An efficient group key management scheme for mobile ad hoc networks

Bing Wu*

Department of Mathematics and Computer Science,
Fayetteville State University,
Fayetteville, NC 28311, USA
E-mail: bwu@uncfsu.edu
*Corresponding author

Jie Wu

Department of Computer Science and Engineering,
Florida Atlantic University, Boca Raton, FL 33431, USA
E-mail: jie@cse.fau.edu

Yuhong Dong

Department of Advanced Technologies,
Alcorn State University, Alcorn State, MS 39096, USA
E-mail: ydong@alcorn.edu

Abstract: Group key management is one of the basic building blocks in collaborative and group-oriented applications in Mobile Ad Hoc Networks (MANETs). Group key establishment involves creating and distributing a common secret for all group members. However, key management for a large and dynamic group is a difficult problem because of scalability and security. Modification of membership requires the group key to be refreshed to ensure backward and forward secrecy. In this paper, we propose a Simple and Efficient Group Key (SEGK) management scheme for MANETs. Group members compute the group key in a distributed manner.

Keywords: group key; key management; MANETs; mobile ad hoc networks.

Reference to this paper should be made as follows: Wu, B., Wu, J. and Dong, Y. (2008) 'An efficient group key management scheme for mobile ad hoc networks', *Int. J. Security and Networks*, Vol.

Biographical notes: Bing Wu is an Assistant Professor in the Department of Mathematics and Computer Science at the Fayetteville State University. He completed his PhD Degree in Computer Science from Florida Atlantic University in 2006. Since 2000 he has been researching in mobile ad hoc and sensor networks and distributed systems.

Jie Wu is a Distinguished Professor at the Department of Computer Science and Engineering, Florida Atlantic University and a Program Director at National Science Foundation. He has published over 400 papers in various journals and conference proceedings. He is currently on the Editorial Board of IEEE Transactions on Mobile Computing. He was on the Editorial Board of IEEE Transactions on Parallel and Distributed Systems and was a co-guest-editor of IEEE Computer and *Journal of Parallel and Distributed Computing*. He has served as an IEEE Computer Society Distinguished Visitor and is the Chairman of IEEE Technical Committee on Distributed Processing (TCDP).

Yuhong Dong is an Assistant Professor in the Department of Advanced Technologies at Alcorn State University. She completed her PhD Degree in Computer Science from Florida Atlantic University in 2006. Her research interest is in the areas of Bayesian network, intrusion detection system (IDS), and imaging processing.

1 Introduction

To secure communications in Mobile Ad Hoc Networks (MANETs), messages are often protected by encryption using a chosen cryptographic key, which, in the scenario of group communication is called the group key. Only group members who know the current group key are able to recover the original message.

Group key establishment means that multiple parties want to create a common secret to be used to exchange information securely. Without relying on a central trusted entity, two people who do not previously share a common secret can create one based on the DH protocol. The 2-party Diffie-Hellman (DH) protocol can be extended to a generalised version of n-party DH. Research efforts have been put into the design of a group key management scheme for the sake of scalability, reliability, and security. Furthermore, group key management also needs to address the security issue related to membership changes. The modification of membership requires refreshment of the group key. This can be done either by periodic rekeying or updating right after member change. The change of group key ensures backward and forward security.

Group key management protocols can be roughly classified into three categories; centralised, decentralised, and distributed (Rafaeli and Hutchison, 2003). In centralised group key protocols, a single entity is employed to control the whole group and is responsible for group rekeying. In the decentralised approach, multiple entities are responsible for managing the group as opposed to a single entity. In the distributed method, group members themselves contribute to the formation of a group key and are equally responsible for the rekeying and distribution of group keys. Recently, collaborative and group-oriented applications in MANETs have been an active research area. Obviously, group key management is a central building block in securing group communications in MANETs. However, group key management for large and dynamic groups in MANETs is a difficult problem because of the requirement of scalability and security under the restrictions of nodes' available resources and unpredictable mobility.

In this paper, we propose a simple and efficient group key management scheme, simply called SEGK, for MANETs. The basic idea of SEGK is that a physical multicast tree is formed in MANETs for efficiency. Group members take turns acting as group coordinator to compute and distribute intermediate key materials to group members. The keying materials are delivered through the tree links. The coordinator is also responsible for maintaining the connection of the multicast group. All group members can compute the group key locally in a distributed manner. The major contributions of our research work are the following:

- We propose a distributed group key management scheme for MANETs. It is based on n-party DH. A physical multicast tree is formed for efficiency and double multicast trees are constructed and maintained for fault tolerance.
- We present detailed operations of the group key establishment protocol. We also propose two methods for handling network dynamic, including joining and leaving.
- We evaluate and analyse the performance of the proposed scheme in the dynamic network environments.

This paper is organised as follows: Section 2 reviews related work. Details of the group key management scheme are described in Section 3. In Section 4, we present and analyse the simulation environment and results. We conclude the paper and discuss possible future work in Section 5.

2 Related work

2.1 Group key management

The secure group communication over insecure channels problem has been an active research topic in wired networks for quite a while. In a traditional centralised model, there is a single entity called a Trusted Third Party (TTP), such as a Key Distribution Center (KDC), which distributes a secret key to group members (Wu et al., 2005). Normally, TTP shares a secret key with each group member. The KDC generates a group key, encrypts it with the pairwise key, and then distributes it to the corresponding group member. This scheme is easy to implement and is storage efficient for every group member. However, it is not efficient for the KDC to handle changes of group membership.

Several optimisations have been proposed for the centralised approach, such as Logic Key Hierarchy (LKH) (Wong et al., 1998; Wallner et al., 1998) and One-way Function Tree (OFT) (Sherman and McGrew, 2003), which are based on the efficiency of key tree structure and hash function. In LKH, each leaf node corresponds to a group member and the root node is a Group Key Controller (GKC), which is a TTP. Members share a pairwise key with the root node as well as a set of intermediate keys from it to the root. In the event of member change, a rekey message is generated containing each of the new sets of keys encrypted with its respective node's children keys. In LKH, all keys generated by GKC are unrelated with each other. In OFT, however, keys are related with each other through the operations of a hash function and a mix function. Each group member needs to store a set of blinded keys from its sibling set and can deduce the set of keys for its ancestor set which includes the group key. The rekeying message is reduced by half compared with LKH.

Although the ideas of the above centralised group key management schemes are technically sound for wired or other fixed networks, they cannot be easily applied to MANETs where no TTP is available. A centralised scheme also has a critical weakness, a single point of failure or hot spot of attacks in a potentially hostile environment. A distributed group key management approach could be more appropriate for MANETs since it assumes no existence of TTP. A number of distributed group key management schemes have been introduced to adapt to the distributed and dynamic environment. The distributed

approach is sometimes called contributory group key management. Each group member contributes an equal share to the common group key. Most distributed schemes are based on the n-party DH protocol.

Steiner et al. (2000) presented a suite of group key agreement protocols. GDH.1 and GDH.2 protocols need a total of $O(n^2)$ exponentiations. GDH.3 needs $O(n)$ exponentiations. In GDH.3, member contributions are accumulated at upflow direction and broadcasted at downflow direction, followed by the factoring out of a corresponding exponent. One user collects all inputs from the previous stages, raises the power and broadcasts the resulting values to the rest of the group. At the end, every group member can compute the group key.

Burmester and Desmedt (1994) proposed another group key scheme called BD. It is an additive DH scheme. That is, each group user will come up with the same secret as $K_G = g^{r_1r_2+r_2r_3+\cdots r_nr_1} \mod p$, which is the group key shared by all group members. Steer, Strawczynski, Diffie, and Wiener put forward STR scheme (Steer et al., 1990). In this scheme, given all blinded member keys, the first and the second member can calculate the group key, that is at the form of $K_G = g^{r_n g^{r_{n-1} g^{\cdots g^{r_3 g^{r_1 r_2}}}}}$. However, users 3 to n require intermediate keying materials to calculate the group key. Either the first or the second user can broadcast the intermediate keys for the rest of the group members. Kim et al. (2000) introduced a group key management scheme called TDGH. Their idea was to combine the efficiency of key tree structure with the contributory feature of DH. In TDGH, a sponsor takes a special role, which can involve computing keys, and broadcasts the blinded keys to the group during events of member join, leave, partition, or merge.

In summary, the above review shows several group key management protocols. Basically, they only give the mathematical models and none of them gives a detailed implementation of group key management protocol to a particular network.

2.2 Multicast

To establish a common group key, a group key management protocol needs mobile nodes to work together closely in order to perform the group key agreement and rekeying at any event of group membership change. Multicast provides a useful means for group communications. Some scalable multicast protocols have been proposed (Bae et al., 2000; Lee et al., 2000). Although the multicast service can be achieved based on unicast, it is not efficient and could cause much unnecessary network traffic. The traditional multicast protocols can be classified into tree-based (such as DVMRP Deering and Cheriton, 1990 and MAODV Royer and Perkins, 1999), and mesh-based approaches (such as ODMRP Lee et al., 2000 and FGMP Chiang et al., 1998). Recently, some efforts have been made in the design of better-performance-multicast-routing protocols in MANETs by using overlay networks or a backbone to constrain the spread of state information. These protocols can be classified into the overlay-based, backbone-based, stateless, and hierarchical approaches. Some of these protocols can be found in Xie et al. (2002), Kaikaeo and Shen (2002) and Ji and Corson (2001).

2.3 Our approach

In this paper, we introduce a simple group key management scheme. Our objective is to adapt the traditional scheme to the MANETs settings. The basic idea of our scheme is that a multicast tree is formed in MANETs for efficiency. Two multicast trees are constructed and maintained in parallel to achieve fault tolerance. When a link of one tree is broken, it is substituted by the other tree. We call one tree a blue tree, the other one a red tree. Group members take turns to act as a group coordinator to compute and distribute the intermediate keying materials to all members through the active tree links. The operation can be made in rounds and the coordinator is selected in a distributed way as described in Wu et al. (2005). The coordinator is also responsible for maintaining the connection of the multicast group. A group coordinator computes and distributes the intermediate keying materials to all members through the underlying tree links. We present the detailed operations of the formation of double trees, the group key established protocols, and the operations to handle the joining and leaving members in next section.

3 The group key management scheme-SEGK

3.1 Notations and assumptions

We assume that every node carries a valid certificate from offline configuration before entering the network. A smart card can be used for this pre-configuration. Therefore, there is an underlying public key infrastructure to manage certificates. In our previous work, we proposed a decentralised public key management scheme (Wu et al., 2005). A number of research papers have addressed this topic. However, most solutions suffer the man-in-the-middle attack. In this paper, we assume that each group member has a unique identifier and all keying materials are digitally signed by corresponding initiators to ensure authenticity and integrity, and to defend against man-in-the-middle attacks. The group access control depends on the group membership policy. A member can carry some secret information (such as a password) in order to join the group or a node can join a group if it can present a valid certificate, etc. Here, for simplicity, we assume that a node can join a group if it has a valid certificate. Some notations used in SEGK are listed in Table 1.

3.2 Overview of SEGK

In SEGK, to form a common group key, every group member contributes a share of the final common group key. The group key can be refreshed periodically or only be

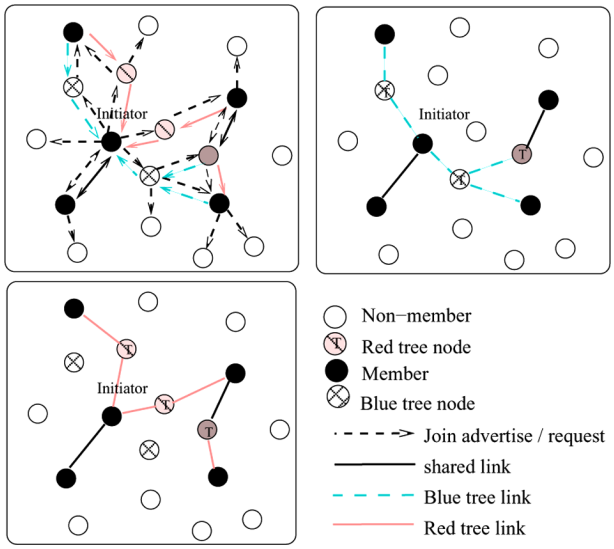
updated in response to changes of group membership. The updating of the group key helps to enforce backward and forward secrecy of group communications. Obviously, efficiently exchanging keying materials is critical in MANETs. In SEGK, all keying materials are disseminated through the underlying multicast tree links. A native broadcast through flooding is obviously not appropriate because of large redundancy which may result in network traffic congestion. There are many multicast routing protocols that have been proposed, as we described in Section 2. Here, we present a reliable double multicast tree formation and maintenance protocol. Our idea is similar to Wei and Zakhor (2004), however, our double tree scheme guarantees that two trees cover all group members. Logically, the two trees are identical from a group member's point of view. In Wei and Zakhor (2004), some group members included in one tree might not be included in the other tree, which is obviously not desirable for group key management. The multicast routing protocol serves as a subsystem of our group key management framework. The detailed protocols are present in the following sections.

Table 1 Some notations used in SEGK scheme

M_i	A group member with ID i
M_c	The current group coordinator
n	Total number of group members
g	Exponentiation base
r_i	a random number generated by member i, also called member key
br_i	Member i's blinded member key, $br_i = g^{r_i} \mod p$
k_i	Internal node i's key, $k_i = (br_i)^{k_i}$, also called intermediate key
bk_i	Blinded internal node i's key, $bk_i = g^{k_i}$, also called blinded intermediate key
$h(m)$	The digest of m
K_G	The common group key

A group initiator (also called group coordinator) starts the group initialisation process by broadcasting a *join advertise* message across the entire network. A sequence number is used to avoid loops. A node is associated with three colours, namely blue, red, and grey. A node will choose grey as it's colour if its total number of neighbours is less than a predefined threshold value (for instance, half of average node degree). All member nodes are grey. Other network nodes randomly choose blue or red as their colour with probability equal to 0.5. For the first received message, a grey node stores the upstream node ID and rebroadcasts the message. For a non-grey node, it stores the upstream node ID and rebroadcasts the message only if the upstream node is the same colour, a sender, or a grey node. Based on the *join response* back from group member to the group initiator, two double multicast trees are formed in parallel. Both trees consist of group members and intermediate non-member nodes. A node could send out join requests to a group. Any existing group member can send replies back. The procedure of handling join requests is similar to the above group advertisement to ensure the consistency of double multicast tree structures. The resultant two trees could be disjoint or may share a common node. Thus, a dynamic double multicast tree structure is constructed. Figure 1 illustrates the construction of a double multicast tree.

Figure 1 Illustration of a double multicast tree structure (see online version for colours)



3.3 Construction of double multicast trees

The problem with the above approach is that the two multicast trees may not be identical, which means some group members may be covered by the blue tree, but are not included in the red tree. This scenario can happen when a group member is surrounded by nodes with only one colour, either red or blue. A group member can detect this scenario if it has received messages from only one colour (either blue or red) of nodes. Then, this member node can request one of its upstream node to change its colour to grey.

Initially the group initiator is responsible for sending out member refresh messages periodically to maintain the connection of the double multicast tree structure. After a predefined amount of time of operation, a group member could decide to act as a group coordinator and notify the group that it is on duty to maintain the group. All members need to take turns acting as group coordinator. The double multicast trees can be used by enabling one tree in an active state and the other one in an inactive state as a backup. After a predetermined period of time for the group initialisation phase, the multicast trees have been formed and the group coordinator can invoke the group key establishment procedure. This procedure is described later in the group key establishment section.

3.4 Detection of leaving members

The processing of handling members who leave is more complicated than handling the joining of new members. To join the group, a new user needs to broadcast a request.

The new user becomes a legitimate group member once its request is approved by any existing group member or by the current group coordinator. However, for the scenario of leaving members, we cannot assume that a leaving member will send out a leaving notice. A member could leave the group silently. Even if it could send out a message and notify its leaving, this notice could get lost in a dynamic environment. We could define a physical leaving and a logical leaving. For the physical leaving, a node moves out the range of the network or it switches its transmitter off. For a logical leaving, a node still stays inside the network, but it does not participate in the group activity. We present two methods to address these problems.

3.4.1 Method one

One strategy is that current group coordinator sends out *member refresh* messages periodically through both tree links. All group members should send an *ack* message back to indicate its affiliation interests (status). The group coordinator will determine whether a member remains attached or has left based on its response within a certain amount of time. It is the member's responsibility to broadcast a message in a controlled flooding scheme to reconnect to the group if it has not heard the periodic member refresh message. If a member does not want to be part of the group it could keep silent without sending the *ack* message. The tree structure is updated based on the control messages. Some links could be pruned and new links could be added since a member could move to a new location. The current group coordinator notifies the member change event to all members through the updated tree structure. This strategy is very efficient and is appropriate for a relatively static network environment.

3.4.2 Method two

Another strategy is that the group initiator or current group coordinator periodically broadcasts *member enforcement* messages in a controlled flooding scheme. The default flooding range is set to the maximum distance from the coordinator to the members. The search range can be increased until it reaches a threshold value or the current network diameter. All group members will send a response back. Thus, members affiliated to the group are refreshed. This strategy is quite costly compared with Method one, and is more appropriate for a highly dynamic environment where nodes move frequently and cause the connections to be broken frequently.

3.5 Group key establishment protocol

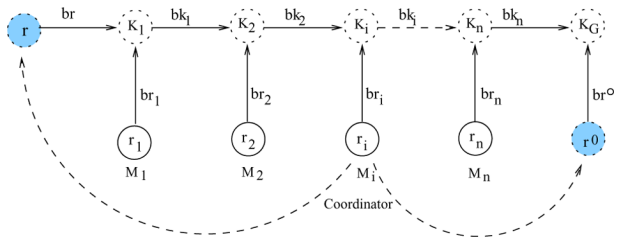
The basic idea of a group key agreement protocol is that all group members maintain a logic key tree in local storage space. The key tree is used to deduce the final common group secret. Most contributory group key approaches maintain a certain type of key repository. They differ in the way they accumulate and distribute intermediate keys. Some are based on the key ring, others may be based on key tree, etc. Our scheme is based on the key tree structure. Although our key tree is similar to STR, we introduce several improvements as we describe below. The key tree structure in SEGK is shown in Figure 2.

- To distribute the workload of keying service, we introduced the concept of coordinator. The coordinator is responsible for computing and distributing intermediate keys to all group members. It also needs to handle member join and leave. The role of coordinator is rotated among all members.
- For efficient switching of the role of coordinator, we introduce two dummy nodes at two ends of the key tree for efficient group key refreshing and the group coordinator role switching.
- A new group member can be easily absorbed into the group by adding new members into the current rightmost position and moving itself to the right.
- When a member leave is detected, the coordinator generates a new random key r and multicast the blinded value br as well as other intermediate blinded keys.

group key initialization algorithm:

1. Round 1: The group initiator $M_c, i \in [1,n]$ advertises a blinded random member key br_c and two virtual blinded member keys br and br_o. Each interested member $M_i, i \in [1,n], i \neq c$ responds with a blinded random member key br_i.
2. Round 2: The initiator M_c computes intermediate keys $k_i = (br_i)^{k_{i-1}}$ and multicasts blinded intermediate keys $bk_i = g^{k_i}, \forall i \in [1,n]$. $k_0 = r$.
3. Every $M_i, i \in [1,n]$ computes $k_i = (bk_{i-1}^{r_j}\ (k_0 = r)$ and recursively $k_j = (br_j)^{k_{j-1}}, \forall j \in [i,n]$, $K_G = br_o^{k_n}$.

Figure 2 Illustration of key tree structure (see online version for colours)



3.5.1 Group key initialisation

At the initialisation phase, the coordinator announces its role and broadcasts two random keys r and r_o. Normally the group initiator acts as group coordinator at the beginning. The order of members on the key tree is sorted by their ID at the initialisation phase. However, at subsequent member add events, a new member is always added at the rightmost position of the key tree. This rule should be followed by all members

to ensure that key trees in all members' local memory are consistent. One solution is that the group coordinator explicitly indicates the structure of the key tree. This can also be done implicitly by the coordinator since it needs to multicast blinded intermediate keying materials to all group members. All keying materials are put in one package and the order of blinded intermediate key materials shows the structure of the key tree. The algorithm for the group key initialisation phase is shown below.

3.5.2 Member addition

A new group member can be easily added into the group by inserting it into the current rightmost position and moving the dummy coordinator to the right. The major advantage of our approach is that the coordinator does not need to generate a new random key but still provides key independence. This means that knowing the previous group key cannot help to deduce the new group key. Given two blinded keys, the new member can deduce the new group key, however, it cannot deduce the former group key. This ensures backward secrecy. Figure 3 illustrates the operation of joining a new member. The algorithm of member addition is shown below.

Member addition algorithm:

1. Round 1: New member M_{n+1} generates a random member key r_{n+1} and broadcasts the blinded value br_{n+1}. The coordinator unicasts the blinded keys bk_n and br_o to the new member.
2. Every $M_i, i \in [1,n]$ computes $k_{n+1} = (br_{n+1})^{k_n}$, the new member computes $k_{n+1} = (bk_n)^{r_{n+1}}$, and $M_i, i \in [1,n+1]$ can compute $K_G = br_o^{k_{n+1}}$.

Member leave algorithm:

1. Round 1: M_c notifies the group of the leaving member M_ℓ. M_c generates a random member key r' and multicasts the blinded value br' as well as blinded intermediate keys, $k_i = (br_i)^{k_{i-1}}$ and $bk_i = g^{k_i}, \forall i \in [1,n]\backslash\{\ell\}$. $k_0 = r'$.
2. Every $M_i, i \in [1,n]\backslash\{\ell\}$ computes $k_i = (bk_{i-1}^{r_j}$ $(k_0 = r')$ and recursively $k_j = (br_j)^{k_{j-1}}, \forall j \in [i,n]\backslash\{\ell\}$, $K_G = br_o^{k_n}$.

3.5.3 Member leave

The leaving group member event can be detected either by explicit notification from the leaving node or through the scheme described in Section 3 at subsection D through Method one or Method two. The coordinator notifies all group members of the member leaving event and multicasts a new blinded random key to all members. All group members can compute the new group key. The algorithm is shown below. Figure 4 illustrates the operation of a leaving member.

3.5.4 Group key refresh/reinforce

The group key may need to change periodically, and may not be related to any change of group membership. The purpose of refreshing the group key periodically is to prevent the long time use of group keys which could be compromised. This process can be implicitly done during the switch of coordinator, or explicitly performed by the coordinator which generates a new random key r'' and multicasts the blinded value br''. We show the group key refresh/reinforce algorithm below.

Member refresh/reinforce algorithm:

1. Round 1: $M_{c'}$ notifies the group of its role of acting as a new coordinator. $M_{c'}$ generates a random member key $r^\diamond$ and multicasts the blinded value $br^\diamond$ as well as blinded intermediate keys, $k_i = (br_i)^{k_{i-1}}$ and $bk_i = g^{k_i}, \forall i \in [1,n]$. $k_0 = r^\diamond$.
2. Every $M_i, i \in [1,n]$ computes $k_i = (bk_{i-1}^{r_j}$ $(k_0 = r^\diamond)$ and recursively $k_j = (br_j)^{k_{j-1}}, \forall j \in [i,n]$, $K_G = br_o^{k_n}$.

Figure 3 Illustration of key updating for joining member (see online version for colours)

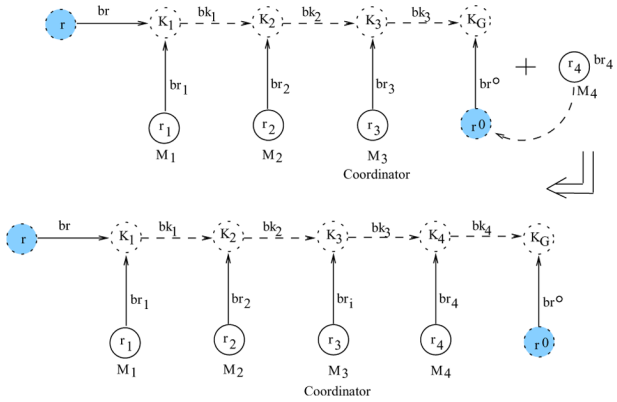


Figure 4 Illustration of key updating for leaving member (see online version for colours)

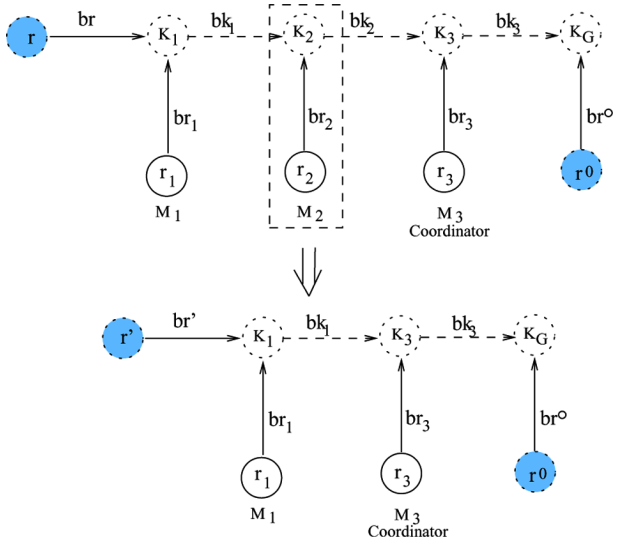
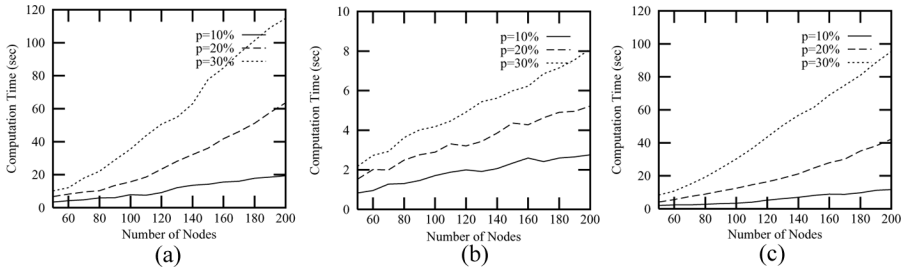


Figure 5 Average computation time for SEGK; (a) group key initialisation; (b) member addition and (c) member leaving



3.5.5 Summary

We described the detailed operation of the group key establishment algorithms in SEGK. The advantage of our scheme is that a member add and leave can quickly respond and update the group key efficiently to ensure backward and forward security. The group maintenance role can be transited from one member to the other in a simple way. The adjustment of underlying multicast communication structure as well as the event of member addition and leaving can be done by two methods. Method one is more efficient and appropriate for a static environment than Method two. Method two is designed for highly dynamic network environments. The algorithms of performing member join, leave and periodic group key refreshing are presented.

4 Experiments

In this section, we give the simulation environment and present the experimental results. We analyse the performance of our key management scheme according to message cost and total computation time. We also compare the performance of our scheme with existing schemes.

4.1 Simulation environment

The simulation was implemented in Matlab. The simulation was conducted in a 100×100 2-D free-space by randomly allocating a given number of nodes in the range from 50 to 200. We conduct experiments in a dynamic network environment. We assume every node has fixed transmission range $r = 20$. Two nodes are directly connected if their distance is within each other's transmission range. For each host in an update interval, the corresponding host may move within the range of l units in any direction or remain stable in the corresponding internal with the possibility ρ (l is 5 and ρ is 0.5 in our simulation). We compared the results with some of the proposed distributed approaches and ignore other centralised approaches. In the experiments, we compared the cost of SEGK with the STR and GDH scheme. We implemented 1024-bit prime number as random key. A base $g = 2$ with the module n (1024 bits) is used to compute the blinded random key as well as the blinded intermediate keys. A time stamp, sequence number, flags, and keying materials are concatenated and hashed using MD5 (256 bit), and then signed by the senders' private key. We used the Maple mathematical package which is included in Matlab to handle exponentiations of large numbers.

4.2 Cost measurement

In the experiments, the cost is measured both by the number of messages and the computation time. For the SEGK scheme, the message cost includes the multicast of blinded random keys and the intermediatory keying materials. The multicast tree nodes includes all group members as well as non-group forwarding nodes needed to forward messages. Every forwarded message is counted. We analyse the message cost for the group key initialisation process and for member joining and leaving scenarios. For the STR scheme, the message cost includes the flooding of all blinded keying materials, which includes random keys and intermediatory keys. In the GDH scheme, the message cost includes unicast messages of blinded random keys, and broadcast messages of blinded intermediate keys. The computation cost mainly

Figure 6 Average computation time when $p = 10\%$; (a) group key initialisation; (b) member addition and (c) member leaving

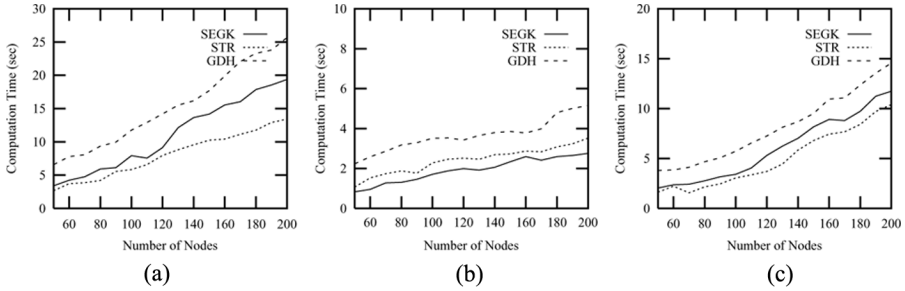
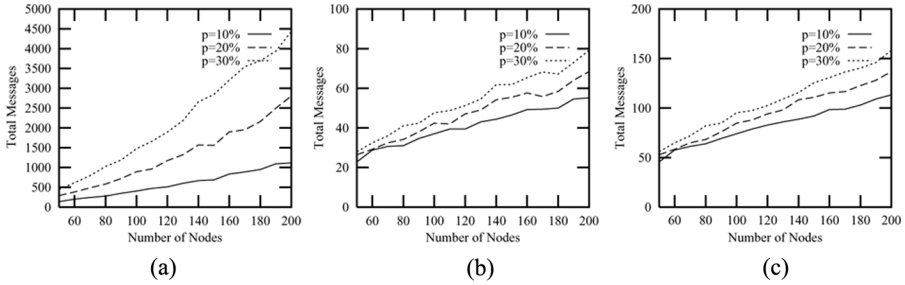


Figure 7 Average message cost for SEGK; (a) group key initialisation; (b) member addition and (c) member leaving



includes generating large prime numbers, performing exponentiations, hashing, and producing signatures.

4.3 Result analysis

Figures 5–8 show the experimental results. Figures 5 and 7 show the computation cost and message cost of the SEGK scheme. Figure 5(a)–(c) show the computation cost for SEGK in processing group key initialisation, member addition, and member leave, respectively. Figure 5(a) shows the computation cost of group key initialisation with group membership rate $p = 10\%, 20\%$, and 30%. Computation time increases as the number of nodes increase. For example, when $p = 10\%$ and $n = 100$, computation time is 7.9 seconds, and 19.3 seconds for $n = 200$. The cost increases quickly with the increase of p. Figure 6(a) shows that the computation cost increases as the number of nodes increase for all three schemes (SEGK, STR, and GDH) in group key initialisation. Under the same network settings, GDH has the highest computation cost for group key initialisation, STR has the least cost, and SEGK has a cost in between.

Figure 5(b) shows computation cost of group rekeying because of a newly joined group member. The computation cost of group key updating is relatively low compared with group key initialisation and the group member leave process. Similarly, computation time increases with the increased number of group members. In addition, the computation cost is higher with larger p value, but the increasing rate is not significant. Figure 6(b) shows the computation cost with $p = 10\%$ for different schemes in processing group rekeying at events of member addition. For all three schemes, the computation cost of group key updating is relatively lower than group key initialisation process. However, SEGK has the smallest computation time among the three, while GDH still produces the largest computation cost.

Figure 5(c) shows the computation cost of group rekeying at the event of member leaving. At the event of group member leave, the cost of group key updating is relatively lower than group key initialisation, but is significantly higher than member addition. Similar to the group key initialisation processing, computation time increases with an increased number of group members. The computation cost at member leave increases quickly with the increase of p. Figure 6(c) shows the computation cost of group rekeying at the event of member leave for different schemes. The computation cost of group key updating for a member leaving is relatively less than the group key initialisation process, but much higher than the member addition process. While GDH still produces the largest computation cost, SEGK generates higher cost than STR. STR has the least cost, which is similar to the group key initialisation process.

Figure 7(a)–(c) show the total number of messages for SEGK in processing group key setup, member addition, and member leave, respectively. Figure 7(a) shows the message cost of group key initialisation with group membership rate $p = 10\%, 20\%$, and 30%. Message cost increases as the number of nodes increase because of the increased number of group members as well as non-member forwarding nodes. However, the increase rate is higher for larger p. For instance, when $n = 100$, the message cost increases from 407 to 1475 for $p = 10\%$ and 30%, respectively. From Figure 8(a) we can see that under the same network settings, STR requires more messages than the other two schemes, GDH has the least messages cost while SEGK requires the medium amount of messages during group key initialisation processing.

Figure 8 Average message cost when $p = 10\%$; (a) group key initialisation; (b) member addition and (c) member leaving

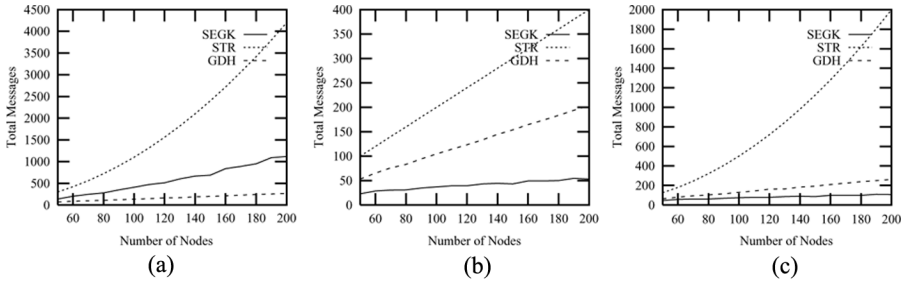


Figure 7(b) shows the message cost for updating group key at the event of member addition with different group membership rate. Message cost increases as the number of nodes increases. However, the message cost is relatively small compared with the previous group key initialisation process. For instance, when $n = 100$ and $p = 10\%$, the message cost is 37. The message cost also increases with the p value. However, the increase rate is not significant. Figure 8(b) shows that under the same network settings, STR and GDH also have similar results with increasing number of nodes and p value. While STR requires more messages than the other two schemes, SEGK has lower message cost than GDH during the group key updating process at the event of member addition.

Figure 7(c) shows message cost for updating group key at the event of member leave. Message cost increases as the number of nodes increases. The message cost is less than the cost in the group key initialisation process, but is much higher than the member addition process. For instance, when $n = 100$ and $p = 10\%$, the message cost is 75 compared with 407 and 37 in the group key initialisation and member addition scenarios, respectively. The message cost also increases with the p value. Figure 8(c) shows that under the same network settings, STR and GDH also have similar results with increasing number of nodes and p value. While STR requires more messages than the other two schemes, SEGK continues to have the least message cost.

4.4 Result summary

The results of the experiments are summarised below:

- With the increase of network nodes, the computation cost of SEGK scheme increases. Computation cost also increases with the increase of group membership percentage p. Computation cost increases more quickly for higher p value.
- Under the same network settings, SEGK has moderate computation cost that is higher than STR and lower than GDH for both group key initialisation and member leave scenarios. SEGK has the least computation cost for the member addition process among the three schemes.
- With the increase of network nodes, the message cost of SEGK increases. The message cost also increases with the increase of group membership percentage p.
- Under the same network settings, SEGK has moderate message cost, higher than GDH and lower than STR for the group key initialisation process. SEGK has the least message cost for member addition and leaving processes among the three schemes.

5 Conclusion

Key management for a large and dynamic group is a tense problem because of scalability and security. In this paper, we have propose an efficient group key management scheme for collaborative and group-oriented applications in MANETs. Our solution is based on n-party Diffie-Hellman protocol so that each member contributes a share of group key. The basic idea of our scheme is that a structure of double multicast trees is formed in MANETs for efficient dissemination of keying information. The purpose of constructing two trees in parallel is to achieve fault tolerance. We have also proposed two methods for detecting absent members. The first detection method is through tree links, which is appropriate for a network environment where node mobility is not significant. The second method is through periodic flooding of control messages, which is more appropriate for a highly dynamic network environment. In SEGK, there is one group member that acts as a group coordinator which computes and distributes the blinded intermediate keying information the group. Every member computes the group key in a distributed manner. To distribute the workload of group rekeying and maintenance, the role of group coordinator is rotated among all members. A new key tree structure is introduced in order to switch the group control role efficiently. A simulation study has been conducted to compare the message cost and computation cost under group key management schemes. Our future work will address multiple group member addition and leave scenarios.

References

Bae, S., Lee, S., Su, W. and Gerla, M. (2000) 'The design, implementation, and performance evaluation of the on-demand multicast routing protocol in multihop wireless networks', *IEEE Network*, Vol. 14, pp.70–77.

Burmester, M. and Desmedt, Y. (1994) 'A secure and efficient conference key distribution system', *Advances in Cryptology – EUROCRYPT '94*, No. 950.

Chiang, C., Gerla, M. and Zhang, L. (1998) 'Forwarding group multicast protocol (FGMP) for multihop mobile wireless networks', *Cluster Computing*, pp.187–196.

Deering, S.E. and Cheriton, D.R. (1990) 'Multicast routing in datagram internetworks and extended LANs', *ACM Transactions on Computer Systems*, Vol. 8, No. 2, pp.85–110.

Jaikaeo, C. and Shen, C. (2002) 'Adaptive backbone-based multicast for ad hoc networks', *Proc. IEEE International Conference on Communications (ICC 2002)*, Vol. 5, pp.3149–3155.

Ji, L. and Corson, M. (2001) 'Differential destination multicast – a MANET multicast routing protocol for small groups', *Proc. INFOCOM 2001*, pp.1192–1202.

Kim, Y., Perrig, A. and Tsudik, G. (2000) 'Simple and fault-tolerant key agreement for dynamic collaborative groups', *Proc. 7th ACM Conference on Computer and Communications Security*, ACM Press, pp.235–244.

Lee, S., Su, W. and Gerla, M. (2000) 'On-demand multicast routing protocol (ODMRP) for ad hoc networks', *Internet Draft*, draft-ietf-manet-odmrp-02.txt.

Rafaeli, S. and Hutchison, D. (2003) 'A survey of key management for secure group communication', *ACM Computing Surveys*, Vol. 35, No. 3, pp.309–329.

Royer, E. and Perkins, C. (1999) 'Multicast operation of the ad-hoc on-demand distance vector routing protocol', *Mobile Computing and Networking*, pp.207–218.

Sherman, A.T. and McGrew, D.A. (2003) 'Key establishment in large dynamic groups using one-way function trees', *IEEE Transactions on Software Engineering*, Vol. 29, No. 5, pp.444–458.

Steer, D., Strawczynski, L., Diffie, W. and Wiener, M. (1990) 'A secure audio teleconference system', *Advances in Cryptology – CRYPTO'88*, pp.520–528.

Steiner, M., Tsudik, G. and Waidner, M. (2000) 'Cliques: a new approach to group key agreement', *IEEE Transactions on Parallel and Distributed Systems, Proc. 18th International Conference on Distributed Computing Systems*, pp.380–387.

Wallner, D.M., Harder, E.J. and Agee, R.C. (1998) 'Key management for multicast: issues and architectures', *Internet Draft*, draft-wallner-key-arch-01.txt.

Wei, W. and Zakhor, A. (2004) 'Conectivity for multiple multicast trees schemes in ad hoc networks', *International Workshop on Wireless Ad Hoc Networks (IWWAN 2004)*, Oulu, Finland, pp.270–274.

Wong, C., Gouda, M. and Lam, S. (1998) 'Secure group communications using key graphs', *Proc. ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp.68–79.

Wu, B., Wu, J., Fernandez, E., Ilyas, M. and Magliveras, S. (2005) 'Secure and efficient key management in mobile ad hoc wireless networks', Appears in *Journal of Network and Computer Applications (JNCA)*, Vol. 30, pp.937–954.

Xie, J., Talpade, R., Mccauley, A. and Liu, M. (2002) 'AMRoute: ad hoc multicast routing protocol', *ACM Mobile Networks and Applicaions*, Vol. 7, No. 6, pp.429–439.