normal operation of a static group. However, for multicast group communication in mobile Internet, there are many dynamic changes such as the changes to the membership and network topology, which bring in more complexity to the key management.

In our protocol, we consider three different types of mobility:

- **join/leave** — User’s joining and leaving the multicast group.
- **transfer/enter** — Member’s roaming from one subgroup to another without breaking the multicast service.
- **insertion/deletion** — The situation where all group members move around or the forwarding node in the multicast network has to move because of network split or merge.

The following subsections explain how to perform secure key assignment and management in the case of the aforementioned three types of mobility.

5.1. Adding a new subgroup

Adding a new subgroup means that a new forwarding node is connected to the multicast network. The neighbourhood will be updated by exchanging some information such as the routing table. In order to insert a new subgroup \( S_i \) into an existing multicast group, CA is required to compute some parameters such as \( e_i \) and related factor \( \gamma \), which are corresponding to the new group. For example, after obtaining the new neighborhood, CA needs to compute changing residue \( e_i \) for this subgroup and related factor \( \gamma \) belonging to \( S_i \) and its direct neighbours. Therefore, after the subgroup is added, the other groups are not affected. Compared with some other schemes like Key Graph, there is even no one single re-keying message and no re-generation of a new TEK for the whole group. The other subgroups do not need to re-generate their own keys at all.

5.2. Deletion of a subgroup

In the case of network breaking and reconfiguration, one subgroup is likely to be deleted from the current multicast service. Similarly, by using the hierarchical access mechanism, leaving of one subgroup will not affect the higher-priority subgroups because the deleted subgroup is incapable of deriving their keys. However, once the departed subgroup \( S_j \) has derived the TEK of its successor subgroup \( S_i \), and \( S_i \) didn’t modify \( k_i \) from that time on, \( k_i \) is exposed to the departed group. Since only the legitimate group members can request for \( \gamma \) from CA and only \( \gamma \) related with the requester can be responded, the groups with lower privilege are supposed to change group keys for higher security. Assuming \( S_i (S_j \succ S_i \succ S_l) \) is about to leave, the leaving procedure is described as follows.
Step 1 $S_l \rightarrow CA$: “Request for leave”;
Step 2 CA collects the information of its lower-priority subgroups;
Step 3 CA $\rightarrow S_l$: “You are deleted from the group”;
Step 4 CA $\rightarrow S_i$: “Choose a new key”;
Step 5 $S_i$ reports to CA its new TEK securely, $E_{PK}(k_i^*) \rightarrow CA$;
Step 6 CA operates $D_{SK}(E_{PK}(k_i^*)) = k_i^*$;
Step 7 CA computes new $e_i^*$ for the lower-priority subgroup $S_i$;
Step 8 CA computes $\gamma_{ji}^*$, $\gamma_{jl}^*$, $\gamma_{li}^*$ from the database stored by CA.

In fact, deleting a subgroup from the group communication system is transparent to the higher-priority subgroup. So there are no re-keying messages to the predecessor of the deleted one. As to a lower-priority subgroup, what it needs to do is to re-generate a new TEK for itself. Compared with the centralized protocols, the re-keying cost is little in the subgroups, because part of the re-keying cost is transferred to the role of CA.

### 5.3. Leaving of subgroup members

If group $S_l (S_j \succ S_l \succ S_i)$ has $f$ members ($m_1, m_2, \ldots, m_f$) and $h$ ($1 \leq h < f$) members ($m_i, m_{i+1}, \ldots, m_{i+h-1}$) have to leave, in order to guarantee that departed members have no right to access the future communications, the TEK of $S_l$ should be renewed. The following steps explain the protocol when some members leave the group.

Step 1 ($m_i, m_{i+1}, \ldots, m_{i+h-1}$) request for leave to the SGM of $S_l$;
Step 2 $S_l : k_l \rightarrow k_l^*$;
Step 3 CA collects the information of all lower-priorities groups;
Step 4 CA $\rightarrow S_l$: “($m_i, m_{i+1}, \ldots, m_{i+h-1}$) have left from the group”;
Step 5 $S_l : k_l \rightarrow k_l^*$;
Step 6 $S_l : E_{PK}(k_l^*) \rightarrow CA$;
Step 7 CA operates $D_{SK}(E_{PK}(k_l^*)) = k_l^*$;
Step 8 CA $\rightarrow S_l$: “Choose a new TEK”;
Step 9 $S_l : E_{PK}(k_l^*) \rightarrow CA$;
Step 10 CA operates $D_{SK}(E_{PK}(k_l^*)) = k_l^*$;
Step 11 The SGM of $S_l$ distributes the new TEK its local users;
Step 12 CA computes new $e_i^*$ for subgroup $S_i$;
Step 13 CA computes $\gamma_{ji}^*$, $\gamma_{jl}^*$, $\gamma_{li}^*$.

### 5.4. Joining of a new subgroup member

This scenario should be the simplest one among all the kinds of mobility. The TEK of the subgroup certainly needs to be updated, but no more updating of other subgroups will occur. The SGM of the subgroup should report the key change to CA and CA will automatically compute the related parameters.
5.5. Migration of member between subgroups

Mobility complicates key management by allowing members to not only leave or join a session but also transfer between network subgroups while remaining in the session. It’s obvious that it cannot be regarded as first leaving and then joining because the re-keying of the whole group TEK is not necessary. Mobility impacts performance only when members move across subgroups, where re-keying messages must cross network boundaries, resulting in additional performance degradation.

There are three approaches used by algorithms defining actions to be performed when a member neither joins nor leaves the group, but just transfers from one subgroup to another. It is reported that the approach of first entry delayed re-keying + periodic (FEDRP) has a low re-keying rate and message rate \[20\]. We adopt this scheme to deal in our protocol. In FEDRP, when a member transfers from SG\(_i\) (SubGroup \(i\)) to SG\(_j\), it sends two signalling messages to SGM\(_i\) and SGM\(_j\) respectively. SGM\(_i\) doesn’t perform re-keying processes to its local users right now. Thus, the member may accumulate the TEK of the subgroup where it moves out as it enters different subgroups. If the entering member has previously visited SG\(_j\), no re-keying occurs for SG\(_j\). If not, the member is unicast the TEK of SG\(_j\) as needed. If the member is entering into SG\(_j\) for the first time, a new TEK is generated by the SGM and reported to the CA. CA will give the necessary re-computation according to obtained information. To have a bound on the maximum amount of time that TEK of SG\(_i\) can be held by a member outside SG\(_i\), each SGM maintains a timer. Once the time reaches a certain value, the original subgroup re-keys itself and the timer is reset to zero. To trace a member’s movement history, SGM\(_i\) maintains a table of subgroup members that hold a valid TEK while residing outside the subgroup. The table is reset when a member holding the TEK leaves the group or when the timer expires. A member is added to the table when it moves out of SG\(_i\), and a member is removed from the table when it moves back to SG\(_i\).

Using such an approach, FEDRP behaves with a lower re-keying rate than merely being treated as first leaving then joining the hierarchical infrastructure consisting of independent subgroups \[20\].

5.6. Attacks analysis

5.6.1. Continuous attack

There is a drawback in Lin’s scheme \[13\] that once the old TEK of a subgroup is exposed, the newly chosen TEK will be revealed in sequence. Our SHKM scheme overcomes this problem efficiently. The equations used in Lin’s scheme and SHKM show their differences. Eqs. (7) and (8) are from the scheme of Lin. Let’s assume that subgroup \(S_i (1 \leq i \leq m)\) changed the TEK from \(k_i\) to \(k_i^*\), but it has no idea that \(k_i\) is exposed to attackers. It’s obvious that the attack has got the value of \((Z^{k_j} \oplus ID_i) \mod P\) by requesting the related factor \(\gamma_{ji}\). After CA has recomputed the new \(\gamma_{ji}^*\), the attacker can easily derive the new group key of \(S_i\) by performing Eq. (9).

\[
\gamma_{ji} = (Z^{k_j} \oplus ID_i \mod P) \oplus k_i
\]  \hspace{1cm} (7)

\[
k_i = (Z^{k_j} \oplus ID_i \mod P) \oplus \gamma_{ji}
\]  \hspace{1cm} (8)

\[
k_i^* = (Z^{k_j} \oplus ID_i \mod P) \oplus \gamma_{ji}^*.
\]  \hspace{1cm} (9)
The reason for causing the drawback is that after TEK is changed, the value of \((Z^{k_j} \oplus ID_i \mod P)\) keeps static and will not change as other parameters \([21]\). So there is a chance that attackers can deduce the new TEK and eavesdrop on encrypted messages once the old TEK is revealed. To be different, in our scheme, the changing residue \(e_i\) is related with \(k_i\). Once \(k_i\) is changed, the value of \(e_i\) is also changed with \(k_i\). Even if the attack has got the old \(h \left( Z^{k_j} \oplus e_i \mod P \right)\) and \(\gamma^*_{ji}\), when it’s performing Eq. (10) it has to face the difficulty of computing new \(h \left( Z^{k_j} \oplus e^*_i \mod P \right)\). For the value of \((Z^{k_j} \oplus e^*_i \mod P)\) is protected by the one-way hash function, so it’s hard to reverse the hash function. Furthermore, even if \((Z^{k_j} \oplus e^*_i \mod P)\) is disclosed, \(n\) is secret and the changing residue \(e^*_i\) is changing with the change of \(k_i\) all the time, thus it’s impossible for attackers to reveal the \(k^*_i\).

\[ k^*_i = h \left( Z^{k_j} \oplus e^*_i \mod P \right) \oplus \gamma^*_{ji}. \]  

5.6.2. Sibling attack

Suppose \(S_j\) has two successor subgroups \(S_l\) and \(S_i\), which want to crack their common predecessor’s TEK \(k_j\). By Eqs. (11) and (12), they can obtain \(h \left( Z^{k_j} \oplus e_i \mod P \right)\) and \(h \left( Z^{k_j} \oplus e_l \mod P \right)\). However, from the above analysis, we can understand it is also infeasible to crack \(k_j\).

\[ h \left( Z^{k_j} \oplus e_i \mod P \right) = \gamma^*_{ji} \oplus k_i \]  
\[ h \left( Z^{k_j} \oplus e_l \mod P \right) = \gamma^*_{jl} \oplus k_l. \]

6. Evaluation and conclusions

We evaluate the effectiveness of the proposed SHKM protocol to the defense of malicious attacks and the communication and computation complexities. From the above analysis, we can prove that our protocol is more difficult to crash compared with Lin’s protocol. As to the computation complexity, we are concerned with the storage of all group members, the number of re-keying messages for membership changes, and the number of encryptions and decryptions. In the proposed protocol, we need no re-keying messages but some reporting messages between CA and subgroups. Since in every subgroup we can use the key graph to manage the inner key distribution, the number of keys held by each user is not obviously reduced. However, to deal with all kinds of mobility dynamics, our scheme reduces the number of encryptions and the number of re-keying messages, which contributes a lot for the scalability of our protocol.

In Table 1, we compare our protocol with some existing scalable key management protocols \([22]\). We use \(n\) to denote the number of group users; \(s\) to denote the number of subgroups; \(t\) to denote the average number of all the successors of one subgroup. Moreover, we use \(m\) to denote the average number of the direct successors of one subgroup, \(e\) to denote a run of residue computation; and \(r\) to denote a run of related factor computation. We can find that SHKM does not need to have re-keying messages but some reporting messages between CA and subgroups. The encryption cost, which occurs when there are
Table 1
The comparison with other protocols

<table>
<thead>
<tr>
<th></th>
<th>Simple</th>
<th>LKH</th>
<th>Key graph</th>
<th>Iolus</th>
<th>Lin’s</th>
<th>SHKM encryption/CA cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of all keys</td>
<td>(O(n))</td>
<td>(O(n))</td>
<td>(O(n))</td>
<td>(O(n))</td>
<td>(O(n))</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Keys held by each user</td>
<td>(2)</td>
<td>(O(\log n))</td>
<td>(O(\log n))</td>
<td>(2)</td>
<td>(2)</td>
<td>(O(\log n))</td>
</tr>
<tr>
<td>Re-keying message size</td>
<td>(O(n))</td>
<td>(O(\log n))</td>
<td>(O(\log n))</td>
<td>(–)</td>
<td>(–)</td>
<td>(–)</td>
</tr>
<tr>
<td>Storage of subgroup agents</td>
<td>(O(n))</td>
<td>(O(n))</td>
<td>(O(n))</td>
<td>(2)</td>
<td>(2)</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Members</td>
<td></td>
<td>(O(n))</td>
<td>(O(\log n))</td>
<td>(O(\log n))</td>
<td>(O(s))</td>
<td>(2)</td>
</tr>
<tr>
<td>Leave</td>
<td></td>
<td>(O(n))</td>
<td>(O(\log n))</td>
<td>(O(\log n))</td>
<td>(O(s))</td>
<td>(2+t)</td>
</tr>
<tr>
<td>Subgroup</td>
<td></td>
<td></td>
<td>(O(s))</td>
<td>(1)</td>
<td>(1)</td>
<td>((e+r))</td>
</tr>
<tr>
<td>Join</td>
<td></td>
<td></td>
<td>(O(s))</td>
<td>(1+t)</td>
<td>(m)</td>
<td>(m(e+r))</td>
</tr>
<tr>
<td>Leave</td>
<td></td>
<td></td>
<td>(O(s))</td>
<td>(1)</td>
<td>(1)</td>
<td>((e+r))</td>
</tr>
<tr>
<td>Delete</td>
<td></td>
<td></td>
<td>(O(s))</td>
<td>(1+t)</td>
<td>(m)</td>
<td>(m(e+r))</td>
</tr>
<tr>
<td>Trust on the third party</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Dynamic changes, is significantly reduced in SHKM compared with other decentralized protocols. The saved cost is added to the CA, which has high computation capability.

Security mechanisms are an urgent requirement for multicast communications in order to ensure a safe and large-scale deployment for confidential group communications. Key management plays an important role in the secure multicast architecture. In our proposed scheme, scalability is achieved by cryptographic key assignment to the hierarchical subgroups with different priorities. In the SHKM protocol, unlike most decentralized protocols, the decryption and re-encryption at the boundaries of subgroups are avoided. We can see that the contribution comes from the role of the CA, which undertakes a lot of tasks for re-computing the related parameters of the architecture when dynamic changes occur frequently. If we can ensure that the CA performs faster in a secure way, the performance of the whole system will be satisfactory. In addition, the redundant usage of CAs makes the whole system more fault tolerant than distributed key agents.

Acknowledgments

This work is supported by the Hong Kong Polytechnic University Central Research Grant G-YY41, the University Grant Council of Hong Kong under the CERG Grant PolyU 5170/03E, and the National Natural Science Foundation of China under Grant Number 60503007.

References


**Prof. Jiannong Cao** received the B.Sc degree in computer science from Nanjing University, Nanjing, China in 1982, and the M.Sc and the Ph.D. degrees in computer science from Washington State University, Pullman, WA, USA, in 1986 and 1990 respectively. He is currently a Professor in the Department of Computing, the Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. He is also the director of the Internet and Mobile Computing Lab in the department. He was on the faculty of computer science at the James Cook University and the University of Adelaide in Australia, and the City University of Hong Kong. His research interests include parallel and distributed computing, networking, mobile computing, fault tolerance, and distributed software architecture and tools. He has published over 160 technical papers in the above areas. He has served as a member of editorial boards of several international journals, a reviewer for...
many international journals/conference proceedings, and also as an organizing/programme committee member for many international conferences. Prof. Cao is a member of the IEEE Computer Society, the IEEE Communication Society, IEEE, and ACM. He is also a member of the IEEE Technical Committee on Distributed Processing, IEEE Technical Committee on Parallel Processing, IEEE Technical Committee on Fault Tolerant Computing, and Computer Architecture Professional Committee of the China Computer Federation.

Ms. Lin Liao received the B.Sc and M.Sc degrees in computer science from Central South University, Changsha, PR China, in 2002 and 2005 respectively. She was a Research Assistant in the Department of Computing, the Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong from October 2004 to March 2005. Her research interests include security issues in mobile Internet, mobile ad hoc networks, and wireless sensor networks.

Prof. Guojun Wang was a Research Fellow (March 2003–March 2005) in the Department of Computing, the Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. He is currently a Professor in the Department of Computer Science, the Central South University, Changsha, PR China. He is also the director of the Mobile Computing Lab in the department. He received his B.Sc. degree in Geophysics, M.Sc. degree in Computer Science, and Ph.D. degree in Computer Science, from Central South University, in 1992, 1996 and 2002, respectively. His research interests include computer networks, group communications, fault tolerance, and mobile computing. He has published over 80 technical papers in the above areas. He has served as a reviewer for many international journals such as IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, and also as a TPC member for many international conferences such as ICC, IWCMC, EUC, ISPA, and MSN. He is also a senior member of the China Computer Federation, a YOCSEF member of the China Computer Federation, and a member of the Hunan Provincial Association of Computers.