

# A Secure Group Key Management Scheme for Hierarchical Mobile Ad-hoc Networks

Dijiang Huang<sup>1</sup>, Deep Medhi<sup>2</sup>

<sup>1</sup> *Department of Computer Science & Engineering, Arizona State University*

<sup>2</sup> *Computer Science & Electrical Engineering, University of Missouri–Kansas City*

---

## Abstract

In this paper, we present a secure group key management scheme for hierarchical mobile ad-hoc networks. Our approach aims to improve both scalability and survivability of group key management for large-scale wireless ad-hoc networks. To achieve our goal, we propose the following approaches: (1) a multi-level security model, which follows a modified Bell-La Padula security model that is suitable in a hierarchical mobile ad-hoc networking environment, and (2) a decentralized group key management infrastructure to achieve such a multi-level security model. Our approaches reduce the key management overhead and improve resilience to any single point failure problem. In addition, we have developed a roaming protocol that is able to provide secure group communication involving group members from different groups without requiring new keys; an advantage of this protocol is that it is able to provide continuous group communication even when the group manager fails.

*Key words:* Security, Group key management, Hierarchy, Ad-hoc Network, Many-to-many group communication.

---

## 1 Introduction

A *Hierarchical Mobile Ad-hoc Network* (HMANET) architecture is formed by multiple groups in a hierarchical network structure in which each group consists of multiple mobile nodes. The HMANET architecture has been extensively studied [1–3]. In this paper, we focus on the security aspect of the HMANET architecture, especially on developing an efficient secure group key management scheme. It may be noted that wireless mobile devices usually have limited energy capacity, which restricts their communication abilities [4]. Thus, compared to the fixed network infrastructure, an efficient group keying scheme in an HMANET that can address reduction of both communication

and computation overhead is desirable. A critical issue in key management in the HMANET framework is whether a scheme allows movement of mobile nodes from one group to another without unduly increasing overhead, which impacts power consumption cost. Furthermore, in a real-time application environment, a device that is visiting another group should be able to communicate without unduly incurring heavy key establishment cost.

In this paper, we focus on a HMANET environment in which devices (i.e., users with devices) may frequently move from one group to another. We have developed a keying scheme that allows such movement without requiring a new key generation and establishment for each communication session. To reduce both communication and computation overhead of group key management schemes for HMANET, we present a shared-key-based group-key management scheme in which the group key management includes two phases: 1) *a key predistribution phase* and 2) *a group communication phase*. In order to discuss these phases, we first introduce a few terminology: an *add* means a new user becomes a member of an existing group; this is distinguished from a *join* when a group member joins an existing group/subgroup communication session. When a member leaves a group permanently, we refer to it as a *delete* or *hard eviction*; if a member leaves an existing group communication session, it is referred to as a *leave* or *soft eviction*.

It is assumed that each group forms initially with a stable group population for which a key predistribution phase is initially invoked. In each such group, a group manager is assumed to be in charge of the key distribution for its group. If a new user is added to an existing group, then a new key distribution is invoked; this is not to be confused with a user from another group visiting an existing group. To avoid frequent key predistribution, a group can be formed that allows some empty slots. For instance, suppose a group is of size 50, but is anticipated to grow to 60 soon; in this case, the group can be started with 60 slots in our group keying scheme, so that the empty slots can be filled with if a new user is added to the group, yet not requiring another key distribution unless the group size actually goes over 60. It should be noted that key predistribution is not required for the following three cases: 1) a subset of existing group members need to communicate impromptu for a secure subgroup communication session, 2) a member is to be excluded from an existing communication session, and 3) a member is involved in multiple subgroup communications simultaneously (subgroups can be overlapped). This three scenarios fall under the communication phase and operations involved fall under join or leave, as defined earlier. Furthermore, our keying scheme has the property that a user can be in multiple sub-group communications sessions without requires news keys.

It may be noted that a hierarchical group structure improves the scalability of key management for large-scale heterogenous MANETs. The drawbacks of

a hierarchical structure are: 1) it reduces the flexibility of the group formations, and 2) it requires an additional mechanism to handle roaming of group members. For example, a mobile user may travel to other groups and might not be able to set up communication back to its original group members, or if the group manager fails, then a new group manager needs to be assigned either by the higher level nodes or elected by other group members. To solve such issues, we present a source-routing-based roaming protocol which utilizes a computational efficient multicast-tree encoding scheme using Prüfer sequence [5].

We next comment on security policy. The Bell-La Padula (BLP) confidentiality security model provides a basis for us to build a security model for HMANET [6]. Briefly, the BLP model describes a secure computer system abstractly, without regard to the system's applications. Our hierarchical structure follows a modified Bell-La Padula confidentiality security policy model to accommodate a distributed hierarchical environment; in this model, the key derivation relation is downward in which the higher level group members can decrypt the lower levels' ciphertext. We address applicability of the BLP model from its abstract form to the secure group key management framework in an HMANET environment.

### 1.1 Related Work

Several secure group communication key management schemes have been presented in the past decade. The previous work can be broadly classified into distributed schemes and centralized schemes [7]. The most well-known distributed scheme is the Group Diffie-Hellman (GDH) method [8–10]. This approach requires a linear number of expensive public-key operations, while there have been efforts to reduce the number of public-key operations [8,11]. However, due to the computational overhead imposed by public-key operations, each user needs to negotiate with its communication peers to maintain the communication group. In general, this class of schemes is not suitable for delay sensitive and real-time interactive applications.

Centralized secure group communication keying schemes (which are all non-public key solutions) have been approached by two different research communities: one from the information theory community, and the other from the Internet community. In the approach that stems from the information theory community, the *key pre-distribution scheme* (KPS) plays a critical role [12,13]. KPS requires a *trust authority* to distribute secret information before group communication; then during group communication, only privileged subsets (pre-specified) of participants are able to compute certain keys. For instance, the *Broadcast Encryption Scheme* (BES) [14] consists of a key pre-distribution

phase, followed by a broadcast message which is to be decrypted only by a privileged subset (pre-specified) of participants. In the approach that stems from the Internet community, the *framework oriented schemes* (FOS) [15–18] is a popular scheme, which uses hierarchical group relations to set up group keys. FOS can be either a single flat group under one management center or multi-level groups with multiple group management centers. For instance, the *Key oriented schemes* (KOS) use key derivation relations to build up the keying scheme. Group members use their secrets to generate the desired group key (See [19–22]). Centralized scheme requires the key server to be always online to maintain the group changes. In our previous work, we have shown that the group management overhead is prohibitively high if the group member join or leave during the communication phase is frequent [23].

Approaches such as key predistribution and pairwise key establishment schemes (e.g., [24,25]) have been proposed in recent years for sensor networks. A set of secrets are preinstalled in each sensor before deployment. After they are deployed, each sensor can set up pairwise keys with its neighboring nodes. Although the key distribution mechanism is somewhat similar to our approach, the goal is different. We target to set up all possible subgroup keys instead of pairwise keys such that during the communication phase there is no key verification cost—a desirable property for real-time interactive applications.

Basagni et al [26] proposed a shared key based solution for key management in large scale ad hoc networks. Their approach assumes that each mobile node is a good node and behaves properly. By combining mobility-adaptive clustering and an effective probabilistic selection of the key-generating node, a shared key is periodically updated among all mobile nodes.

Rhee et al. [27] have presented a secure group key management architecture in HMANET environment. They create two levels of groups: a cell group located at the bottom of the hierarchy uses centralized group key management, and a control group located on top of cell groups uses distributed group key management. In each cell group, the group key management is managed by the group manager, which is not suitable for frequent subgroup formation/deletion and delay sensitive applications. In addition, the group member migration will invoke heavy public key operations.

In current research literature, there are no security models for the HMANET. In a hierarchical structure, the Bell-La Padula (BLP) security model [28,29] is a possible choice. However, the BLP security model was originally designed for computer systems, not distributed hierarchical networks. Although, the BLP model has been extended to traditional multi-level secure message systems [30], its suitability to the HMANET has not been addressed so far.

## 1.2 Contributions

Our work focuses on the following directions in a HMANET framework: 1) we present a multi-level security model, 2) we have developed a hierarchical secure group keying scheme to decentralize the group key management overhead to multiple group managers, and 3) we present an inter-group roaming protocol for inter-group key management to improve the survivability when group managers fail or members from one group roam in another group. Specifically, we present a unique solution by integrating the group keying and group management within a hierarchical networking framework that allows scalability of group key management.

We also discuss how our approach is applicable to the BLP confidentiality security model. In each group, the group keying scheme follows the same group formation structure. As a result, the group keying scheme can be applied to every group in the multi-level hierarchical group structure. A second advantage is that due to the decentralized group management structure in HMANET, our approach is resilient to any single point failure problem. In addition, our roaming protocol is able to provide secure group communication for group members in different groups and it is able to provide continuous group communication when a group manager fails. We present performance assessment of our scheme to support our claims.

## 1.3 Organization

The rest of the paper is organized as follows: In Section 2.1, the HMANET architecture is presented along with the security model and the attack model. In Section 3, the hierarchical group key management keying scheme and the roaming protocol are described in details. The performance issues are given in Section 4. Finally, we summarize our work in Section 5.

# 2 HMANET: Architecture, Security Model, and Attack Model

## 2.1 Overall Architecture

We first describe the hierarchical mobile ad-hoc network architecture. It consists of three levels: root level, mobile backbone network level, and mobile user network level; in each level, there can be multiple nodes. Thus, we can envision a root network followed by multiple mobile backbone networks (MBNs), which

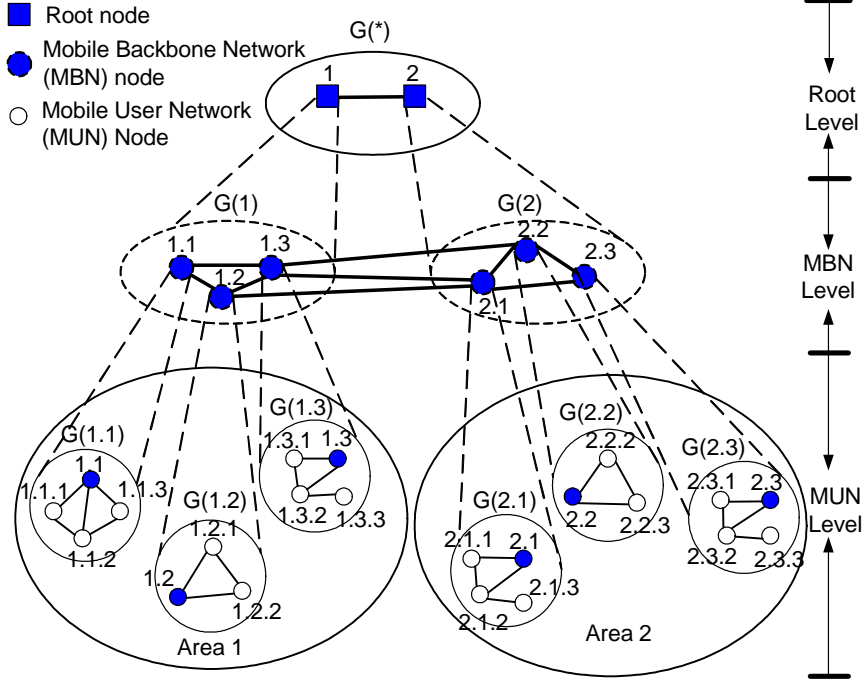


Fig. 1. Group architecture of hierarchical Ad-hoc networks.

is associated with a node in the root network; in turn, a mobile user network (MUN) is associated with each MBN node. For example, MBNs can be based on unmanned ariel vehicles (UAVs). The general architecture is depicted in Fig. 1. In order to identify the entities and group structure of the HMANET, we use a *dot notation*. A *dot* separates two neighboring levels, the higher level on the left and the lower level on the right. A notation always starts from the root level. For example, as shown in Fig. 1, we use  $n(i) = n_1.n_2.i$  to represent an MUN node in a three-level hierarchy, where  $n = n_1.n_2$  can be used to identify the group where the node  $i$  locates.

Root level. At the top of the hierarchy, a root node is associated with an MBN. The root node can provide the shared beam to its MBN nodes to maintain connectivity for one area of operations in lower levels. In Fig. 1, the root-node group is represented as  $G(*)$ ; a root node has the highest computation and communication power among all mobile nodes in the system; we assume that the root node cannot be easily compromised. Due to the highest security level of root-level nodes, encrypted messages sent from lower-level nodes can be decrypted by the root level nodes, and messages encrypted and sent by a root level node cannot be decrypted by lower-level nodes. In our multi-level security model, which is discussed later, designated nodes at each level are assumed to be trusted entities at their levels.

MBN level. Multiple MBN nodes form a control group. We assume that MBN nodes have more communication and computation power than MUN nodes. Each MBN node can set up a direct wireless link with the root level nodes.

Multiple MUN nodes and at least one MBN node form a cell group in the hierarchical group structure where the MBN node acts as the group manager for a set of MUN nodes. In Fig. 1, the control group is represented as  $G(n)$ , where  $n$  is the group number.

*MUN level.* MUN nodes are mobile devices that have limited computation and communication capabilities. The communication among MUN nodes can be set up directly if they are located within each other's communication range. If an MUN node cannot set up the wireless links directly to its group members, it can set up multi-hop wireless connections via its group manager.

It may be noted that a member of an MUN might move or travel to another MUN or MBN. This member must be able to still communicate with its subgroup in its native network on a real-time basis without authenticating every time. Thus, an inter-group secure group management scheme is necessary that allows the roaming capability.

## 2.2 HMANET Security Model

An important issue for the HMANET is the security model. This, in turn, is related to the secure group communication. In this section, we present the security model that extends Bell-La Padula security model for HMANET. We further describe how security is enforced. We then present the attack model that can be addressed by our secure group scheme, followed by how roaming is architected.

### 2.2.1 Multi-level Security Policies

In the hierarchical structure presented in Fig. 2.1, the key derivation relation is downward and thus the higher level group members can decrypt the lower levels' ciphertext. We define *subjects* as mobile users and *objects* as data (or messages). Compared to multi-level security model [28], we map the rights *decrypt* to *read*, *encrypt* to *write*, and *key generate* (for group members) to *execute*. We can define the *dom* (dominates) as follows:

#### Definition 1

- (1) *The security level  $(L(u_i), P(i))$  dominates the security level  $(L(u_j), P(j))$  if and only if  $L(u_j) \leq L(u_i)$  and  $P(i) \Rightarrow P(j)$ .*
- (2) *The security level  $(L(u_i), P(i))$  dominates the security level  $(L(C), K)$  if and only if  $L(C) \leq L(u_i)$  and  $P(i) \Rightarrow K$ .*

In the definition 1-(1), security levels of  $u_i$  and  $u_j$  can be represented by dot notations  $n(i)$  and  $n(j)$ , respectively; we define  $n(i) \text{ dom } n(j)$  if and only if

$n(i) \subset n(j)$  (e.g.,  $n(i) = n_1.n_2$  and  $n(j) = n_1.n_2.n_3$ ).  $P(i) \Rightarrow P(j)$  represents that the user  $u_i$  can generate  $P(j)$  from his/her preinstalled secrets set  $P(i)$ . The dot notation is defined similar to the compartment or categorization of a subject or an object in BLP model. In the definition 1-(2),  $L(u_i)$  is the ciphertext receiver's group level (is represented as the dot notation  $n(i)$ );  $L(C)$  is the encryption level of the ciphertext by using the set of keys  $K$ ;  $P(i) \Rightarrow K$  represents that the user  $u_i$  can derive at least one key in  $K$  from his/her preinstalled secrets set  $P(i)$ , where the  $C$  is encrypted by the  $keys \in K$ .

The least upper bound (LUB) of the HMANET is the root node (denoted by  $u_0$ ) of the hierarchical structure, i.e.,  $(L(u_0), P(0))$  dominates  $(L(u_i), P(i))$  and  $(L(C), K)$  for any user  $u_i$ , ciphertext  $C$ , and key set  $K$ . Due to the decentralized hierarchical structure of HMANET, the greatest lower bound (GLB) does not have meaning when two users are located in different groups. This is because they cannot derive a shared key without the help by each other's group manager to decrypt a ciphertext. Within a group, the GLB is the ciphertext encrypted by the group key  $(L(C), K(G))$ , where  $G$  is the group and  $\forall u_i \in G$ ,  $(L(u_i), P(i))$  dominates the security level  $(L(C), K(G))$ .

The definition 1-(1) of *dom* tells us if a user *dom* another group member, he/she must be able to derive all the secrets possessed by that group member; the definition 1-(2) of *dom* tells us if a user *dom* a ciphertext, he/she must be able to derive at least one encryption key used to encrypt the ciphertext. Then, we can have the following *simple security conditions*:

- **Simple Security Conditions:**

- (1)  $u_i$  can decrypt ciphertext  $C$  if and only if  $u_i$  *dom*  $C$  and  $u_i$  has discretionary *decrypt* access to  $C$ .
- (2)  $u_i$  can distribute secrets set  $P(j)$  to  $u_j$  if and only if  $u_i$  *dom*  $u_j$  and  $u_i$  has discretionary *key distribute* access to  $P(j)$ .

Based on the above described hierarchical encryption policy, it is easy to see that the trust relations of the security model are from the bottom level to the top level. As shown in Fig. 2.1, the root level nodes are the group managers of control groups. They are responsible for distributing keys to MBN nodes in the MBN level and they can also derive the group/subgroup keys used in the MUN level. An MBN node is the group manager of a cell group, it is responsible for distributing keys to MUN nodes in its cell group. A group manager hosts a cluster rooted by itself, however it cannot derive secrets used in other clusters. This means that any group member dominates the lower level group members within the same cluster. For example, as shown in Fig. 1, node 1 creates a cluster rooted from itself. Node 1 can derive keys used in groups  $G(1)$ ,  $G(1.1)$ ,  $G(1.2)$ , and  $G(1.3)$ , but it cannot derive keys used in groups  $G(2)$ ,  $G(2.1)$ ,  $G(2.2)$ , and  $G(2.3)$ .

Compared to the *write* policy of BLP security model [29], we define the following property to prevent a higher level from encrypting and sending data to a lower level.

- **\*-Property (Star Property):**  $u_i$  can encrypt plaintext  $T$  by using  $P(i) \Rightarrow K$  if and only if  $L(u_i) \leq L(T)$  and  $P(j) \not\Rightarrow K$  when  $L(P(j)) < L(P(i))$ , and  $u_i$  has discretionary *encrypt* access to  $T$ .

### 2.2.2 Security Enforcements

Our multi-level security model is composed of multiple clusters rooted by each root level nodes. The clustering structure regulates the key derivative relations following the hierarchical framework of an HMANET. To enforce the simple security conditions, due to the hierarchical key derivative relations, our secure keying scheme allows a higher level node  $u_i$  to decrypt the ciphertext  $C$  sent from a lower level node  $u_j$  when  $(L(u_i), P(i)) \text{dom}(L(C), K)$  and  $(L(u_i), P(i)) \text{dom}(L(u_j), P(j))$ . The dominating scope of the node  $u_i$  is the cluster structure rooted from  $u_i$  (see Fig. 2.1).

The star property mentioned earlier prevents the higher level group members to *encrypt* data using lower-level group keys. We solve this problem by introducing trusted entities (i.e., the group managers or leaders). Thus, to enable a higher level group member to encrypt down, we rely on the group manager who distributes keys to his/her immediate lower-level group members. For example, as shown in Fig. 2.1, if node 1.1 wants to send an encrypted message to node 1.2.1, the encrypted message (using the same level group key) must be checked by the group manager 1.2 for sanitization, and then 1.2 re-encrypts the message using the same level group key and sends it to 1.2.1. Due to the cluster structure, 1.1 cannot derive the shared key used by node 1.2.1.

There are two situations in which the right to *encrypt* is possible following the star property:

- If  $L(u_i) = L(T)$ ,  $u_i$  can use any subgroup key to encrypt the message  $T$  within the same security level.
- If  $L(u_i) < L(T)$ ,  $u_i$  must use key  $k_{i_0}$  to encrypt the message  $T$ .

Here, key  $k_{i_0}$  is the  $i$ -th initial key element to derive a sequence secrets (see Section 3.1 for the construction of group keys); it can be derived only by the group manager (i.e., the trust entity), the group manager can use a higher level key to re-encrypt the message. In this way, we can allow a lower level group member to encrypt the message at a higher level. Note that in an HMANET environment, the message originator must specify the security level of the message. This can be achieved by specifying the destination node *id*  $n(i)$ , where the group number  $n$  is the security level (i.e., the group *id*) of the

destination node  $u_i$ .

### 2.3 Attack Model

We next consider the attack model for which we assume the following: (1) lower level group members are more prone to be captured than higher group members; (2) the higher level communication equipment are tamper-proof devices. The adversaries cannot easily derive the set of secrets preinstalled in the devices. Thus, the adversaries can be either compromised devices at the lower levels or malicious higher-level members. The goal of the adversaries is to breach the security model presented in Section 2.2.

For intra-group key management, the adversaries can share their pre-distributed keys and try to compromise the group keys used by other subgroups that do not include the adversaries. We refer to this attack as a *colluding attack*. To counter a *colluding attack*, the deployed intra-group keying scheme should provide the security feature such that a collection of compromised nodes will gain *zero* information on uncompromised keys used by other subgroups. Such a non-colluding property is achieved through our previous solutions proposed in [23].

For inter-group key management, the attacks are HMANET routing attacks [31]. For example, the adversaries can: (i) spoof, alter, or replay messages, (ii) selectively forward messages, (iii) deploy Sinkhole and Wormhole attacks, (iv) deploy Sybil attacks, etc. In our attack model, these attacks aim to compromise our security model and they can be classified into three types: DoS attacks, message disclosure attacks, and session-hijacking attacks. Our group key management scheme and roaming protocol are vulnerable to DoS attacks. Additional security framework to counter the DoS attacks should be deployed, which is not our research focus in this paper. To prevent the message from being disclosed and prevent session-hijacking attacks, end-to-end encryption and authentication must be provided.

### 2.4 Roaming Among Groups

In HMANET, a flat group is originally assigned to a group member as the group member's *home group*; then, others are *host groups* (or *roaming groups*). A mobile node may move out of the communication ranges from its home group members; the group manager of a group may fail due to software or hardware failure. In these scenarios, roaming protocols are used to maintain the connections among group members. For example, as shown in Fig. 2 scenario 1, node 1.2.1 moves out of the communication range of cell group  $G(1.2)$  and it is

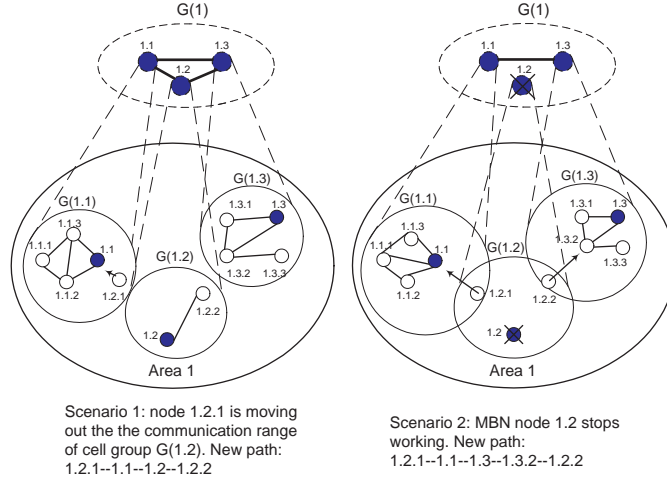


Fig. 2. Roaming among groups.

within the communication range of cell group  $G(1.1)$ . Mobile node 1.2.1 needs to set up secure tunnels to its home group  $G(1.2)$  via MBN node 1.1. One possible secure tunnel is 1.2.1--1.1--1.2--1.2.2. In scenario 2, if MBN node 1.2 fails and group members cannot set up direct wireless links, a secure tunnel 1.2.1--1.1--1.3--1.3.2--1.2.2 can be set up.

In order to support secure and seamless roaming that involves minimal connection setup delay, a roaming node is required to set up a secure key with roaming group members before the roaming begins. This requires the MBN nodes to maintain the network topology formed by control group nodes in realtime. To this end, we can use topology-aware ad-hoc routing protocols, such as TBRPF [32] or protocols based on positioning system enabled devices (e.g., GPS devices), such as [33]. Based on the topology information of MBN nodes in the system, each MBN node maintains a topology graph formed by all MBN nodes and a neighbor list of the topology graph; e.g., Fig. 3 shows that the neighbor list of node 1.5 is  $\{1.3, 1.4, 1.7, 1.8\}$ . To support roaming among groups, each cell group  $G(m.n)$  maintains a roaming key denoted by  $k_v|_{G(m.n)}$ . Due to the multi-level security model, we require the end-to-end en-

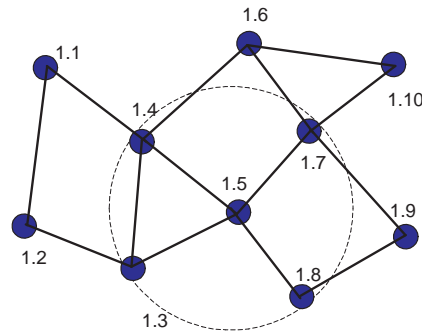


Fig. 3. Network topology of control group and neighbor list.

encryption during roaming; the intermediate nodes can only help the roaming nodes to set up security tunnels to their home group. Hence, a roaming key is only used to prevent outsiders from disclosing the transferred ciphertexts. A group manager (MBN node) exchanges its cell group roaming keys with other group managers in its neighbor list. When a group manager receives a neighbor group’s roaming key, it distributes the neighbor group roaming key to its cell group members. Thus, when a host group’s member moves to a neighboring group, it can use the roaming key to set up secure connections with its host group members. In this way, we can use this proactive key agreement method to reduce the key establishment time due to roaming by group members.

### 3 Secure Group Key Management for HMANET

In this section, we present our secure group key management scheme for HMANET. This approach is built on our previous keying scheme for a flat architecture [23]. Briefly, our extended scheme is a multi-level, multi-group keying structure. In this scheme, each group member maintains the same number of preinstalled keys and each group member can serve as the group manager for a lower level group. We present two roaming protocols to solve the roaming issues in a hierarchical secure multi-group communication environment. We first briefly summarize a few details of the keying scheme for a flat architecture [23] that is relevant for the describing the proposed new scheme for HMANET.

#### 3.1 Secure Group Keying Scheme for a Flat Architecture: An Overview

This scheme is for a single group with a stable population where a subset of group members want to set up secure group/sub-group communication services, without having the group key establishment for each instantiation of a communication—thus, the scheme is suitable for real-time services, saving on set-up delay. The scheme is based on two phases: a centralized *key pre-distribution phase* and a distributed *group communication phase*. In the *key pre-distribution phase*, a key distribute center (*KDC*) distributes the keys (or secrets) and index  $\langle \text{key } ID, \text{ user } ID \rangle$  to each group member, for example, via offline methods. During the *group communication phase*, a group member can derive subgroup communication keys from its pre-distributed keys *without* relying on *KDC* without incurring set-up delay.

In order to reduce the storage overhead of a member, we build a key-forest structure via multiple key chains. Each “tree” (a linear key chain) in the key-forest structure represents a particular key derivative relations among all

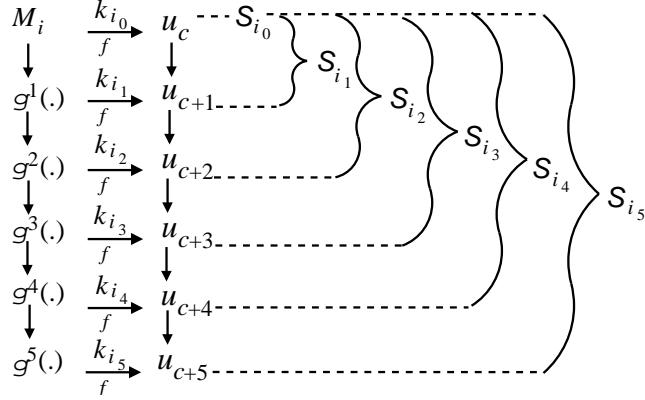


Fig. 4. An example of one-way function chain.

group members. A one-way function is used to construct a one-way function chain, and subsequently, to form a key chain. A *one-way function chain* (OFC) is a sequence of values with linear derivation relations among them, see Fig. 4. For example, we use  $g^j(\cdot)$  to denote using  $g(\cdot)$  on a message  $j$  times, i.e.,  $g^j(\cdot)$  can derive  $g^r(\cdot)$  when  $j < r$ .

Now set  $k_{i_0} = M_i$  to represent the  $i^{\text{th}}$  initial key element to derive a sequence of secrets. For brevity, we do not differentiate a secret and the corresponding key. A key is represented as  $k_{i_j} = f(g^j(M_i))$  and  $f$  is a publicly known key generating function. Thus based on the derivative relations of an OFC, we have the following key derivate relations (i.e., a key chain):

$$k_{i_0} \rightarrow k_{i_1} \cdots \rightarrow k_{i_j} \rightarrow \cdots \rightarrow k_{i_r}, \quad \text{where } 1 < j < r. \quad (1)$$

The derivative relation of a key chain is from the head ( $k_{i_0}$ ) to the tail ( $k_{i_r}$ ) where  $i_j$  can uniquely identify a subgroup composition. A keying example is shown in Fig. 4, which is composed of 6 keys ( $r = 5$ ). If we allocate a key to each group member from  $u_c$  to  $u_{c+5}$ , each group member can derive its lower-level keys. Then 6 subgroups can be formed via the key chain, denoted by  $\mathcal{S}_{i_0}$  to  $\mathcal{S}_{i_5}$ . Note that these subgroups are *non-colluding* subgroups, in which a subset of members cannot collude to derive the keys used by a given subgroup unless at least one of colluding members belongs to that subgroup. It can be shown that if there are  $n$  group members, we can systematically distribute a unique subset of key elements from  $\frac{n(n-1)}{2}$  key chains. Using the predistributed key elements, each group member can derive a sub set of keys (denoted as set  $K$ ) to cover all possible subgroup compositions. On average, the size of  $K$  is in the order of a sub-linear function of the subgroup size [23].

We utilize the following abstract presentations for group keying relations. We define a user's identifier as  $u_i = n(i)$ , where  $n(i)$  is the dot notation of the group member identifier,  $1 \leq i \leq G$  and  $G = |G(n)|$  is the flat group size (we use *flat group* and *group* interchangeably). The group member  $u_i$  holds a set

of predistributed secrets represented as  $P(i)$ . For a subgroup key, we present the following definition:

**Definition 2** *The set of subgroup keys shared between user  $u_i$  and the subgroup members in  $S = \{u_1, \dots, u_s\}$ , where  $|S| \geq 2$  and  $u_i \in S$ , is represented as  $K(n(i), S) = \{k | \forall j, \{P(i) \rightarrow k\} \cap \{P(j) \rightarrow k\} \neq \emptyset, j \neq i, \text{ and } n(j) \in S\}$ .*

Note that the set of shared keys can be derived from both  $P(i)$  and each secret set in  $\{P(1), \dots, P(s)\} \setminus P(i)$ . In particular,  $K(n(i), S)$  represents the minimum number of keys that can be derived by  $u_i$  to cover all subgroup members in  $S$ .

During the secure group communication phase, a session key  $k_s$  is used to encrypt data and  $K(n(i), S) = \{k_{i_j}\}$  are used as *key-encrypting key (KEK)* to fulfil desired subgroup composition where  $i_j$  is used to identify the key element location in a key chain, see (1). In this way,  $n$ ,  $i$ , and set  $\{i_j | \forall j \in S \setminus j\}$  can uniquely identify a subgroup composition and the corresponding subgroup keys. Using the encrypting function  $E(\cdot)$ , we can encrypt data as follows:

$$\langle \{ \{i_j | \forall j \in S \setminus j\}, E_{k_{i_j}}(k_s) \}, E_{k_s}(Data) \rangle, \quad (2)$$

where  $i_j$  is a key identifier, see (1), and  $k_s$  is a session key. After receiving the message, a receiver first checks the list  $\{i_j | \forall j \in S \setminus j\}$  to see if he/she belongs to the subgroup. If he/she is involved in the subgroup, the receiver can derive a proper subgroup key to decrypt the session key and then decrypt the message.

We next discuss suitability of this scheme for group key management. Subgroup formation and deletion are the main group management operations. For example, as shown in Fig. 5, the directed arrow at the lower portion of the figure represents the subgroup key derivative relations using the one-way function. This approach has two primary benefits. First, a group member can generate the group/subgroup keys without the help from its group manager. Thus, once the key sets are distributed, the group/subgroup key negotiation is unnecessary. This feature is crucial for delay sensitive applications that require minimal group key setup delay. Second, the group key revocation is simple and no additional group management overhead is required. This is because revoking a group member is equivalent to generating a different subgroup key.

### 3.2 Multi-level Multi-group Secure Group Keying Scheme

We are now ready to describe our extended scheme. First, note that a *hierarchical group keying structure* can be built using a one-way function. The

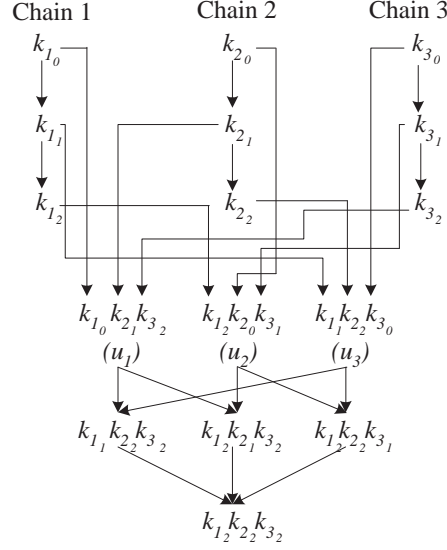


Fig. 5. An example of key distribution for three group members within a flat group.

relation among different levels in the structure is in a top-down fashion. We utilize the derivative relations of one-way function chains to build the hierarchical structure. In this way, the size of the pre-installed key set for each node in the HMANET remains the same.

In order to explain our new scheme, we first present a simple example to illustrate the group key distribution scheme for a 3-member flat group composed of  $u_1$ ,  $u_2$ , and  $u_3$  (see Fig. 5). In this example, three keys are distributed to each group member from three different key chains. Any two of the group members can derive corresponding 2-member subgroup keys without the colluding problem, as described in [23].

In an HMANET, each mobile node can create its own cluster and serves as a group manager for a lower level group. The mobile node uses its pre-distributed secrets (or keys) to generate lower level group members' secrets (or keys). Before generating a key set for a lower group member, the group manager needs to regenerate its key set. We accomplish this step using a shuffle algorithm, which is presented as Algorithm 1. Using the shuffle algorithm, the key distribution as shown in Fig. 5 for flat architecture is essentially shuffled.

### Algorithm 1 (Shuffle algorithm)

*Input:* for group manager  $u_i$ ,  
 $\{k\}_{u_i} = \{\mathbf{k}_1, \dots, k_{i_0}, \dots, \mathbf{k}_n\}$ , where  $1 \leq i \leq n$ .  
*Output:*  $\{k'\}_{u_i} = \{k'_{1_0}, \dots, k'_{i_0}, \dots, k'_{n_0}\}$

- 1 : Begin
- 2 :  $j = 1$ ;

```

3 : Loop    $k'_{j_0} = g(k_{i_0}, \mathbf{k}_j);$ 
4 :         if ( $j == n$ ) goto End;
5 :          $j = j + 1;$ 
6 :         goto Loop;
7 : End

```

\* Note: the subscript in  $\mathbf{k}_j$  represents the position in key set  $\{k\}_{u_i}$  possessed by user  $u_i$ ; it is not used to identify the key chain position described in (1).  $\square$

After shuffling the original key set  $\{k\}_{u_i}$ , the group manager  $u_i$  derives a new key set  $\{k'\}_{u_i}$ . For example, before shuffling,  $u_1$ 's original key set is  $\{k\}_{u_1} = \{k_{1_0}, k_{2_1}, k_{3_2}\}$ . After shuffling,  $u_1$  has the key set  $\{k'\}_{u_1} = \{g(k_{1_0}, k_{1_0}), g(k_{1_0}, k_{2_1}), g(k_{1_0}, k_{3_2})\}$ , as shown in Fig. 6, where  $g$  is a publicly known one-way function. Within the home group,  $k_{1_0}$  is unique for  $u_1$  and we call  $k_{1_0}$  as the *key seed* of  $u_1$ . In this way, the group manager  $u_1$  still uses the key set  $\{k\}_{u_1}$  to communicate the mobile nodes ( $u_2$  and  $u_3$ ) within its home group, while it uses the key set  $\{k'\}_{u_1}$  to generate the key sets for its controlled group members  $\{u'_1, u'_2, u'_3\}$  in the lower level. Thus, lower-level group members can only collude to derive the key set  $\{k'\}_{u_i}$  but not the key set  $\{k\}_{u_i}$ . It may be noted that the changes of keys in a group manager's key set  $\{k\}_{u_i}$  affects only a limited number of keys. For example, if the key set possessed by  $u_3$  is compromised by attackers, they cannot derive the key  $k'_{3_0}$ . This is because only  $u_1$  knows  $k_{1_0}$  and the function  $g(k_{1_0}, k_{3_2})$  prevents attackers from deriving the key  $k'_{3_0}$ .

There are three important properties of the shuffling algorithm: 1) the shuffle algorithm prevents group members from deriving group manager's keys, 2) shuffling the group manager's key set can present "1 affects n" problem, i.e., it avoids requiring all group members key sets to be changed if one member's key set needs to be changed, 3) it allows to create the clustering hierarchical structure shown in Fig. 1, which follows the multi-level security model presented later.

Since we want the ability of members of a group to roam to other groups, and still be able to do secure group/sub-group communication, it is important that the keying scheme allows that. This primarily depends on 1) how roaming keys are distributed and 2) how group communication sessions are set up among host groups and from a host group to a node's home group. These are addressed next.

### 3.2.1 Distributing Roaming Keys

The roaming key for group  $i$  is represented as  $k_v^i$ . It has two restrictions: (i) all group members in group  $i$  can derive the key  $k_v^i$  easily; (ii) a roaming node cannot derive extra key information of group  $i$  from the roaming key. As shown in the example presented in Fig. 5, each group member can derive the group

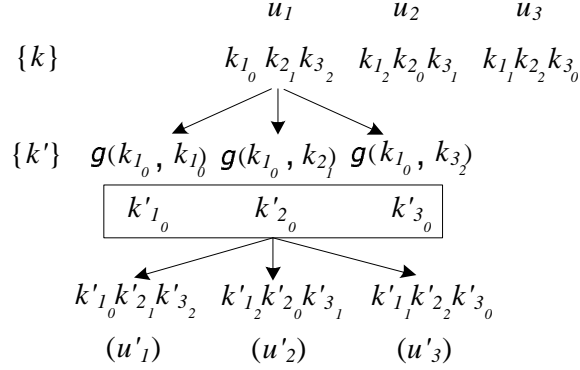


Fig. 6. An example of multi-level group key distribution for three group members. The lower level groups managed by  $u_2$  and  $u_3$  are omitted.

key  $k^i = g(k_{1_2}, k_{2_2}, k_{3_2})$  in group  $i$ . We use one-way function  $g$  to generate the roaming key for the group  $i$ :  $k_v^i = g(k^i)$ . Thus, all group members in group  $i$  can derive the roaming key  $k_v^i$  and a host node cannot derive any group key information of group  $i$  from the roaming key  $k_v^i$ .

### 3.2.2 Roaming to Home Group

In Section 2.4, we discussed the scenario where a mobile node roams among groups. One of the important research goals is to be able to set up a shared key between the roaming node and host group manager with minimal key setup delay (note that the shared key is only shared between the node, roaming group manager, and home group manager). The established shared key can be used to set up secure tunnels from the roaming node to the home group manager. In order to provide roaming ability for group members, we present a roaming protocol as summarized in Table 1.

Briefly, in our roaming protocol, time stamps ( $N_1$ ,  $N_2$ , and  $N_3$ ) are used to guarantee that the message is new; here function  $I$  is an incremental function, e.g., “add 1”. In step ①, roaming node  $i$  sends a request to host group manager  $j$ . Then the request is forwarded to home group manager  $l$  in step ②. After verifying the request, the home group manager picks session key  $k_s$  and encrypts the session key by two pairwise keys:  $k_{lj}$  is shared between two group managers and  $k_{li}$  is shared between the roaming node and the home group manager. The response from the home group manager is sent back to the host group manager in step ③. In step ④, the host group manager forwards the encrypted session key  $k_s$  to the roaming node. Note that if a roaming node does not have a direct connection with the host group manager, in step ④, the host group manager needs to distribute session key  $k_s$  to its group members that are intermediate nodes in the setup tunnel. If the host group manager does not have a direct connection with the home group manager, the messages in step ② and ③ should also be encrypted/decrypted by the pairwise keys on

each hop from the source group to the destination group.

### 3.2.3 Roaming Among Host Groups

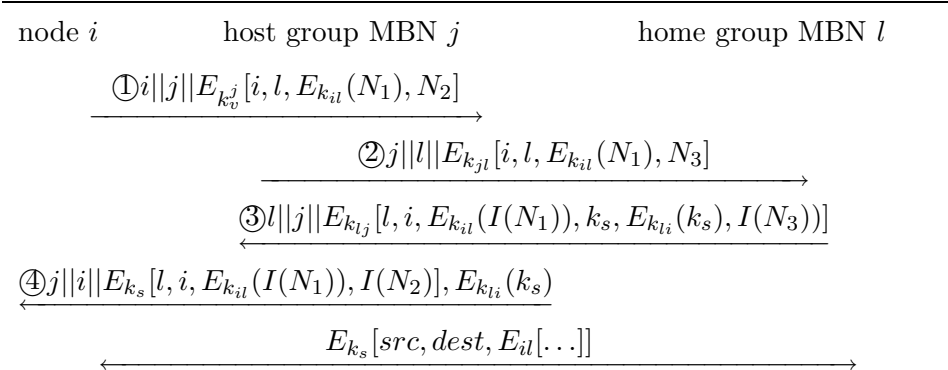
In an HMANET, the MBN node may fail. In order to maintain the group communication, each MUN node needs to be registered at the host group manager and depends on the host group to connect to the rest of its home group members.

As described in Section 2.4, we assume that an MBN node is running either a topology-aware ad-hoc routing protocol or a positioning system enabled device. Each MBN node knows the network topology at the MBN level. When an MBN node fails, the rest of the MBN nodes will notice it immediately and they are ready to accept the requests from MUN nodes in the failure group. When an MUN node approaches a host group, it sends a request to the host group manager. The request is protected via the host group's roaming key previously distributed via the MUN node's home group manager. After a roaming node sets up a secure channel to the host group manager, the host group manager propagates the MUN node's information to other MBN nodes. After all MUN nodes have established secure channels in their host groups, each host group manager creates a reachable member list based on the information received from other MBN nodes and then forwards the list to the MUN nodes in its group.

When a roaming node wants to communicate to a subgroup of its home group members, it creates the message based on the encryption message presented in (2) and sends the message to the host group manager by encrypting the whole message via the group roaming key.

Table 1

Roaming Protocol between roaming node and its home group manager



$N_1, N_2, N_3$ : time stamps;  $node_{\#}$ : node  $id$ ;  $i, j, l$ : group  $id$ ;  $k_s$ : session key

$f()$ : incremental function;  $E()$ : encryption algorithm;  $k_{il} = k_{li}$ ,  $k_{jl} = k_{lj}$

$k_v^j$ : roaming key of group  $j$

Home group members may be located within multiple host groups. In order to forward the message to proper host group managers, a communication efficient multicast roaming protocol is required. We present a multicast roaming protocol based on Prüfer sequence [5]. Using Prüfer sequence, we can significantly reduce the multicast roaming protocol's complexities in terms of both communication overhead and operational overhead [33].

A high-level description of the multicast roaming protocol is as follows:

- i. A roaming node uses derived subgroup/group key to encrypt the message.
- ii. The roaming node encrypts the message again by using the group's roaming key. Together with the desired group member list, the messages are sent to the host group manager.
- iii. The host group manager decrypts the message by using the group's roaming key. Based on the desired group member list, the host group manager creates a multicast packet by using Prüfer encoding algorithm to generate a Prüfer sequence; the derived Prüfer sequence is set in the multicast packet header; and then the multicast packet is sent out.
- iv. Based on the received Prüfer sequence and Prüfer decoding algorithm, an MBN node makes decisions to drop or forward the multicasting message. If the MBN node is one of destination nodes, it forwards the packet to the corresponding receivers.
- v. The receiver uses the corresponding subgroup/group key to decrypt the encrypted message.

#### Prüfer encoding/decoding algorithms

The Prüfer encoding/decoding algorithms are as follows:

**Algorithm 2 (Prüfer encoding algorithm)** *Let  $(i, j)$  represents the edge between node  $i$  and  $j$ . The corresponding Prüfer sequence  $P$  of a tree  $T$  can be obtained by the following steps:*

- i. Let node  $i$  be the lowest labeled leaf node of  $T$  and  $j$  be the node incident to  $i$ ; append  $j$  to the end of  $P$  from left to right.*
- ii. Remove node  $i$  and edge  $(i, j)$  from  $T$ ;*
- iii. Go back to step 1 until one edge is left. □*

**Algorithm 3 (Prüfer decoding algorithm)** *For a Prüfer sequence  $P$  and a set  $P'$  of eligible node labels not included in  $P$ , a unique tree  $T$  can be obtained by the following steps:*

- i. Let  $i$  be the lowest integer of  $P'$  and  $j$  be the left-most integer of  $P$ . Add edge  $(i, j)$  into  $T$ . Remove  $i$  from  $P'$  and  $j$  from  $P$ . If  $j$  does not occur elsewhere in  $P$ , put it into  $P'$ .*
- ii. Repeat step 1 until no element is left in  $P$ .*

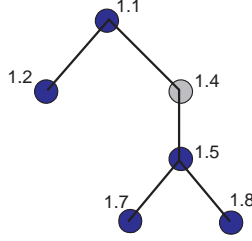


Fig. 7. A *Steiner tree* for multicast group, rooted at MBN node 1.1.

iii. Now that there are exactly two nodes  $r$  and  $s$  left in  $P'$ , add edge  $(r, s)$  into  $T$ .  $\square$

The Prüfer encoding/decoding schemes are used when a node wants to communicate to multiple receivers in different groups, which is equivalent to multicasting a message to multiple receivers over a multi-hop network. Since an HMANET has no fixed routers, all nodes are moveable and can be connected dynamically in an arbitrary manner. We assume that the cost of each link in the HMANET is assumed to be 1. We adopt Prim's algorithm to compute a minimum cost multicast tree, known commonly as the Steiner tree.

Earlier in the example shown in Fig. 3, we assumed that MBN node 1.1 receives a packet for multicasting securely to multicast group  $MG = \{1.1, 1.2, 1.5, 1.7, 1.8\}$ . Fig. 7 shows the network topology graph computed by node 1.1 when it receives the packet for multicasting. Source node 1.1 derives a *Steiner tree* with 6 nodes. Note that the node 1.4 is not a message receiver but it is in the *Steiner tree*. Thus, we use gray color to represent node 1.4.

To illustrate the Prüfer encoding algorithm, we note that node 1.2 is the lowest labeled leaf node and node 1.1 connects to node 1.2. Using the Prüfer encoding algorithm, node 1.1 becomes the first element of  $P$  and edge  $(1.2, 1.1)$  is removed from the graph. Next node 1.1 is the lowest leaf node with node 1.4 connecting to it. Append node 1.4 to  $P$  and remove node 1.1 and edge  $(1.1, 1.4)$ . Repeat this process until only edge  $(1.5, 1.8)$  is left and Prüfer sequence  $P = (1.1, 1.4, 1.5, 1.5)$  is obtained ( $P' = (1.2, 1.7, 1.8)$  and  $P'$  is the set of nodes in  $MG$  but not in  $P$ ). Thus, node 1.1 encodes the multicast tree with 6 nodes as the Prüfer sequence  $(1.1, 1.4, 1.5, 1.5)$  of length 4. Prüfer sequence  $P$  and its complement  $P'$  are put in the multicast packet header. When an MBN node receives the multicast packet, it first decrypts the Prüfer sequence by using Prüfer decoding algorithm. 1.2 is the smallest number in  $P'$  and 1.1 is the leftmost number in  $P$ . We do the following: adding edge  $(1.1, 1.2)$  to the tree, removing 1.2 from  $P'$ , and moving the leftmost integer 1.1 from  $P$  to  $P'$  since 1.1 is no longer exists in  $P$ . Thus, we can derive  $P = (1.4, 1.5, 1.5)$  and  $P' = (1.1, 1.7, 1.8)$ . Next, 1.1 becomes the smallest node in  $P'$  and 1.4 is the leftmost node in  $P$ . We do the following: adding edge  $(1.1, 1.4)$  in tree, removing node 1.1 from  $P'$ , and moving 1.4 from  $P$  to  $P'$ . In the third step,

edge (1.4,1.5) is added to the tree and node 1.4 is removed from  $P'$ . Finally, edge (1.5,1.7) is added to the tree and only 1.5 and 1.8 are left in the  $P'$ . We then add the edge (1.5,1.8) in the tree. In this way, we can reconstruct the *Steiner tree* shown in Fig. 7.

After reconstructing the *Steiner tree*, the MBN does the following:

- Algorithm 4 (Prüfer sequence routing)**
- i. If it is not in the sequence and belongs to the multicast group and it is a leaf node of the Steiner tree, it receives the packet, but does not forward it any further (e.g., node 1.2, 1.7, and 1.8).*
  - ii. If it neither belongs to the multicast group nor is in the Prüfer sequence, it simply discards the packet (e.g., node 1.3).*
  - iii. If it is not in the multicast group but in the Prüfer sequence, it realizes it is an interior node of the multicast tree, and it just forwards the packet (e.g., node 1.4).*
  - iv. If it is in the multicast group and Prüfer sequence, it realizes it is an interior node of the multicast tree, and it receives the packet and forwards the packet (e.g., node 1.5).  $\square$*

Using Algorithm 4, only the MBN nodes in the *Steiner tree* will forward the received packets to the next hop. Thus, by using Prüfer sequence, the communication overhead for group communications is reduced.

## 4 Performance Assessments

We present performance assessments from two perspectives: group key management (i.e., storage requirements, the communication overhead of group key management, and the complexity of group and subgroup key derivation) and the roaming protocol.

### 4.1 Performance Assessments of Group Key Management

We assume that the height of a hierarchical group structure is  $h$  and each group has the same size  $n$ . In an HMANET network structure, there are two extreme cases (see Fig. 8):

- i. each group only has one immediate lower-level group; and
- ii. each group member has one immediate lower-level group.

In general, these two cases address the lower bound and upper bound of the corresponding performance metrics. Our following discussion is based on these

two cases.

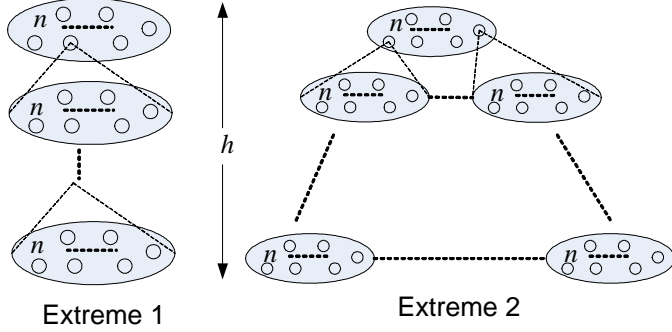


Fig. 8. Group hierarchical structures: two extreme cases.

#### 4.1.1 Storage Requirements

The biggest gain of our hierarchical group keying scheme is the reduction of the storage overhead since the storage requirements of our hierarchical group keying structure is independent of the size of the overall HMANET. For example, if the number of keys possessed by each group member is determined by the size of its home group, i.e., a flat group, the storage requirements of the home group with  $n$  members is  $n(n - 1)/2$  keys for both group manager and group members, see [23]. Fig. 9 shows the comparison of the group key size using 56 bits, 128 bits, and 160 bits. It provides a guideline to determine the group size under certain memory and key size restrictions. For example, if the storage is limited to 600 KB, then a group size of  $n = 419$  can be easily fulfilled with a 56 bits key size, but can only support a group size of  $n = 248$  if the key size is increased to 160 bits.

If we increase the number of levels of a hierarchical group structure, the total number of supported group members increases exponentially, while the number of predistributed keys for each group member remains the same. Using two cases stated earlier, we derive the following Lemma:

**Lemma 1** *When each group has only one immediate lower-level group, the number of supported group members is  $N = n \cdot h$ ; when each group member has one immediate lower-level group, the number of supported group members is  $N = \frac{n(n^h - 1)}{n - 1}$ .*

**Proof 1** In the first case, the total number of supported group members in each group is  $n$ . If the height of an HMANET is  $h$ , the total number of supported group members is  $N = n \cdot h$ . In the second case, at the level  $h = 1$ , we have  $n^1 = n$  group members; at level  $h$ , we have  $n^h$  group members. Accumulating the number of supported group members in all levels, the number of members in an HMANET framework is  $N = \sum_{i=1}^h n^i = \frac{n(n^h - 1)}{n - 1}$ . Thus, the

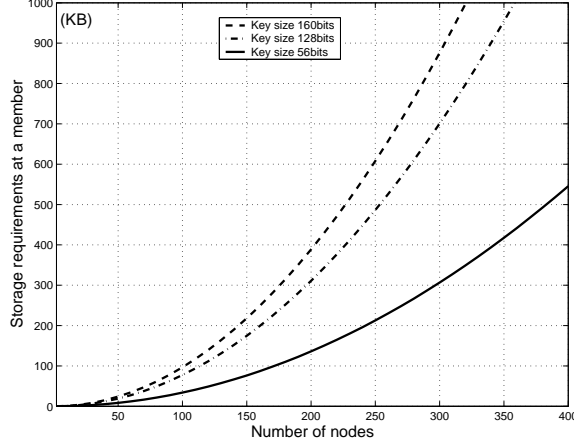


Fig. 9. Storage requirements for group members.

overall group size is between  $n \cdot h$  and  $\frac{n(n^h-1)}{n-1}$ , where  $n(n-1)/2$  number of keys are preinstalled for each member.  $\square$

#### 4.1.2 Communication Overhead

To assess communication overhead, we assume that each group member has the same probability to be selected as a receiver in the HMANET. For a subgroup size  $L$ , we use  $T(L)$  to represent the average number of key messages involved in each group and  $T_{tot}$  to represent the average number of total key messages. In [23], we showed that  $T(L)$  can be modeled by the function  $a + bL + cL \ln L$  for a flat group, where  $a, b, c$  are constants.

**Lemma 2** Assume that a subgroup member is equally likely with probability  $p$  to be a member of any of the groups and the subgroup size is  $k$ . Then,

(1) For the first case: the average communication overhead  $T_{tot}(k)$  is given by:

$$T_{tot}(k) = h \cdot \sum_{i=0}^k \binom{k}{i} p^i (1-p)^{k-i} T(i), \quad \text{where } p = 1/h. \quad (3)$$

(2) For the second case: on the average, the total number of key messages is

$$T_{tot}(k) = \sum_{i=1}^h n^{i-1} \cdot \sum_{i=0}^k \binom{k}{i} p^i (1-p)^{k-i} T(i), \quad \text{where } p = 1/\sum_{i=1}^h n^{i-1}. \quad (4)$$

**Proof 2** We assume that each user has the same probability to be located in each group. The subgroup size is denoted as  $k$ . The probability that a user is located in a group is denoted by  $p$ . Then, the probability that there are  $i$  members in a group is  $Pr(x=i) = \binom{k}{i} p^i (1-p)^{k-i}$ . Thus, the average number of messages is  $Pr(i)T(i)$  when there are  $i$  subgroup members. The expected number of messages sent to a group is  $\sum_{i=0}^k Pr(i)T(i)$ . If the communication system contains  $m$  groups, the overall average communication overhead is

$T_{tot}(k) = m \sum_{i=0}^k Pr(i)T(i)$  for a subgroup communication with size  $k$ . Thus, we can derive the formulas (3) where  $m = h$  and (4) where  $m = \sum_{i=1}^h n^{i-1}$  in Lemma 2. The worst scenario is  $2T(k)$  when  $k$  subgroup members are evenly distributed in the HMANET and each group manager duplicates and disseminates the messages to its group members. From [23], we know  $T(k)$  is a sub-linear function of  $k$ . Thus, in the worst case, the communication overhead is doubled in the HMANET compared to the flat group structure. However, the number of supported group members increases as  $O(2^{h-1}n)$  of a full key-tree HMANET where  $h$  is the height of the key tree and  $n$  is the size of a flat group. Thus the gain is substantial.  $\square$

#### 4.1.3 Complexity analysis of the group and subgroup key derivation

Recall that group and subgroup keys are derived from group member's key sets. Before the communication begins, a group member needs to know every other group member. When a group member sends an encrypted message, it needs to attach the key *ids* along with the encrypted message. Then a receiver can derive the proper subgroup key to decrypt the message.

Note that the longest key chain is equal to the size of the group. If function  $g$  is a one-way function and the subgroup size is  $k$ , the maximum required hash operations for a subgroup member is  $k - 1$  and the derivation complexity is  $O(k)$ . It has been shown that the generation of a hash chain has the complexity of  $O(\log_2 n)$  (see [34,35]). We assume that every subgroup is formed with an equal probability  $\frac{1}{2^n - 1}$ ; this implies that the probability of forming a subgroup with size  $k$  is  $\frac{\binom{n}{k}}{2^n - 1}$ . The average subgroup size is given by  $\sum_{k=1}^n k \frac{\binom{n}{k}}{2^n - 1}$ . It is easy to show that the average number of key-derivation operations is  $n/2$ . By using the scheme presented in [34,35], it can be shown that the average number of hash operations is  $\lceil \log_2 n - 1 \rceil$ .

We have conducted benchmark studies of various one-way function algorithms. The benchmark is computed on an HP iPAQ 4700 pocket PC with 624MHz Intel processor, and 64MB memory. Fig. 10 represents the plots for the CPU processing time utilized by various hashing algorithms along with the increased size of a subgroup (bottom-up: MD5-512, SHA-160,256,384,512; the numbers after MD5 and SHA are the input size – bits). Note that the benchmark is computed based on the average number of hash operations  $\lceil \log_2 n - 1 \rceil$  (a step function). In an HMANET, deriving the corresponding keys also depends on its height  $h$ . We can derive the following Lemma.

**Lemma 3** *When each group has one immediate lower-level group, the average number of key-derivation operations is linearly proportional to the height of the HMANET, i.e.,  $h = N/n$ , where  $h$  is the height,  $N$  is the total number of mobile users, and  $n$  is the size of a flat group. When each group member has an*

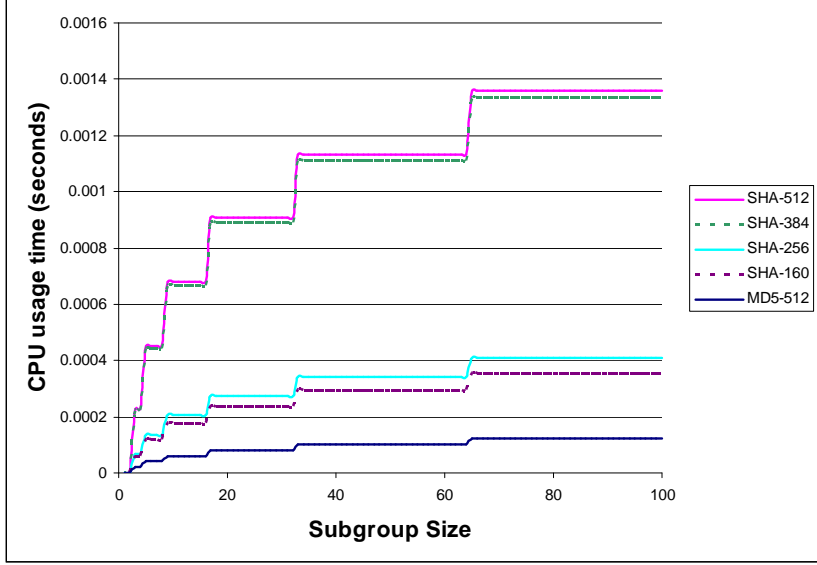


Fig. 10. CPU usage by hashing algorithms for Generating the encryption/decryption keys.

*immediate lower-level group, the average number of key derivative operations is proportional to the height of the HMANET, i.e.,  $h = \log_n N$ .*

From this lemma, we can determine that the number of derivative operations is between  $(\log_n N) \cdot D$  and  $(N \cdot D)/n$ , where  $D$  is the average number of derivative operations within each group.

#### 4.1.4 Remark

The HMANET structure is targeted to reduce the storage overhead for each group member. However, with the increases in the number of groups and the height of the hierarchical structure, the communication overhead and the key derivative complexity do increase, i.e., a sub-linear function of a subgroup size. Compared to the storage and communication overhead, the key derivative complexity is not a major issue, since the height of an HMANET increases slowly (in the order of  $\log_n N$ ). Thus, there is a tradeoff in balancing the storage overhead and the communication overhead.

## 4.2 Performance Analysis of Roaming Protocols

Roaming requires MBN nodes to maintain the network topology in realtime. Topology-aware ad-hoc routing protocols, such as TBRPF [32] or protocols based on positioning system [33] (i.e., using GPS devices) can build network topology. Each MBN node will compute a multicast tree (i.e., a minimum cost

multicast tree – a *Steiner tree*) by using heuristic algorithms such as [36,37].

Once the multicast tree is determined, an MBN node encodes the tree in a Prüfer sequence. The derived Prüfer sequence is put in the header of the multicast packet. Using Prüfer sequence is communication efficient since mobile nodes do not need to send the whole description of the multicast tree (e.g., node adjacency matrix). The size of Prüfer sequence is  $O(n)$ , where  $n$  is the size of multicast group. It can be proved that the complexities of Prüfer encoding and decoding algorithms are both in the order of  $O(n^2)$ . Based on the Prüfer sequence multicast routing algorithm, the receiver makes a decision to forward or drop the packets. Since an MBN node knows all its neighbors, it can select the next-hop forwarding nodes.

## 5 Summary

In this paper, we present a secure group key management framework for hierarchical mobile ad-hoc networks. Our approach considers 1) a security model, 2) a hierarchical group keying scheme using a key-chain approach, and 3) a roaming protocol between host groups and home groups. Our framework is suitable for ad-hoc network applications where the overall group population is stable, the subgroup communication is frequent and highly dynamic, and the mobility of a mobile node is not restricted within its communication range. We propose the first solution for hierarchical mobile ad-hoc networks using a modified Bell-La Padula model.

## References

- [1] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, T.-W. Chen, Scalable Routing Strategies for Ad Hoc Wireless Networks, *IEEE Journal on Selected Areas in Communications* 17 (8) (1999) 1369–1379.
- [2] G. Pei, M. Gerla, X. Hong, C.-C. Chi, A Wireless Hierarchical Routing Protocol with Group Mobility, in: *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 1999, pp. 1538–1542.
- [3] R. Ramanathan, M. Steenstrup, Hierarchically-Organized, Multihop Mobile Wireless Networks for Quality-of-Service Support, *Mobile Networks and Applications* 3 (1) (1998) 101–119.
- [4] C. E. Jones, K. M. Sivalingam, P. Agrawal, J. C. Chen, A Survey of Energy Efficient Network Protocols for Wireless Networks, *Wirel. Netw.* 7 (4) (2001) 343–358.

- [5] H. Prüfer, Neuer Beweis eines Satzes ueber Permutationen, *Archiv für Mathematik und Physik*, 27 (1918) 742–744.
- [6] M. Bishop, *Computer Security: Art and Science*, Addison-Wesley Professional, 2002.
- [7] S. Rafaei, D. Hutchison, A Survey of Key Management for Secure Group Communication, *ACM Computing Surveys* 35 (3) (2003) 309–329.
- [8] Y. Kim, A. Perrig, G. Tsudik, Simple and fault-tolerant key agreement for dynamic collaborative groups, in: *Proceedings of ACM Conference on Computer and Communications Security*, 2000, pp. 235–244.
- [9] G. Ateniese, M. Steiner, G. Tsudik, Authenticated group key agreement and friends, in: *Proceedings of the 5th ACM conference on Computer and communications security*, ACM Press New York, NY, USA, San Francisco, California, United States, 1998, pp. 17 – 26.
- [10] M. Burmester, Y. Desmedt, Efficient and Secure Conference-key Distribution, in: *Proceedings of Security Protocols Workshop*, Springer-Verlag, Cambridge, UK, 1996, pp. 119 – 129.
- [11] J. Alves-Foss, An Efficient Secure Authenticated Group Key Exchange Algorithm for Large and Dynamic Groups, in: *Proceedings of 23rd National Information Systems Security Conference (NISSC)*, National Institute of Standards and Technology, National Computer Security Center, Baltimore, MD, USA, 2000, pp. 254 – 266.
- [12] R. Blom, An Optimal Class of Symmetric Key Generation Systems, in: *EUROCRYPT’84*, Vol. 209 of *Lecture Notes in Computer Science*, Springer-Verlag, Paris, France, 1985, pp. 335–338.
- [13] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, Perfectly-Secure Key Distribution for Dynamic Conferences, *Information and Computation* 146 (1) (1998) 1–23.
- [14] A. Fiat, M. Naor, Broadcast Encryption, in: *CRYPTO’93*, Vol. 773 of *Lecture Notes in Computer Science*, Springer-Verlag New York, Inc., Santa Barbara, California, United States, 1994, pp. 480–491.
- [15] T. Ballardie, Scalable Multicast Key Distribution, RFC 1949 (1996) <http://www.ietf.org/rfc/rfc1949.txt>.
- [16] L. Gong, N. Shacham, Elements of Trusted Multicasting, in: *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, Fairfax, Virginia, 1994, pp. 176–183.
- [17] H. Harney, C. Muckenhirn, Group Key Management Protocol (GKMP) Architecture, RFC 2094.
- [18] S. Mitra, A Framework for Scalable Secure Multicasting, in: *Proceedings of ACM SIGCOMM*, 1997, pp. 277–288.

- [19] C. K. Wong, M. Gouda, S. S. Lam, Secure group communications using key graphs, *IEEE/ACM Transactions on Networking* 8 (1) (2000) 16–30.
- [20] A. T. Sherman, D. A. McGrew, Key Establishment in Large Dynamic Groups Using One-Way Function Trees, *IEEE Transactions on Software Engineering* 29 (5) (2003) 444–458.
- [21] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, B. Plattner, The VersaKey Framework: Versatile Group Key Management, *IEEE Journal on Selected Areas in Communications* 17 (9) (1999) 1614 – 1631.
- [22] D. M. Wallner, E. J. Harder, R. C. Agee, Key Management for Multicast: Issues and Architectures, RFC 2627.
- [23] D. Huang, D. Medhi, A Key-chain Based Keying Scheme For Many-to-Many Secure Group Communication, *ACM Transactions on Information and System Security* 7 (4) (2004) 523 – 552.
- [24] L. Eschenauer, V. D. Gligor, A Key-management Scheme for Distributed Sensor Networks, in: *Proceedings of 9th ACM Conference on Computer and Communication Security (CCS-02)*, 2002, pp. 41–47.
- [25] H. Chan, A. Perrig, D. Song, Random Key Predistribution Schemes for Sensor Networks, in: *Proceedings of 2003 Symposium on Security and Privacy*, IEEE Computer Society, Los Alamitos, CA, 2003, pp. 197–215.
- [26] S. Basagni, K. Herrin, D. Bruschi, E. Rosti, Secure Pebblenets, in: *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, ACM Press New York, NY, USA, 2001, pp. 156–163.
- [27] K. Rhee, Y. Park, G. Tsudik, An Architecture for Key Management in Hierarchical Mobile Ad-Hoc Networks, *Journal of Communications and Networks* 6 (2) (2004) 156–162.
- [28] D. E. Bell, Looking Back at the Bell-La Padula Model, in: *ACSAC '05: Proceedings of the 21st Annual Computer Security Applications Conference*, IEEE Computer Society, Washington, DC, USA, 2005, pp. 337–351.
- [29] D. E. Bell, L. J. LaPadula, Secure Computer Systems: Unified Exposition and Multics Interpretation, Tech. rep., MITRE Corporation (1976).
- [30] C. Landwehr, C. Heitmeyer, J. McLean, A Security Model for Military Message Systems: Retrospective, *ACM Transaction on Computer System* 2 (3) (1984) 198–222.
- [31] C. Karlof, D. Wagner, Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures, *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols* 1 (2–3) (2003) 293–315.
- [32] R. G. Ogier, F. L. Templin, M. G. Lewis, Topology Dissemination Based on Reverse-Path Forwarding (TBRPF), IETF Internet RFC 3684.

- [33] S. Basagni, I. Chlamtac, V. R. Syrotiuk, Location Aware, Dependable Multicast for Mobile Ad-Hoc Networks, *Computer Networks* 36 (2001) 659–670.
- [34] D. Coppersmith, M. Jakobsson, Almost Optimal Hash Sequence Traversal, in: *Proceedings of Financial Cryptography*, InterVarsity Press, Southampton, Bermuda, 2002, pp. 102–119.
- [35] M. Jakobsson, Fractal Hash Sequence Representation and Traversal, in: *Proceedings of the IEEE International Symposium on Information Theory (ISIT02)*, 2002, pp. 437–444.
- [36] F. K. Hwang, D. S. Richards, Steiner Tree Problems, *Networks* 22 (1992) 55–89.
- [37] K. Makki, N. Pissinou, O. Frieder, Efficient Solutions to Multicast Routing in Communication Networks, *Mobile Networks and Applications* 1 (2) (1996) 221 – 232.