Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 48 (2005) 657–682
Computer Networks
www.elsevier.com/locate/comnet

Efficient and robust key management for large mobile ad hoc networks

Bo Zhu [a,b,*], Feng Bao [b], Robert H. Deng [c], Mohan S. Kankanhalli [a], Guilin Wang [b]

[a] School of Computing, National University of Singapore, Singapore 117543, Singapore
[b] Department of InfoComm Security, Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613, Singapore
[c] School of Information Systems, Singapore Management University, Singapore 259756, Singapore

Received 28 January 2004; received in revised form 5 October 2004; accepted 1 November 2004
Available online 30 December 2004

Responsible Editor: G. Schaefer

Abstract

Existing research efforts in key management can only handle very limited number of nodes and are vulnerable to active attacks. In addition, the flexibility and adaptivity of handling dynamic risks in different parts of networks, although critical in the practical usages of ad hoc networks, have been largely ignored. In this paper, we propose a novel hierarchical scheme based on threshold cryptography to address both security and efficiency issues of key management and certification service in *Mobile Ad hoc Network* (MANET). The main contributions of our key management scheme include: 1. providing various parts of MANET the flexibility of selecting appropriate security configurations, according to the risks faced; 2. providing the adaptivity to cope with rapidly-changing environments; 3. handling of MANETs with a large number of nodes; 4. issuing certificates with different levels of assurance. We also propose two algorithms, which can be used independently from the hierarchical structure, to protect certification services in ad hoc networks from active attacks. Our simulation results show that, compared to the previous work [16,18,19], our second algorithm is much faster in a friendly environment. When the key length is 1024 bits, the process of generating or renewing a certificate in our second algorithm is around six to eight times faster, and the process of generating partial certificates in our second algorithm is around 20–80 times faster. The latter advantage is critical in MANET where intrinsically the less help a node requests from its neighbors, the higher is the chance of obtaining the help. Furthermore,

* Corresponding author. Address: Department of InfoComm Security, Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613, Singapore. Tel.: +65 68748272.
E-mail addresses: zhubo@i2r.a-star.edu.sg, zhubo@comp.nus.edu.sg (B. Zhu), baofeng@i2r.a-star.edu.sg (F. Bao), robertdeng@smu.edu.sg (R.H. Deng), mohan@comp.nus.edu.sg (M.S. Kankanhalli), glwang@i2r.a-star.edu.sg (G. Wang).

1389-1286/$ - see front matter
doi:10.1016/j.comnet.2004.11.023

simulation results also show that our two algorithms work well in a hostile environment in which existing schemes work poorly.

Keywords: Ad hoc; Key management; Threshold signature; Verifiable secret sharing; Active attacks

1. Introduction

Among all the security issues in Mobile Ad hoc Network (MANET), key management is the most crucial one, because it is the essential assumption of many other security services. For instance, many secure routing protocols, such as ARAN [26] and SRP [22], assume that a pair of private and public keys and a certificate signed by a Trusted Third Party (TTP) have been assigned to nodes. Current research work in key management [33,16,18,19,17,14,30] can only handle limited number of nodes. When the number of nodes increases, most of them become either inefficient or insecure. In addition, since there is no clear line of defense in MANET, we cannot classify nodes in advance according to risks that they face. Even worse, in some cases, e.g. soldiers in the battle field, not only different parts of networks face different degrees of risks, but also such risks may change rapidly due to the dynamic property of MANET. Therefore, flexibility and adaptivity are two crucial properties that should be considered when we design a key management scheme for MANET. Besides that, a major difference between MANET and wired network is that, in the former, nodes normally have very limited power supplies. Therefore, any protocol that requires high computation cannot be of practical use.

There are two types of possible attacks on certification services in MANET: passive attacks and active attacks. In passive attacks, adversaries simply drop and refuse to forward other nodes' requests of assigning or renewing certificates. In active attacks, in contrast, adversaries may return a fake reply (e.g. an invalid partial certificate) to the node requesting certification service. Most of current approaches for certification services [16,18,19,17,14] mainly focused on passive attacks, and are inefficient in cases that malicious nodes launch active attacks. Unfortunately, active attacks may result in great risks, especially in the military field. Due to the poor physical security in MANET, adversaries may take over some nodes in the network. By launching active attacks, adversaries can disable the certification services of the whole network without being caught, with only a very small portion of nodes.

In this paper, we propose a novel hierarchical security scheme, called *Autonomous Key Management* (AKM), which can achieve flexibility and adaptivity, and handles MANET with a large number of nodes. AKM also enables the ability to issue certificates with different levels of assurance with help from a relatively small number of nodes. In addition, we propose two algorithms, which are based on threshold cryptography and Verifiable Secret Sharing (VSS) and are independent from AKM, to resist active attacks towards certification services. Simulation results show that our second algorithm is much more efficient than previous works [14,16–19].

The rest of the paper is organized as follows. Section 2 studies the related work in the literature. In Section 3, we give an overview of our approaches for securing key management and certification services in MANET. After introducing some cryptographical primitives in Section 4, in Section 5 and Section 6, we present the general design of AKM and show how scalable secret share updates are achieved in AKM to satisfy the dynamic environment like MANET. Following that, in Section 7, we propose two algorithms that can protect certification services from active attacks, and show the method of issuing certificates with different levels of assurance. Simulation results on the security and efficiency of our scheme and algorithms are given in Section 8. In Section 9, we draw the conclusions.

2. Related work

In [33], Zhou and Haas focused on how to establish a secure key management service in an ad hoc networking environment. They proposed to use threshold cryptography [3,2] to distribute trust among a set of servers. The focus of their work is to maximize the security of the shared secret in the presence of possible compromises of the secret share holders. It assumes a small group of servers with rich connectivity. Therefore, it is not suitable for purely ad hoc environments. Furthermore, the authors proposed to employ proactive schemes [13,12,11,9,8] to achieve share refreshing and to adapt to changes in the network in a scalable way. However, their solution is only suitable for a small group of servers and is inefficient for large ad hoc networks.

In [16,18,19], Kong et al. also took advantage of threshold secret sharing to distribute the functions of the Certificate Authority (CA) to normal nodes. In other words, each node holds a secret share, and multiple nodes in a local neighborhood jointly provide complete services. It minimizes the effort and complexity for mobile clients to locate and contact the service providers. One of the two major weaknesses of this scheme is that it is difficult to set an appropriate threshold k, which is a globally fixed parameter that is honored by each entity in the system. This scheme assumes that each node has *at least* k one-hop legitimate neighboring nodes. If k is set to a large number, many nodes may have problems in localizing certification service. On the other hand, if k is too small, the probability of global secret key being compromised is quite high. Another serious problem is that, due to the inability of distinguishing adversaries who provide invalid partial certificates from honest nodes, both of their algorithms for certification renewals are vulnerable to active attacks, such as sending invalid partial certificates which result in the failure of renewing or assigning a certificate. In [17], Lehane et al. presented a similar scheme based on shared RSA key generation, and thus the scheme shares the pros and cons of Kong's scheme. In addition, according to their empirical results, the efficiency of their protocol is not good.

One recent work by Narasimha et al. [21] pointed out that Kong's algorithm [16,18,19] is vulnerable to active attacks, and proposed a Threshold DSA signature scheme. However, their scheme is based on an early protocol proposed by Feldman [5], which has been shown to have a security flaw. A secure protocol has been proposed in [10], which we will use for our algorithms. In addition, their scheme is in fact similar to our first algorithm, in the sense that both of them are standard VSS processes based on *Discrete Logarithm Problem* (DLP), and thus are more costly, compared to our second algorithm.

In [14], the authors tried to combine the ideas of ID-based and threshold cryptography. Their scheme avoids the need for users to generate their own public keys and distribute these keys throughout the network, since the user's identity acts as her public key. Besides that, users only need to propagate their identities instead of the certificates. This can lead to huge savings in bandwidth. However, the usage of ID-based cryptography instead of certificates also results in a few weaknesses. One major weakness is that the same ID, i.e. the public key in their scheme, can never be reused by a different user at a later time, because ID itself cannot present the variance of time. As a result, to avoid two nodes holding the same ID, users have to remember all the IDs that have ever appeared in the system. It is infeasible as the increase of the storage for IDs is proportional to the time span since the network is setup, given that users join and leave the network randomly. In addition, even if collisions can be avoided by letting users existing in the network select and assign an ID to the newly joined user, this ID is meaningless, since it does not provide useful information about the identity of the user. Finally, the secret and public key pair of a user in this scheme, in fact, is not a real Public Key Infrastructure (PKI) key pair. Consequently, users can only ensure the authentication by signing messages using their secret keys but cannot ensure the confidentiality by encrypting messages using their public keys, namely their identities.

In [31], Yi and Kravets proposed a scheme named Mobile Certificate Authority (MOCA). Compared to [16,18,19], this scheme limits the candidates who hold a share of the secret key of the network to a subset of users instead of all the users. However, it leads to problems, such as

who and how to judge the level of security and choose MOCAs, how to ensure that MOCAs are distributed uniformly, etc. Besides that, they assumed that there exist some nodes which are more trustworthy, computationally more powerful, and physically more secure. Therefore, the scheme is inappropriate for purely ad hoc networks. They also proposed a new pattern of communication, termed as "Manycast", between a client and MOCAs. The pattern is based on a strong assumption that the client knows which nodes are MOCAs and their positions. Furthermore, the authors ignored potential passive and active attacks from malicious users, which may weaken the benefit of employing MOCAs to save the packet overhead.

Both [33] and [16,18,19] require a Trusted Authority (TA), though the latter needs the authority only at the start-up phase. In [30], a more extreme case, where there is no central authority at all, was considered. It is a fully self-organizing public-key management system, in which each user is her own authority domain and issues public-key certificates to other users. When user u wants to verify user v, they merge their local certificate repositories and find an appropriate certificate chain from u to v in the merged repository. This method has a few weaknesses. First, the initialization phase (i.e. bootstrapping the local certificate repository) is relatively expensive. In addition, to achieve better assurance about the user-key binding, authentication metrics are used. However, how to find the most appropriate metric for MANET remains an open issue. Even if such a metric exists, the dynamic property [1] of MANET results in very high computation costs in reconstructing local certificate repositories of all the nodes. Furthermore, this scheme is also vulnerable to active attacks. The metric's confidence in someone's honesty can be easily cheated, as any user can create an arbitrary number of public keys and issue many false certificates.

[1] The dynamic property here does not mean the join/drop operations of nodes but revocations of certificates due to compromises or other reasons.

3. Overview of our approaches

To overcome the challenges described in Section 1, we provide approaches from both the architecture level and the algorithm level.

At the architecture level, we propose AKM which is based on the hierarchical structure and secret sharing to distribute cryptographic keys and provide certification services. In order to be employed in MANET, AKM is designed with several characteristics which are different from previous hierarchical key management schemes [20,15]. First, the hierarchical structure of AKM is a logical tree, in which all the leaf nodes represent real wireless devices, while all the branch nodes only exist logically. In other words, AKM does not require the existence of real branch nodes (i.e. trusted key servers in [20,15]), and thus is suitable for purely ad hoc environments. Second, flexibility and adaptivity can be obtained in AKM, since not only the structure of key management may change according to the increase/decrease of nodes, but also different parts of the structure have the freedom to set appropriate configurations to cope with various levels of risks. In addition, simulation results show that computation costs due to the variations are very small under common threshold and region size settings. Third, in AKM, secret keys of all branch nodes originate from one global secret key either directly or indirectly. The secret key of each branch node is shared by its sub-nodes (either branch nodes or leaf nodes) using the Shamir secret sharing scheme [28]. Such secret sharing process is performed recursively from top down to the lowest level. This characteristic allows us to issue certificates with different levels of assurance.

Once AKM is in operation, each real node holds a secret share which is used cooperatively with other nodes to maintain distributed key management services, such as assigning a secret share or a certificate to a newly-joined node.

At the algorithm level, we propose two algorithms, which are based on threshold cryptography and VSS and are independent from AKM. Both algorithms can resist active attacks. Given that there is no communication error, our first algorithm can assign a certificate within one round

with help from a group of $2k-1$ nodes, in spite of active attacks. In contrast, the second algorithm need help from only k nodes, although it may need more than one round to assign a certificate. Here, one round is defined to be the whole procedure that begins from a node requesting to be assigned a new certificate or renew its certificate to the combination and validation process of the certificate assigned or renewed. Simulation results show that, compared to the previous work [16,18,19,17,14], our second algorithm is not only much faster in a friendly environment, but it also works well in a hostile environment in which existing schemes work poorly. Furthermore, the process of generating partial certificates in our second algorithm is extremely fast. Such advantage is critical in MANET where intrinsically the less help a node requests from its neighbors, the higher is the chance of obtaining the help. Consequently, using our second algorithm, a node can easily find enough neighboring nodes that provide the certification service.

4. Cryptographic primitives

In this section, we briefly describe various cryptographic techniques underlying our approaches.

4.1. Secret sharing

The first secret sharing scheme was proposed by Shamir in 1979 [28]. This scheme is also called a (n,k)-*threshold scheme*, since it has the following properties: (1) any k or more users can reconstruct the secret from their shares; (2) for $k-1$ or fewer users, it is impossible to reconstruct the secret. The integer k is called the *threshold*. The details of the scheme are shown as follows.

Let $p > n$ be a large prime, let $c_1,\ldots,c_n \in Z_p$ be the user identifiers, let k be the threshold, and let $S \in Z_p$ be the secret, where Z_p is the set of residues modulo p (i.e. $Z_p = 0,1,\ldots,p-1$). A TA chooses random elements $a_1,\ldots,a_{k-1} \in Z_p$ and sets up the polynomial

$$f(x) = a_{k-1}x^{k-1} + \cdots + a_1 x + S \in Z_p[x]$$

of degree at most $k-1$. The shares are obtained by

$$S_i = f(c_i) \pmod{p} \quad \text{for } 1 \leqslant i \leqslant n$$

and then distributed to the users.

Using the Shamir secret sharing scheme, it is easy to add new users without changing the shares of the existing users. The TA just chooses a non-zero identifier $c_{n+1} \in Z_p$ that has not been used before and assigns the share $S_{n+1} = f(c_{n+1}) \pmod{p}$. This does not affect the existing shares. On the other hand, the Shamir secret sharing scheme can be used only once. As soon as the members of a privileged coalition have disclosed their shares to recover the secret, these shares are compromised.

4.2. Proactive security

Distribution of the key makes it harder for an adversary to expose the secret key, but does not remove this risk. Common mode failures, flaws that may be present in the implementation of the protocol or the operating system being run on all servers, imply that breaking into several machines may not be much harder than breaking into one. Thus, it is realistic to assume that even a distributed secret key can be exposed. Proactive schemes [5,7,12,9,32] address this to some extent, requiring all of the break-ins to occur within a limited time frame.

A proactive threshold cryptography scheme uses share refreshing, which enables users to compute new shares from old ones in collaboration without disclosing the service private key, i.e. the shared secret, to any user. The new shares constitute a new (n,k) sharing of the service private key. After refreshing, users remove the old shares and only keep the new ones. Because the new shares are independent of the old ones, the adversary cannot combine old shares with new shares to recover the private key of the service. Thus, the adversary is challenged to compromise k users between periodic refreshings. Share refreshing relies on the following homomorphic property. If $(s_1, s_2,\ldots,s_n)$ is a (n,k) sharing of S and $(s'_1,s'_2,\ldots,s'_n)$ is a (n,k) sharing of S', then $(s_1+s'_1, s_2+s'_2,\ldots, s_n+s'_n)$ is a (n,k) sharing of $S+S'$. If S' is 0, then we get a new (n,k) sharing of S.

4.3. Verifiable secret sharing schemes

There are two weaknesses in the Shamir secret sharing scheme.

Honesty of the dealer. If the dealer distributes erroneous subshares to some or all the users, how can users verify whether the subshares received are correct?
Honesty of users. During the share recovery process, if some compromised users provide false subshares, how can other users detect them?

Share refreshing must tolerate missing subshares and erroneous subshares from compromised users. A compromised user may not send any subshares. However, as long as correct users agree on the set of subshares to use, they can generate new shares using only subshares generated from k users in a (n,k) secret sharing scheme. To detect incorrect subshares, a few VSS schemes were proposed in [5,24]. A VSS scheme generates extra public information for each (sub)share using a one-way function. The public information can testify the correctness of the corresponding (sub)shares without disclosing them.

5. Autonomous key management for large Ad Hoc networks

To be more intuitive, in this section, we first give an example to show the method of constructing the hierarchical structure in AKM, instead of presenting the details of the scheme, such as notation and definitions, directly.

5.1. An example of constructing the hierarchical structure in AKM

Suppose that, at the beginning, there are two users that are physically nearby and can communicate directly with each other. However, they do not trust each other. Therefore, they decide to generate a $(2,2)$ secret sharing system cooperatively. [2] A distributed VSS scheme (e.g. the one proposed in [10]) is employed, since it is not suitable to assume the existence of a TTP in purely ad hoc networks. The structure of AKM at the initial stage is shown as Fig. 1, where the square and circle signs represent real nodes (i.e. users) and logical nodes, respectively. Each user holds a share B_i $(i = 1,2)$ of the secret A, where i is the identity of the user in the group, but none of them can deduce A by itself. The secret can only be recovered under the cooperation of at least two users.

Afterwards, when a new user wants to join this group, she requires help from at least two members currently in the group. Once the new user is accepted, she is assigned with a new share of A corresponding to her identity (e.g. 3 in the example) in the group. And the configuration of the secret sharing system is changed from $(2,2)$ to $(3,2)$, as shown in Fig. 1. This ''Join'' operation is to be explained in details in Section 6.2.1.

Although the shares holding by the current members are refreshed periodically to protect the shared secret, the system becomes more and more insecure, when more and more users join the group. Suppose that, in our example, members in the group think that the system is insecure when the ratio of the threshold of the secret sharing system to the number of users in the group is less than 0.4. Thus, when the number of users reach six (as shown in Fig. 1), members in the group have to divide themselves into a few groups, such as two groups in the example. More specifically, B_1, B_2 and B_3 form a new group, members of which share a newly-generated secret B_7. Similarly, B_4, B_5 and B_6 form another group and share the secret B_8. The resulting structure of AKM after the division is shown in Fig. 1. This operation increases the height of the tree structure, and is called ''Expansion'' operation, which is to be explained in details in Section 6.2.5. In AKM, B_7 and B_8 are generated in such a way that both of them are shares of the initial secret A. However, the secret sharing config-

[2] In some cases, if all the initial users think that the number of adversaries among them is less than a value k at that time, the threshold of the initial secret sharing system could be set to k, which is less than the number of initial users.

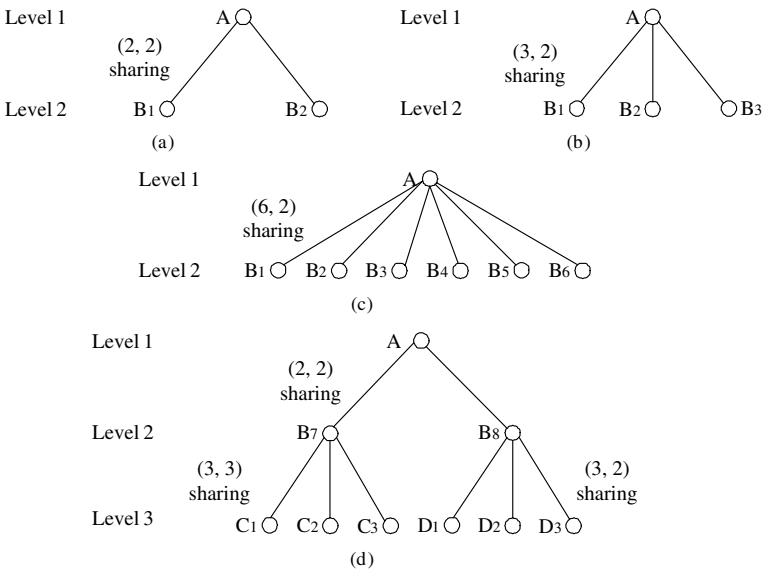

Fig. 1. An example of constructing the hierarchical structure in AKM.

urations within two newly-formed groups are independent from each other, and are determined by users that are initial members in the newly-created group. In this example, C_1, C_2 and C_3 (i.e. B_1, B_2 and B_3) totally distrust each other and employ a (3,3) secret sharing scheme within group C_i. In contrast, D_1, D_2 and D_3 (i.e. B_4, B_5 and B_6) think that there are less than two adversaries among them between consecutive secret share updates. Thus, they choose a (3,2) secret sharing scheme. Note that, as shown in Fig. 1, the branch nodes holding B_7 and B_8 are logical nodes, and thus do not exist in reality. And secrets B_7 and B_8 are neither stored on any real node nor recovered explicitly at a later time.

Similarly, when more and more users want to join group C_i ($i = 1,2,3$) or group D_i ($i = 1,2,3$), they need help from at least k users in that group, where k is the threshold of the secret sharing configuration of that group (i.e. $k = 3$ for group C_i, and $k = 2$ for group D_i).

5.2. Notation and definitions

Here we explain the definitions of some concepts used in this paper, using a four-level MANET system (shown in Fig. 2) as the example:

- *Real nodes*. The leaf nodes in the hierarchical structure of AKM. They have their own personal PKI key pairs, and they are corresponding to real devices in MANET. In Fig. 2, node A to R are real nodes.
- *Virtual nodes*. The branch nodes in the hierarchical structure of AKM. They are virtual and thus do not represent real devices in MANET. In Fig. 2, the root node and node S to Z are virtual nodes.
- *Master node*. The branch node that a node originates directly from. For example, node Z is the master node of node V to X.
- *Region*. Consists of all the real and virtual nodes originating directly from the same virtual node. Each virtual node corresponds to a region. For example, region T consists of real node D to F, and region Z consists of virtual node V to X.
- *Overall region size* (ORS). The number of nodes which possess secret shares originated from the same secret, i.e. the secret key of the same region, between two consecutive secret share updates of this region. For example, we assume that Fig. 2 shows the structure of the network just after a secret share update. At this time, the ORS of region S is 3. If node B leaves region S, the ORS of region S is still 3 until the next

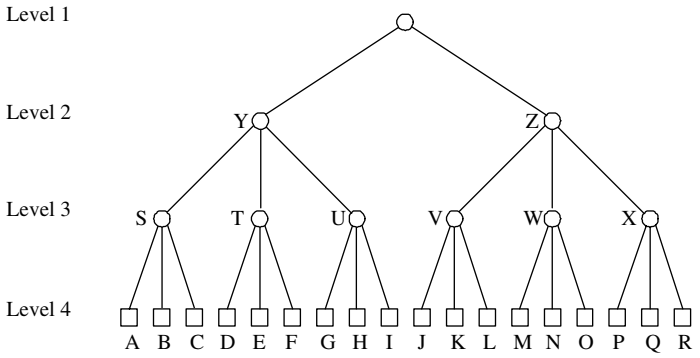

Fig. 2. A four-level MANET system.

secret share update, even though in fact the number of the nodes currently in region S is 2. We propose this new parameter, in view of the fact that the leaving node may still keep a secret share of the secret key of the region.

- *Regional trust coefficient* (RTC). The parameter to indicate how secure a region is. It is defined as the ratio of the threshold of a region to its ORS.
- *Global trust coefficient* (GTC). The global lower limit on RTC.
- *Identifier* (ID). The entity to represent a node.
- *Global secret key* (GSK). The secret key of the whole MANET system.
- *Global public key* (GPK). The public key of the whole MANET system.

We use the notations shown in Table 1.

5.3. Assumptions

We make the following assumptions in this paper:

Table 1
Notations in AKM

PK_i	The public key of a region "i"
SK_i	The secret share held by node "i". If node "i" is virtual, the secret share is also its secret key
ID_i	The identifier of a node "i"
pk_i	The public key of a real node "i"
sk_i	The secret key of a real node "i"
$SK_i(M)$	A message M is signed by SK_i

1. Each real node i has a personal PKI key pair $[pk_i, sk_i]$. The key pair can be generated by either a TTP or even the node itself.
2. Each real node joins and leaves the network randomly.
3. Given the public key of any virtual node, including the root node, it is computationally infeasible to obtain the corresponding secret key.
4. Between any two consecutive secret share updates, the number of adversaries that hold secret shares originated from the same secret key (i.e. the secret key of the same region) is less than k. The adversaries may still stay in the region, or have already left it.

5.4. Design of AKM

In AKM, we assume that each real node joins and leaves the network randomly. Therefore, due to the hierarchical nature of AKM, for a virtual node, the higher level it belongs to, the lower is the probability of occurrences of region-based operations (e.g. "Merge" and "Partition") involving this virtual node. In other words, the higher levels of the hierarchical structure remain relatively static, in spite of the fact that in MANET the lowest level of AKM (i.e. those real nodes) is highly dynamic. Together with this property, we can set appropriate RTCs so that the region-based operations are limited in lower levels.

AKM is self-organizing, since it does not rely on any TTP or TA at any stage. During the initial-

ization, as shown in the example in Section 5.1, in contrast to [16,18,19] in which a TA is required to bootstrap the initial nodes, in AKM the role of the TA has been replaced by those initial nodes themselves, which jointly generate GSK using a distributed VSS scheme [10]. In addition, when the environment changes, e.g. some nodes join or leave, those nodes currently in a group/region would cooperate and adjust accordingly by themselves. In other words, our solution is adaptive to dynamic environments without a TTP. Details about node-based and region-based operations are shown in Section 6.2.

In AKM, all the regions' secret keys originate from GSK either directly or indirectly. The secret sharing process based on [28] is performed recursively from top down to the lowest level. Each node in AKM except the root virtual node, either real or virtual, holds a share of the secret of its master node and a PKI key pair. For a virtual node, the secret share that it holds is also its secret key, and its public key is generated from the secret key. In this scheme, we employ an asymmetric cryptographic scheme based on DLP to generate the region's public key. For example, given that the secret key of region i is SK_i, its public key can be computed by $PK_i = g^{SK_i} (\mathrm{mod}\, p)$. Unlike virtual nodes, each real node generates its own PKI key pair through some other method, e.g. by a TTP or even itself. However, it also needs to store a share of the secret key of its region (i.e. the secret key of its master node). For the sake of clarity, we list how real nodes and virtual nodes generate their keys in Table 2. Note that, to ensure the security of AKM, the secret key of any virtual node should never be recovered explicitly, even when we need it to assign a certificate. Instead, partial certificates are generated individually with those shares of the secret key, and then they are combined to create the certificate. Details of assigning or renewing certificates are presented in Section 7 at a later time.

Besides the public key of itself, a real node needs to store public keys of other nodes in the hierarchical structure, since in purely ad hoc environments like MANET we cannot assume the existence of a trusted public server or directory storing public keys of other members. For a real node with sufficient storages, it will store two kinds of public keys: (1) public keys of itself and virtual nodes that are on its reverse path to the root. We denote the group of all these nodes as V; (2) public keys of nodes that are within the same region of any member in group V. For instance, in Fig, 2, node A needs to store (1) PK_A, PK_S, PK_Y and GPK; (2) PK_B and PK_C (B and C are in the same region of A), PK_T and PK_U (T and U are in the same region of S), PK_Z (Z is in the same region of Y). Public keys of those higher-level virtual nodes are useful, when we want to verify certificates with different levels of assurance. The generation of certificates with different levels of assurance is to be presented in Section 7.1.3. However, there is a trade-off between the storage and this new characteristic provided by AKM. In reality, if a real node has limited storages, it can store a subset of these public keys. More specifically, it can discard public keys of those nodes that are d levels higher than it. For instance, in the previous example, when $d = 2$, public keys stored on node A consist of (1) PK_A, PK_S; (2) PK_B and PK_C (B and C are in the same region of A), PK_T and PK_U (T and U are in the same region of S). The minimum set of public keys that must be stored on a real node consists of (1) public keys of all the real nodes in the same region as itself; (2) the public key of its master node. These public keys are required to assign or renew common certificates (in contrast to certificates with different levels of assurance), which are signed by the secret key of the real node's master node, as

Table 2
Key pairs and secret shares of nodes

	Real node	Virtual node
Secret share	Obtained during the secret sharing process	Obtained during the secret sharing process
Secret key	Obtained before the initialization of AKM	The secret share is used as its secret key
Public key	Obtained before the initialization of AKM	Generated from its secret key after receiving its secret share

presented in Section 7. Thus, A should at least store (1) PK_A, PK_B and PK_C (A, B, C are within the same region); (2) PK_S (S is the master node of A). For simplicity, we assume that in Fig. 2 the number of nodes in each region is n, and the secret sharing process within each region uses the same (n,k) configuration. Given that the height of the hierarchical structure is denoted as l, in such a system, there are n^{l-1} real nodes, and each of them needs to store $n+1$ (at minimum) to $(l-1)n+1$ (at maximum) public keys instead of storing public keys of all the n^{l-1} nodes.

It should be absolutely clear that the proposed AKM scheme does not requires a preset hierarchical structure for all the nodes. If it did, this requirement would be really inappropriate for ad hoc networks where nodes are highly dynamic and thus we cannot determine their positions in advance. Instead, AKM only requires a small number of nodes, which are physically nearby and can communicate directly with each other, during the initialization. More importantly, the structure of the whole network may vary when more nodes join or leave. Therefore, the hierarchical structure of the whole network is adaptive. In the above example, the number of initial nodes needed is only k, and the size of this network can be expanded to n^l at maximum at a later time. Actually, if there is no limit on the height of the hierarchical structure in AKM, AKM can contain nodes of arbitrary size.

Besides adaptivity, another main advantage of AKM is the flexibility of its structure. Each region [3] can determine its own size and the threshold of secret sharing, as long as it obeys the following rule: its RTC should be no less than GTC. The flexibility of the threshold parameter is very desirable, since in some cases different regions of the MANET may face risks of different intensities. Thus, the threshold should not be set to be globally uniform. Moreover, such property is very useful in balancing the security and efficiency requirements.

[3] In fact, as shown in the example in Section 5.1, it is not the virtual node representing the region but all the real nodes in this region that determine the secret sharing configuration during the initialization of the region, because the virtual node is logical and thus cannot take any operation by itself.

6. Scalable share updates

In AKM, in order to improve the robustness of share updates, we distribute the functionality of the dealer in the Shamir secret sharing scheme [28] to many real nodes. The share updating process occurs under two conditions. One is the regular periodic renewal of secure shares. The other condition happens during the node-based and region-based operations.

To ensure the security of share updates, we employ proactive secret sharing schemes [5,7,12,9] to adapt the configuration of secret sharing to variations in a highly dynamic environment, such as MANET. However, if a proactive scheme requires a large amount of computation, nodes in MANETs cannot handle such requirement because of their limited computation power. In order to reduce the cost to a tolerable level, we keep the threshold of a region unchanged in the lifetime of this region. It can greatly improve the efficiency of the proactive scheme, since the cost of changing the configuration of a secret sharing scheme from (n,k) to (n',k') is very high.

6.1. Regular periodic renewal of secret shares

In AKM, a proactive threshold cryptography scheme is used to enable nodes of a region to compute new shares from old ones in collaboration without disclosing the secret key of the region. It relies on the homomorphic property.

We notice that it is unnecessary to require all the nodes involved in the share refreshing process. Instead, the task can be done by only k nodes, since we assume that, between any consecutive secret share updates, the number of adversaries who hold secret shares originated from the same secret key is less than k. To detect those incorrect subshares, the VSS scheme [29] is employed.

Details are shown as follows. To renew the secret shares of all the n nodes in a region, first, k nodes are chosen from this region. Without lose of generality, we denote them as $(1,\ldots,k)$. Each of the k nodes, denoted as node i, randomly generates a (n,k) sharing of 0, denoted as $(SK_{i1}, SK_{i2},\ldots,SK_{in})$, and then distributes the corresponding subshare SK_{ij} to node $j \in \{1,\ldots,n\}$.

After receiving all the subshares generated by the k nodes, each node in the region, denoted as node j, can compute a new share from them and its old share ($SK'_j = SK_j + \sum_{i=1}^{k} SK_{ij}$). The new shares constitute a new (n,k) sharing of the service secret key. After refreshing, nodes remove the old shares and use the new ones to generate partial signatures. Because the new shares are independent of the old ones, the adversary cannot combine old shares with new shares to recover the secret key of the service. Thus, the adversary is challenged to compromise k nodes in the same region between periodic refreshing.

6.2. Common node-based and region-based operations

A comprehensive key management scheme must handle adjustments to secret shares subsequent to all membership-changing operations in the underlying communication system.

6.2.1. "Join" operation

"Join" operations happen when one real node wants to join a region. It is, in fact, the process of changing a region's configuration of the threshold scheme from (n,k) to $(n+1,k)$. For example, node i wants to join a region. First, it chooses a group of k nodes in this region denoted as group $G = \{1,\ldots,k\}$, and multi-casts its request to them. Once node j receives the request, it checks node i's certificate and its CRL. If node j decides to serve the request, it calculates a partial share for node i as

$$SK'_j = SK_j l_j(i) + \Delta_j \pmod q,$$

where Δ_j is the shuffling factor of node j, and $l_j(i) = \prod_{r=1,r\neq j}^{k} \frac{ID_i - ID_r}{ID_j - ID_r} \pmod q$. The shuffling factor is imported to prevent SK_j from being disclosed, because node i can easily recover SK_j from SK'_j if there is no shuffling factor. One method to generate the shuffling factor requires that each pair of nodes (j,r) in G exchanges a number S_{jr}, and then Δ_j is computed as

$$\Delta_j = \sum_{r=1,r\neq j}^{k} \sigma(j-r)\cdot S_{jr},$$

where $\sigma(x)$ is the sign function. Namely,

$$\sigma(x) = \begin{cases} 1, & x > 0, \\ -1, & x < 0, \\ 0, & \text{otherwise.} \end{cases}$$

Then node j returns the partial share to node i. After receiving k partial shares, node i can construct its secret share SK_i by adding them together

$$\sum_{j=1}^{k} SK'_j = \sum_{j=1}^{k} SK_j l_j(ID_i) + \sum_{j=1}^{k} \Delta_j = SK_i \pmod q$$

and verify its secret share by

$$g^{SK_i} = \prod_{j=1}^{k} (PK_j)^{l_j(ID_i)} \pmod p.$$

6.2.2. "Leave" operation

It happens when one node wants to quit from a region. Compared to "Join" operation, it is easier. When nodes receive a "Leave" request from a node in the same region or detect that a node leaves, they simply remove the certificate of that node from their key management records without recomputing secret shares. However, ORS does not decrease and thus RTC remains unchanged when a node "leaves" the region, because the node may still possess a secret share of the secret key of this region.

6.2.3. "Merge" operation

"Merge" operations happen when the number of nodes within a region drops under its threshold. As we know, the RTC of a region drops very quickly, if ORS increases a lot while the threshold remains unchanged. Therefore, we do not merge a region directly into a nearby region with the least size or the highest RTC. Instead, region i is divided into a few parts and each part is combined into one nearby region. Since the thresholds of the target regions are invariable, a "Merge" operation can be viewed as a series of "Join" operations (see Fig. 3).

6.2.4. "Partition" operation

"Partition" operations happen when the RTC of a region drops under GTC or is lower than

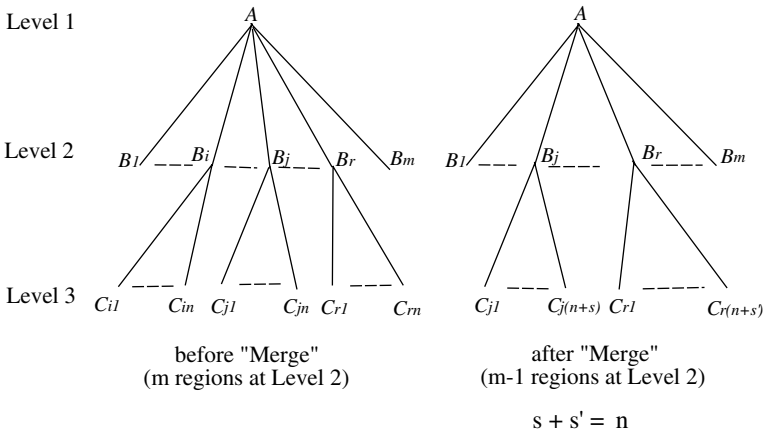

Fig. 3. "Merge" operation.

the security level expected. In AKM, the threshold of a region is fixed. Therefore, if the ORS of a region increases greatly, adversaries have a large chance to compromise the threshold system. A straight-forward approach is to partition the region into two regions with almost the same size. In the hierarchical key tree, new regions lie at the same level as the original one. For example, as shown in Fig. 4, region B_i with size $2n$ and the threshold k is partitioned into two regions B_i and B_{m+1}, each of which has n nodes and keeps its threshold as k. For the n nodes remaining in region B_i, they just need to renew their secret shares as discussed in Section 6.1.

In order to assign new secret shares to the n nodes in region B_{m+1}, first, region B_{m+1} chooses k regions at level 2, and then selects k nodes from each of the k regions. Without lost of generality, we denote the group of the k regions, the group of the k nodes from region B_j ($j = 1, \ldots, k$), and the group of all these k^2 nodes as $G_B = \{B_1, \ldots, B_k\}$, $G_j = \{C_{j1}, \ldots, C_{jk}\}$, and $G = \{C_{11}, \ldots, C_{1k}, \ldots, C_{k1}, \ldots, C_{kk}\}$, respectively. Then, a "Partition" request signed by the secret key of region B_i is generated and multicasted to all the nodes in group G. The IDs of region B_i, B_{m+1}, and the k regions are sent together with the request. By Lagrange interpolation, we know that

$$SK_{B_i} = \sum_{j=1}^{k} SK_{B_j} l_{B_j}(ID_{B_i}) \pmod{q}, \tag{1}$$

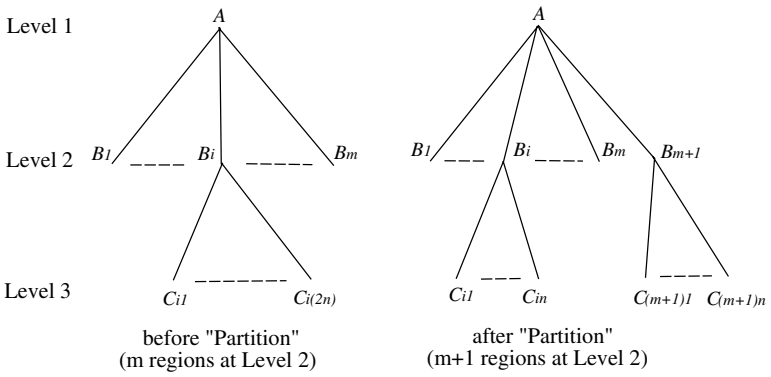

Fig. 4. "Partition" operation.

where $l_{B_j}(ID_{B_i}) = \prod_{r=1,r\neq j}^{k} \frac{ID_{B_i} - ID_{B_r}}{ID_{B_j} - ID_{B_r}} (\mathrm{mod}\, q)$. Since we know that

$$SK_{B_j} = \sum_{h=1}^{k} SK_{C_{jh}} l_{C_{jh}}(0) \quad (\mathrm{mod}\, q), \tag{2}$$

where $l_{C_{jh}}(0) = \prod_{r=1,r\neq j}^{k} \frac{ID_{C_{jr}}}{ID_{C_{jr}} - ID_{C_{jh}}} (\mathrm{mod}\, q)$. Combine Eqs. (1) and (2)

$$SK_{B_i} = \sum_{j=1}^{k} \sum_{h=1}^{k} SK_{C_{jh}} l_{C_{jh}}(0) l_{B_j}(ID_{B_i}) \quad (\mathrm{mod}\, q). \tag{3}$$

Similarly, we get

$$SK_{B_{m+1}} = \sum_{j=1}^{k} \sum_{h=1}^{k} SK_{C_{jh}} l_{C_{jh}}(0) l_{B_j}(ID_{B_{m+1}}) \quad (\mathrm{mod}\, q), \tag{4}$$

where $l_{B_j}(ID_{B_{m+1}}) = \prod_{r=1,r\neq j}^{k} \frac{ID_{B_{m+1}} - ID_{B_r}}{ID_{B_j} - ID_{B_r}} (\mathrm{mod}\, q)$.

From Eqs. (3) and (4), we get:

$$SK_{B_{m+1}} - SK_{B_i} = \sum_{j=1}^{k} \sum_{h=1}^{k} SK_{C_{jh}} l_{C_{jh}}(0) R_j \quad (\mathrm{mod}\, q), \tag{5}$$

where $R_j = l_{B_j}(ID_{B_{m+1}}) - l_{B_j}(ID_{B_i})$.

Consequently, to help nodes in region B_{m+1} generate the new shares of $SK_{B_{m+1}}$, each node C_{jh} in group G first computes $SK'_{C_{jh}} = SK_{C_{jh}} l_{C_{jh}}(0) R_j$. Then distributes the partial shares, i.e. the secret shares of $SK'_{C_{jh}}$, to nodes in region B_{m+1} using distributed VSS scheme proposed in [24]. According to Eq. (5), using the homomorphic property, each node in region B_{m+1} can compute its new share of $SK_{B_{m+1}}$ by adding k^2 partial shares from members in group G to its original share of SK_{B_i}.

6.2.5. "Expansion" operation

Similar to "Partition" operations, "Expansion" Operations happen when the RTC of a region drops under GTC or is lower than the security level expected. However, there is a special case in which we must take "Expansion" Operations. More specifically, when AKM has reached its capacity upper limit (namely, the RTCs of all the regions in AKM are equal to GTC), to ensure that the RTC of any region should not be less than GTC, we have to undertake a "Expansion" Operation, which increases the height of the hierarchical key tree.

For example, as shown in Fig. 5, before the "Expansion" operation, the height of AKM is L. Now there is a new node that wants to join region C_1, but the RTCs of all the regions in AKM including region C_1 are equal to GTC. Therefore, the "Expansion" operation is executed. We assume that, original secret sharing of region C_1 is executed by following:

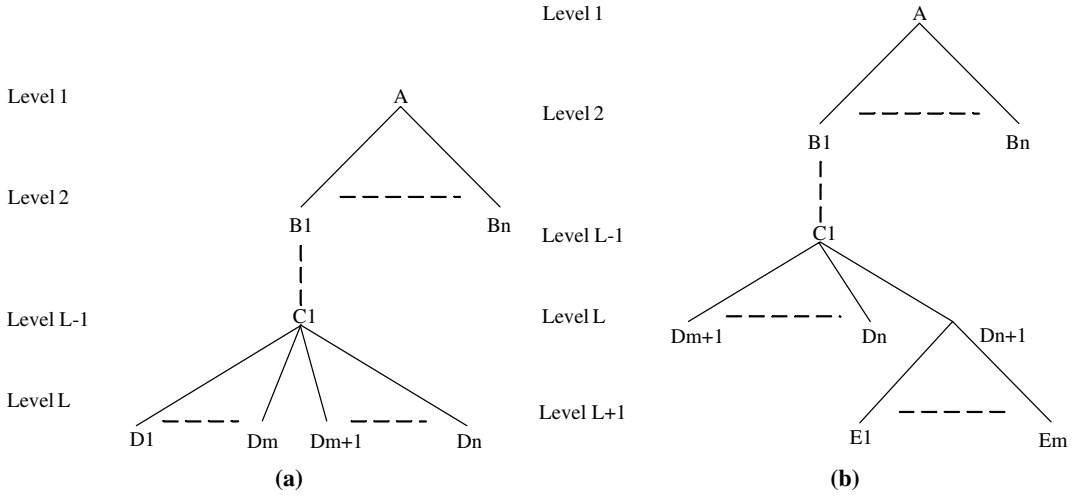

Fig. 5. "Expansion" operation: (a) before "Expansion" and (b) after "Expansion".

$$SK_{D_i} = a_0 + a_1 \cdot ID_{D_i} + \cdots + a_{k-1} \cdot ID_{D_i}^{k-1} \pmod{q}. \quad (6)$$

First, chooses a group of m nodes from region C_1 which will be degraded to Level $L+1$, where $k \leqslant m \leqslant n-k+1$. Without loss of generality, let the group be $R = \{D_1, \ldots, D_k\}$. Following that, chooses a new identity denoted as $ID_{D_{n+1}}$ for the master node of all the nodes degraded. According to the Shamir secret sharing scheme [28], the secret share for node D_{n+1} can be calculated by the k nodes in group R. However, during the "Expansion" operation, this secret share would be never calculated out or recovered explicitly, since we do not assume the existence of a TA at this stage.

For simplicity, the same (n,k) threshold scheme is employed in the newly created region D_{n+1}. Without loss of generality, we assume that a new identity ID_{E_i} is assigned to the node whose old identity is ID_{D_i}, or the identity is chosen by the node itself. Then each node in group R calculates the following partial secret share denoted as SK'_{E_i} and distributes it to the node with the new identity ID_{E_i}:

$$SK'_{E_i} = SK_{D_j} \cdot l_{D_{n+1}} + \sum_{r=1}^{k-1} b_{jr} \cdot ID_{E_i}^{r} \pmod{q}, \quad (7)$$

where $l_{D_{n+1}} = \prod_{h=1, h\neq j}^{k} \frac{ID_{D_{n+1}} - ID_{D_h}}{ID_{D_j} - ID_{D_h}}$. Finally, each node recovers its new secret share in the new region by combining all the secret shares as

$$SK_{E_i} = b_0 + b_1 \cdot ID_{E_i} + \cdots + b_{k-1} \cdot ID_{E_i}^{k-1} \pmod{q}, \quad (8)$$

where $b_0 = SK_{D_{n+1}}$, $b_r = \sum_{j=1}^{k} b_{jr}$ $(r = 1,2,\ldots,k-1)$. Therefore, all the coefficients of the secret sharing polynomial of the new region are cooperatively determined by the k nodes. At the end of the operation, the height of AKM increases to $L+1$.

In Fig. 5, there are n real nodes at level L before "Expansion". After performing the "Expansion" operation, m of them move to level $L+1$, while others still remain at level L. From the security aspect, it means that those real nodes at level L have higher privileges than those moving to level $L+1$, since each of them holds a share of SK_{C_1} directly. For nodes at level $L+1$, it requires k nodes of them to recover a share of SK_{C_1} cooperatively. It may not be desirable in some circumstances.

In Fig. 6, the "Expansion" operation groups previous n real nodes under C_1 into r regions, and the size of each region is denoted as S_i $(i = 1,2,\ldots,r)$. Therefore, we have $n = \sum_{i=1}^{i=r} S_i$. On one hand, the operation taken in Fig. 6 can be viewed as a series of operations (i.e. r operations) described in Fig. 5, which are undertaken separately for the r regions. On the other hand, the operation in Fig. 5 can also be viewed as a special case of the operation described in Fig. 6, in which $r-1$ of r regions has only one member, and this member remains its old share instead of computing a new one.

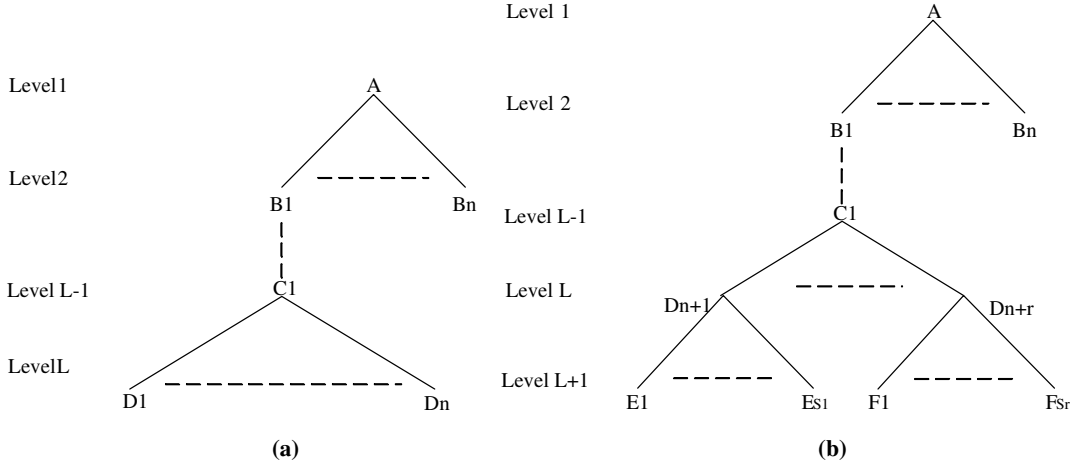

Fig. 6. Generic "Expansion" operation: (a) before "Expansion" and (b) after "Expansion".

For communication efficiency, when nodes cooperate to undertake region-based operations, they may take their (relative) physical locations into consideration as well. In other words, they may prefer neighbors nearby to those nodes that are far away, when region-based operations, e.g. "Partition" and "Expansion" operations, are executed.

6.2.6. "Contraction" operation

"Contraction" Operations happen when AKM reaches its capacity lower limit. To ensure that the number of nodes in any region in AKM is not less than the threshold set before, we have to decrease the level of the structure. Similar to a "Merge" operation, a "Contraction" Operation can be viewed as a series of "Join" operations as well.

7. Certification services against active attacks

7.1. Certificate initialization and renewal

There are two ways to issue a new certificate or renew a certificate. A node may be issued an initial certificate by an online or offline TA, after the authority verifies the authenticity through external means (e.g., in-person ID). However, it is both costly for the TA to maintain certificates of all the nodes and is inconvenient for new nodes to request their certificates from the root TA. An alternative approach is to use any coalition of k networking nodes to issue an initial certificate via collaborative admission control for this new node. In AKM, we use the second approach, and extend it to support certification services with different levels of assurance. A certificate with the lowest level of assurance is assigned with the cooperation of k real nodes within the same region, while a certificate with higher-level assurance can be achieved with the coalition of more nodes from different regions.

In AKM, the certificate of node i, denoted as $CERT_i$, is a statement $cert_i$ that is signed by the secret key of its master node. The statement $cert_i$ consists of the association between node i and its public key, the ORS and the threshold of the master node, and the expired time. The ORS and the threshold are included in $cert_i$, since receivers may be interested in and thus calculate the RTC of the region from which node i comes. For example, they are useful in performing region-based operations.

In [16,18,19], Kong et al. employed the RSA scheme to provide the certification services. It works well under the flat structure. However, we find that it is not suitable for the hierarchical structure like AKM. In RSA, the certificate of node i is denoted as $CERT_i = (cert_i)^{SK} (\mathrm{mod}\, N)$, where SK is the secret key of the master node of node i. According to the requirements of RSA, SK should be coprime to $\phi(N)$, i.e. the Euler function of N. Namely,

$$\gcd(SK, \phi(N)) = 1. \tag{9}$$

In AKM, all the secret keys of virtual nodes are generated during the recursive secret sharing process. In addition, in a purely ad hoc network, each node has the right to choose its ID instead of being predetermined by any entity. That is to say, during the secret sharing process, the secret shares are generated with arbitrary IDs according to the Shamir secret sharing scheme. Therefore, there is a very large possibility that they are not coprime to $\phi(N)$, which has the factor "4".

To handle such problem, we design the scheme based on the difficulty of DLP. Both ElGamal [4] and Digital Signature Standard (DSS) [6] require the computation of the inverse of the secrets, and such an operation is costly. To be more efficient, variants of the Schnorr signature scheme [27] and the signature scheme proposed by Park and Kurosawa [23] are employed to fulfill the task. Here we only show two algorithms based on the Schnorr signature scheme. In the former, nodes i selects a group of $2k - 1$ nodes, and these nodes cooperate to assign node i a certificate. It requires only one round to assign a certificate. In the latter, node i selects a group of k nodes, and these nodes cooperate to assign node i a certificate. It may need more than one round to assign a certificate, when there are malicious nodes launching active attacks.

The first algorithm has been proposed by Stinson and Strobl [29], which is provably secure, i.e., one can break this threshold signature iff one

can break the Schnorr signature scheme. The second algorithm is modified from [29] by us, which is also provably secure. It is very efficient but not fault-tolerant. These two algorithms, especially the second one, are more efficient and secure than Kong et al.'s scheme. First, both of the two algorithms proposed by Kong et al. cannot handle active attacks. It can only verify whether the combined certificate is valid, but fails to find out the invalid partial certificates. Consequently, as long as there is one fake partial certificate among the k partial certificates chosen to generate the certificate, all the work done by other honest nodes are useless. Even worse, adversaries can perform the attack without being caught. It makes Kong et al.'s scheme inefficient for MANET. To prevent such attack, in our scheme nodes can verify each partial certificate to detect those malicious nodes. Second, our algorithms are based on DLP, which is faster than RSA on which Kong et al.'s algorithms are based. Third, the time for generating a partial certificate in our algorithms is 7–150 times shorter than in Kong et al.'s scheme, when there is no communication error. Such advantage is critical in MANET where by nature the less help a node requests from its neighbors, the higher is the chance of obtaining the help. Furthermore, Kong's algorithm requires a k-bounded offsetting to recover the real certificate [16,18,19], while we can generate the real certificate directly.

7.1.1. Assigning certificates based on $2k-1$ nodes

In this algorithm, a VSS scheme is employed to find out adversaries who launch active attacks. Since Pedersen's VSS scheme [24] requires a dealer, it is not suitable for ad hoc environments. Instead, we follow the distributed way proposed in Stinson's scheme [29] to achieve verifiable secure sharing. This algorithm requires cooperation from $2k-1$ nodes, [4] and it ensures that the whole process of assigning a certificate can be finished within one round, in spite of active attacks launched by malicious nodes, since there are at most $k-1$ adversaries.

[4] On consideration of the dynamic property of MANETs, a relative higher number of nodes may need to be involved in this algorithm.

Let p and q be two large primes such that q divides $p-1$, and let G_q is the unique multiplicative subgroup of $\mathbb{Z}_p$ with order q. Let m be the statement claiming that a new node i's public key is PK_i, let $h(\cdot)$ be a one-way hash function: $\{0,1\}^* \to \mathbb{Z}_q$.

First, node i chooses a group of $2k-1$ nodes from its neighbors. Without loss of generality, let the group be $G = \{ID_1, \ldots, ID_{2k-1}\}$. Then node i broadcasts the request m together with the IDs of the $2k-1$ nodes among the group G. To achieve VSS, we need to generate a random shared secret denoted as r within group G. Details is shown as follows.

Once a node $j \in G$ receives the request and decides to serve the request, it chooses $r_j, r'_j \in \mathbb{Z}_q$ at random, and verifiably shares them among G acting as the dealer according to Pedersen's VSS scheme. Let the sharing polynomials be $f_j(u) = \sum_{t=0}^{k-1} a_{jt}u^t$, $f'_j(u) = \sum_{t=0}^{k-1} a'_{jt}u^t$, where $a_{j0} = r_j$, $a'_{j0} = r'_j$. Let the public commitments be $C_{jt} = g^{a_{jt}} \cdot h^{a'_{jt}} (\mathrm{mod}\, p)$ for $t \in \{0, \ldots, k-1\}$, where g and h are two generators of G_q and no one knows $\log_g h$. Let H_0 be the set of nodes which are not detected to be cheating. Then the shared secret r is defined as $r = \sum_{j \in H_0} r_j$, and node j sets its share of the secret as $e_j = \sum_{t \in H_0} f_t(ID_j) (\mathrm{mod}\, q)$, and the value $e'_j = \sum_{t \in H_0} f'_t(ID_j) (\mathrm{mod}\, q)$.

Next, each node j ($\in H_0$) broadcasts $A_{jk} = g^{a_{jt}}$ for $t \in \{0, \ldots, k-1\}$, and each node s in H_0 can verify the values broadcasted by other nodes in H_0 by checking if

$$g^{f_j(ID_s)} = \prod_{t=0}^{k-1} (A_{jt})^{ID_s^t} \pmod p. \tag{10}$$

If the check fails for an index j, node s complains against node j by broadcasting the values $f_j(ID_s)$, $f'_j(ID_s)$ that satisfy

$$g^{f_j(ID_s)} \cdot h^{f'_j(ID_s)} = \prod_{t=0}^{k-1} (C_{jt})^{ID_s^t} \pmod p \tag{11}$$

but do not satisfy Eq. (10). For node i which received at least one valid complaint, other nodes run the reconstruction phase of Pedersen's VSS scheme to compute $r_j, f_j(\cdot)$, A_{jt} for $t = 0, \ldots, k-1$. Therefore, all the players in H_0 set $X_j = g^{r_j}$.

After performing these steps, the following equations hold:

$$X = \prod_{j \in H_0} X_j = g^r \pmod p,$$

$$f(u) = r + a_1 u + \cdots + a_{k-1} u^{k-1},$$

where $a_t = \sum_{j \in H_0} a_{jt}$, for $t \in 1, \ldots, k-1$, and $f(j) = e_j \pmod q$ for $j \in H_0$.

$$C_t = g^{a_t} \quad (t = 0, \ldots, k-1).$$

In [10], the above scheme is proved to be robust under the assumption that $k \leqslant n/2$. In AKM, this assumption is satisfied, since the threshold of a region is normally set to be much less than the number of nodes in this region.

Then for each node $j \in H_0$, reveals its partial certificate

$$\gamma_j = e_j + h(m \| X) \cdot SK_j \pmod q$$

and γ_j can be verified by

$$g^{\gamma_j} = X \cdot \prod_{t=1}^{k-1} C_t^{j^t} \cdot PK_j^{h(m\|X)}$$

for all $j \in H_0$. Let (SK, PK) denote the key pair of the region to which the $2k-1$ nodes belong. Let H_1 denote the set of nodes not detected to be cheating in the above step. After verifying the partial certificates, node i selects an arbitrary subset $H_2 \subseteq H_1$ with $|H_2| = k$. Without loss of generality, we denote $H_2 = \{1, \ldots, k\}$, and compute

$$\sigma = \sum_{j \in H_2} \gamma_j l'_j(0) \pmod q,$$

where $l'_j(0) = \prod_{r=1, r \neq j}^{k} \frac{ID_r}{ID_r - ID_j} \pmod q$. Then node i's signature for m (i.e. $CERT_i$) is the pair (X, σ). Since $\sigma = e + h(m\|X)SK \pmod q$, other nodes can verify the certificate by

$$g^{\sigma} = X \cdot PK^{h(m\|X)} \pmod p.$$

7.1.2. Assigning certificates based on k nodes

Although the algorithm presented Section 7.1.1 can complete the certification service within one round, the computation overhead for achieving VSS is heavy for nodes in MANET. To solve this problem, in this section we present another algorithm which requires lightweight computation. This algorithm needs cooperation from only k nodes, but it may take more than one round to assign a certificate, when malicious nodes are launching active attacks. Although this algorithm is not fault-tolerant, we can distinguish honest and malicious nodes, and those honest nodes are selected directly as members of group G of the next round. Simulation results show that, when there is no communication error, more than 96% of certification renewals can be finished within two rounds, even if malicious nodes launch active attacks. It is much higher than that of Kong et al.'s scheme, which declines very fast when the threshold increases.

Let p and q be two large primes such that q divides $p-1$, let G_q is the unique multiplicative subgroup of $\mathbb{Z}_p$ with order q, and let g be a generator of G_q. Let m be the statement claiming that a new node i's public key is PK_i, let $h(\cdot)$ be a one-way hash function: $\{0,1\}^* \to \mathbb{Z}_q$.

Details of the algorithm for generating a threshold Schnorr signature are shown as follows. First, node i chooses a group of k nodes from its neighbors. Without loss of generality, let the group be $G = \{ID_1, \ldots, ID_k\}$. Then node i broadcasts the request m together with the IDs of the k nodes in group G.

Once a node $j \in G$ receives the request and decides to serve the request, it first chooses a random integer $e_j \in \mathbb{Z}_q$ and broadcasts $x_j = g^{e_j} \pmod p$ and $PK_j = g^{SK_j} \pmod p$ within the group G. Then node j calculates its partial certificate γ_j that is specific to group G

$$\gamma_j = e_j + h(m\|X) \cdot SK_j \cdot l_j(0) \pmod q,$$

where $l_j(0) = \prod_{r=1, r \neq j}^{k} \frac{ID_r}{ID_r - ID_j} \pmod q$ and $X = \prod_{j=1}^{k} x_j \pmod p$, and returns it to node i. Node i can verify the partial certificate as follows:

$$g^{\gamma_j} = x_j \cdot PK_j^{h(m\|X) l_j(0)} \pmod p. \tag{12}$$

If the k partial certificates are valid, node i calculates $\sigma = \sum_{j=1}^{k} \gamma_j$, and its signature for m (i.e. $CERT_i$) is the pair (X, σ). Other nodes and node i can verify the certificate by

$$g^{\sigma} = X \cdot PK^{h(m\|X)} \pmod p, \tag{13}$$

where the public key of the region to which k nodes belong is denoted as PK. In practice, node

i can first verify (X,σ) using Eq. (13). If it is valid, the task is completed. Otherwise, node i then verifies the partial certificates using Eq. (12).

7.1.3. Assigning certificates with higher level assurance

In AKM, all the secret shares originate from the same global secret key. Making use of this property, our scheme can provide the ability to assign certificates with different levels of assurance at relatively small costs. Sections 7.1.1 and 7.1.2 present how to assign a certificate with the lowest level assurance. Both of the two algorithms can be extended to assign certificates with higher-level assurance, but here we just discuss the latter with an example of assigning a certificate with 2-level assurance.

As shown in Fig. 7, the secret key of a virtual node A (i.e. SK_A) is distributed to n virtual nodes $\{B_1, B_2, \ldots, B_n\}$ using a (n,k) threshold scheme, and again SK_{B_j} $(j = 1,2,\ldots,n)$ is distributed to n_j real nodes $\{C_{j1}, C_{j2}, \ldots, C_{jn_j}\}$ with a (n_j,k_j) threshold scheme. For simplicity, we assume that $n_j = n$ and $k_j = k$, for all $j = 1,2,\ldots,n$.

Node i which wants to get a certificate with 2-level assurance first needs to choose k regions, and then choose k real nodes from each of these k regions.

Without losing of generality, let the group of k^2 nodes be $G = \{C_{11},\ldots,C_{jh},\ldots,C_{kk}\}$, which belong to k regions $\{B_1, B_2, \ldots, B_k\}$. Then node i broadcasts the request m together with the node IDs of the k^2 real nodes and k regions among the group G.

Once the request is received, real node C_{jh} which decides to serve the request from node i first chooses a random integer $e_{jh} \in \mathbb{Z}_q$ and broadcasts $x_{jh} = g^{e_{jh}}$ and $PK_{jh} = g^{SK_{jh}}$ within the group G. Then node C_{jh} calculates its additive share γ_{jh} that is specific to group G

$$\gamma_{jh} = e_{jh} + h(m\|X)SK_{jh}l_{jh}(0)\lambda_j(0) \pmod q,$$

where $X = \prod_{j=1}^{k}\prod_{h=1}^{k} x_{jh} \pmod p$, $l_{jh}(0) = \prod_{r=1,r\neq h}^{k} \frac{C_{jr}}{C_{jr}-C_{jh}} \pmod q$, and $\lambda_j(0) = \prod_{r=1,r\neq j}^{k} \frac{B_r}{B_r-B_j} \pmod q$, and returns it to node i. Node i can verify the partial certificate as follows:

$$g^{\gamma_{jh}} = x_{jh} \cdot (PK_{jh})^{h(m\|X)l_{jh}(0)\lambda_j(0)} \pmod p. \tag{14}$$

If all the k^2 partial certificates are valid, node i calculates $\sigma = \sum_{j=1}^{k}\sum_{h=1}^{k}\gamma_{jh}$, and its signature for m (i.e. $CERT_i$) is (X,σ). Other nodes and node i can verify the certificate by

$$g^{\sigma} = X \cdot (PK_A)^{h(m\|X)} \pmod p. \tag{15}$$

In this scheme, to obtain a certificate with b level assurance, node i needs the cooperation of at most k^b honest nodes. Such requirement is reasonable and this scheme is efficient, since in AKM k^b is much smaller than the size of the whole MANET, i.e. n^b. Again, in practice, node i can first verify (X,σ) using Eq. (15). If it is valid, the task is completed. Otherwise, node i then verifies the partial certificates using Eq. (14).

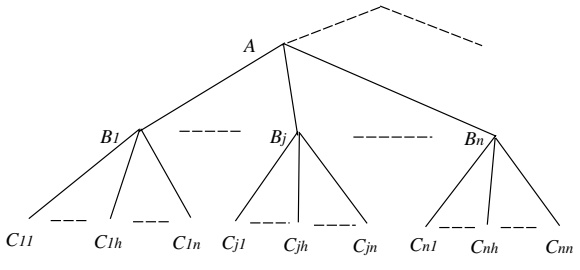

Fig. 7. An example of assigning A 2-level assurance.

7.2. Certificate Revocation

In AKM, Certificate Revocation List (CRL) is based on the accusations from other nodes. In a multi-hop wireless network like MANET, the reliability of an accusation is based on the security of all the nodes that pass and broadcast the accusation. As such, the further an accusation comes from, the higher the probability that this accusation is compromised or malicious is. As a result, in MANET, messages from far away are not as trustworthy as those generated by neighbors (nodes or regions). In addition, large accusation range results in rapidly increased communication and storage overheads. Therefore, the range of the accusations is limited to the same region. If node i receives an accusation on node j, it marks node j as "suspect" when there are less than k accusations towards it. Otherwise, node j is

marked as "compromised" and is added into node i's CRL. In addition, node i launches a request to the k nodes which accused node j to sign a certificate revocation message. For example, in Fig. 2, a $(3,2)$ threshold system is employed within region T, and node E and F accuse on node D. Thus, they can cooperate to generate and send out a message signed by SK_T to revoke node D's certificate.

8. Simulations on security and efficiency

In highly dynamic environment like MANET, small region size may result in rapid variances of the structure of the key tree. On the other hand, if the size is too large, we may have problems in intra-region routing. Current on-demand routing protocols, such as AODV [25] and OLSR [1], handle well when the size of MANET is around 100–250 nodes. Thus, it is suitable to set the region size within this range. Let p_n be the probability of a node being compromised, p_r be the probability of a region being compromised, and n be the size of the region. Table 3 shows the settings on the threshold of a region under different conditions. From the table, we find that, when p_n is not less than 0.01, to ensure p_r lower than 10^{-4}, the threshold of a region should be set to at least 7 and 11 for a region with 100 and 250 nodes, respectively.

8.1. Hierarchical structure vs. flat structure

To show advantages of the hierarchical structure over the flat structure in key management based on threshold cryptography, we compare the probabilities of GSK being compromised in AKM and [16,18,19]. In our implementation, a 3-level structure is employed, and the threshold of any region in AKM is set to be 10. As shown in Fig. 8, under the flat structure, the security of MANET is very weak when the size of the whole network increases. However, under the hierarchical structure, the probability of GSK being compromised is very small and quite stable in spite of the great size of MANET.

Table 3
Settings on the threshold of a region

p_n	$n = 100$		$n = 250$	
	$p_r < 10^{-4}$	$p_r < 10^{-5}$	$p_r < 10^{-4}$	$p_r < 10^{-5}$
0.01	7	8	11	13
0.05	16	17	28	30
0.1	24	26	45	48

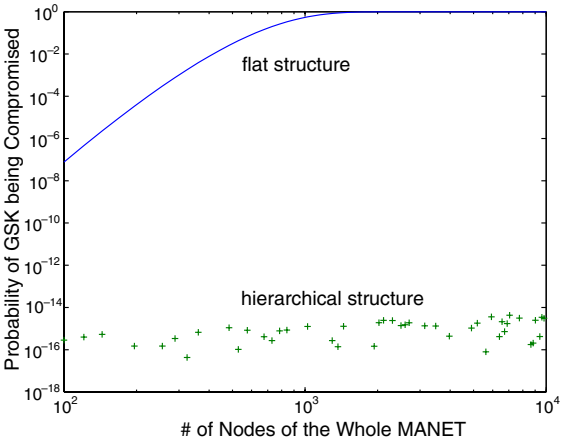

Fig. 8. Flat structure vs. hierarchical structure.

8.2. Regions with the same RTC

As shown in Fig. 9, the higher the RTC is, the smaller p_r is. Therefore, it is suitable to use RTC as an approximate index of the security condition of a region. In addition, the higher the RTC is, the

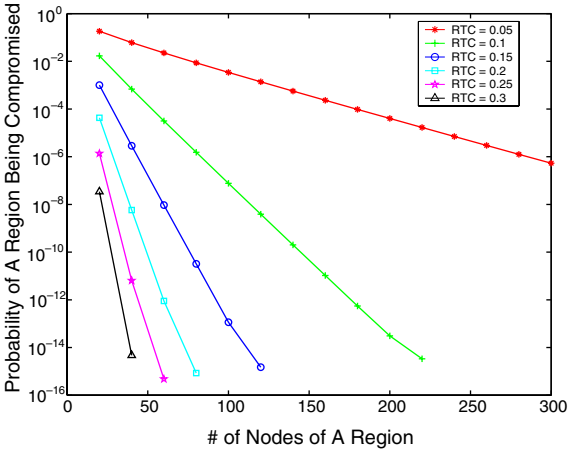

Fig. 9. P_r with fixed RTC.

faster p_r decreases. Due to these two properties, a higher RTC is good for the sake of security. However, on the other hand, it results in a higher threshold which requires more computation power.

In addition, we find that, when both the RTC and the region size are small, this region is easy to be compromised. For example, in Fig. 9, given that the p_n is 0.01 and the RTC of a region is 0.05, p_r may be higher than 0.01 when the size of the region is less than 75. Consequently, we need to be aware of this during the generation of a region. In other words, when the number of initial nodes in a region is small, the threshold should be set to a relative high value.

8.3. Computational costs of region-based operations

Since both "Merge" and "Contraction" operations can be viewed as a series of "Join" operations, our simulation focuses on "Partition" and "Expansion" operations. We run the simulation on a Pentium III 800 laptop.

Table 4 shows the computation cost of "Partition" and "Expansion" operations under different ORS and threshold. In the table, GPSS stands for time for generating partial secret shares for all the nodes in the newly generated region, while GNSS stands for time for generating the new secret share. Simulation results show that the computation cost of both "Partition" and "Expansion" operations is quite small under common threshold and region size settings. For example, when the ORS of a region is 100 and its threshold is 15, it only takes 31 ms to complete the "Partition" operation. As to the "Expansion" operation, the whole cost is less than 8 ms.

8.4. Computational costs of certification services

We measure and compare the performance of Kong's algorithm and our second algorithm on two Windows 2000 machines. One is a Pentium IV 2.2G desktop, and the other is a Pentium III 800 laptop. Due to the limited space, we only show the implementation results on the Pentium III 800 laptop. Both of algorithms are implemented in Java. We run the experiments under different settings, e.g. different key lengths, thresholds, and region sizes. For each setting, we run the two algorithms for 20 times, respectively, and then calculate the average values.

Here, the time for generating or renewing a certificate is calculated as the sum of all the processes taken by nodes, including the requesting node and those neighbors that provide certificate services, as described in the algorithms. We argue that, in MANET, due to the limitation on computation power, computational costs on nodes are more important for providing key management successfully. Furthermore, the main objective of this simulation is to show that compared to Kong's algorithm our second algorithm has much looser requirements on the computational power of nodes. Therefore, we do not consider the communication overhead, when we compare the two algorithms.

Table 4
Computation cost of "partition" and "expansion" operations

ORS	Threshold	"Partition" Operation (ms)			"Expansion" Operation (ms)		
		GPSS	GNSS	Total	GPSS	GNSS	Total
100	5	8.87	0.03	8.90	1.59	0.01	1.60
100	10	18.59	0.11	18.70	4.52	0.02	4.54
100	15	30.96	0.28	31.24	7.91	0.02	7.93
100	20	46.24	0.60	46.84	11.69	0.02	11.71
100	25	62.00	0.80	62.80	16.42	0.03	16.45
250	10	47.63	0.11	47.74	11.00	0.03	11.03
250	20	124.20	0.68	124.88	27.67	0.03	27.70
250	30	205.27	1.29	206.56	54.71	0.07	54.78
250	40	302.27	2.24	304.51	90.16	0.07	90.23
250	50	422.73	3.77	426.50	135.78	0.07	135.85

As shown in Fig. 10, the total time for generating or renewing a certificate in our second algorithm varies from 20 to 250 ms in the experiments, depending on the setting on the key length, threshold, and region size. In particular, when the key length is 1024 bits, it takes our second algorithm around 40–70 ms to generate or renew a certificate. We also find that, in our second algorithm, the process that a neighbor generates a partial certificate for the new node is very fast. As shown in Fig. 11, such process takes less than 32 ms under all the settings tested in the experiments. In particular, when the key length is 1024 bits, it takes less than 9 ms to generate a partial certificate.

To compare with previous work [16,18,19], in Figs. 12 and 13, we show the ratio of the total time for generating or renewing a certificate in Kong's algorithm (denoted as $T_{C\text{-}KONG}$) to that of our second algorithm (denoted as $T_{C\text{-}OUR}$) and the ratio of the time for generating a partial certificate in Kong's algorithm (denoted as $T_{PC\text{-}KONG}$) to that of our second algorithm (denoted as $T_{PC\text{-}OUR}$), respectively. As shown in Fig. 12, our second algorithm is more efficient, when we consider the total time for assigning a new certificate. For instance,

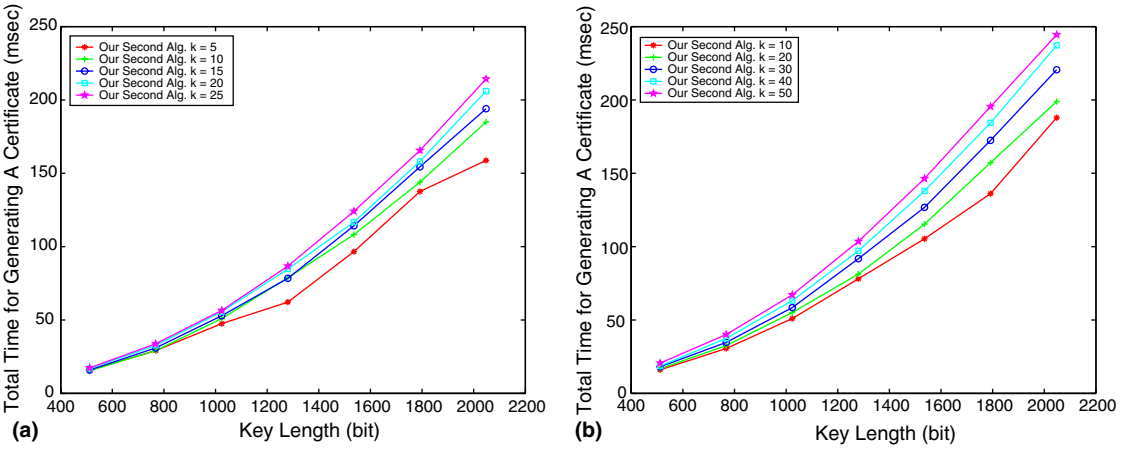

Fig. 10. Total time for generating or renewing a certificate in our second algorithm: (a) $n = 100$ and (b) $n = 250$.

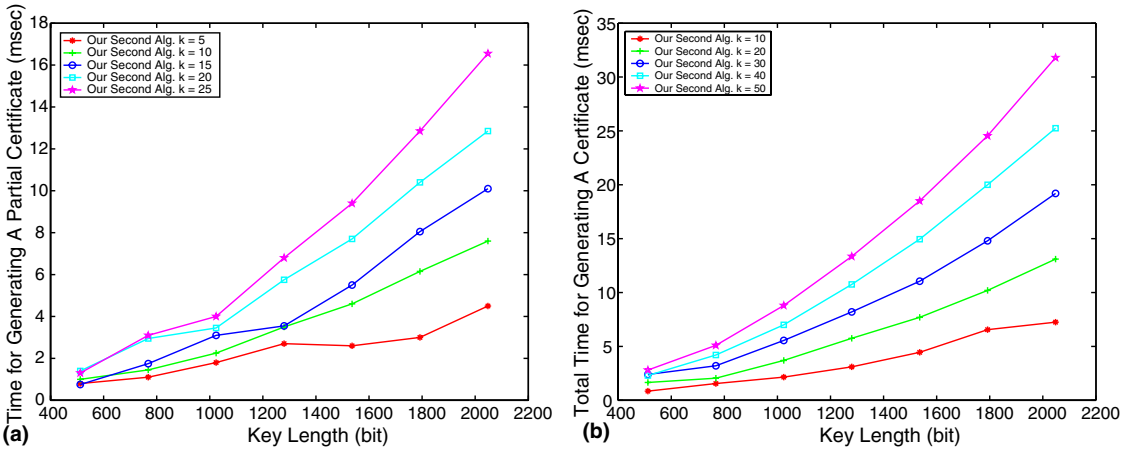

Fig. 11. Time for generating a partial certificate in our second algorithm: (a) $n = 100$ and (b) $n = 250$.

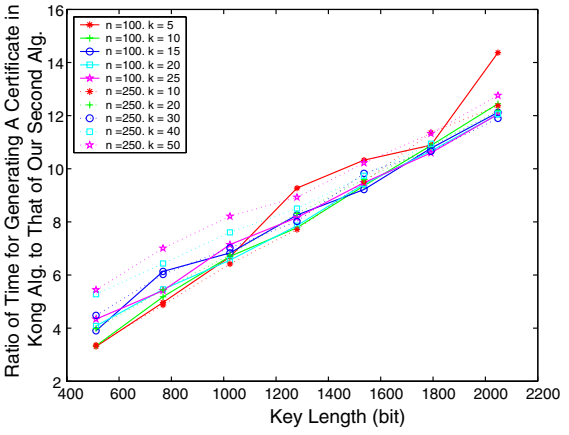

Fig. 12. Ratio of $T_{C\text{-}KONG}$ to $T_{C\text{-}OUR}$.

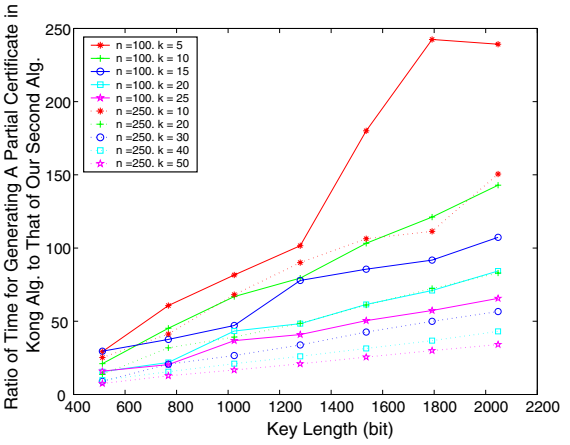

Fig. 13. Ratio of $T_{PC\text{-}KONG}$ to $T_{PC\text{-}OUR}$.

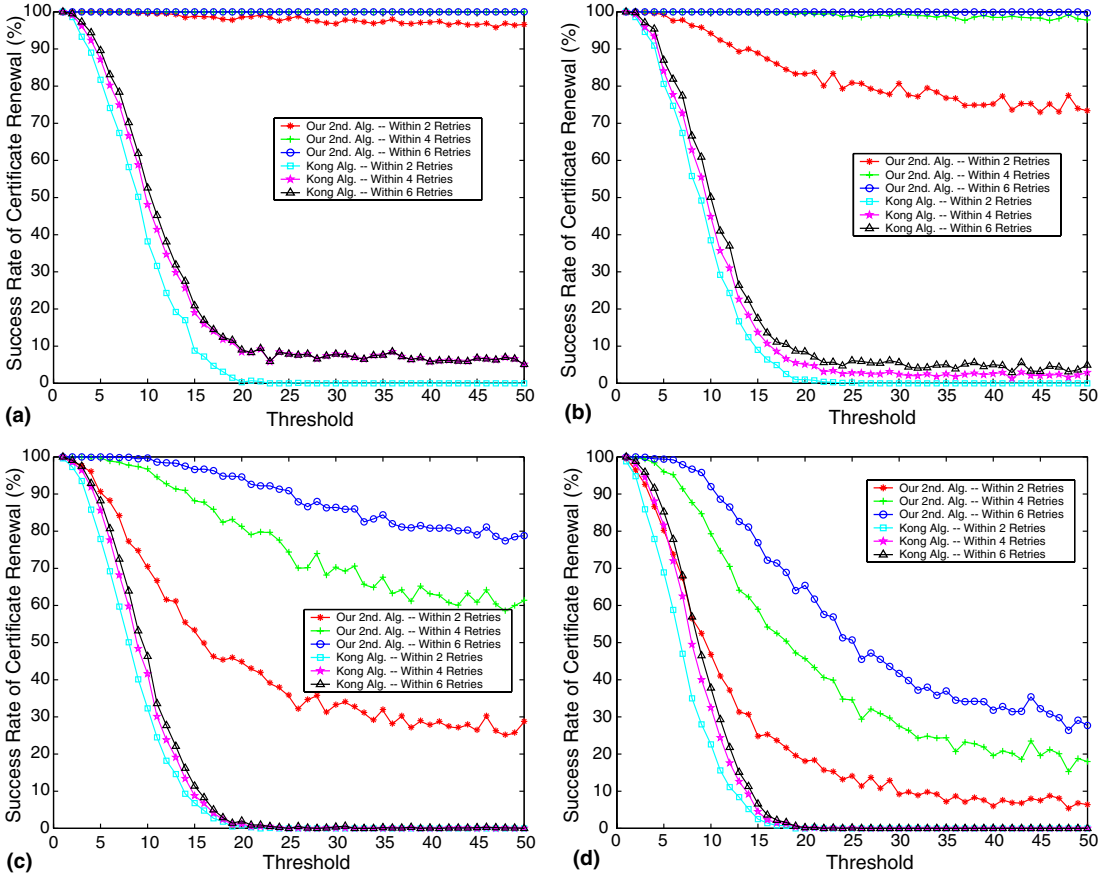

Fig. 14. Success rate of certification renewals—$n = 100$: (a) error rate = 0%; (b) error rate = 1%; (c) error rate = 5% and (d) error rate = 10%.

when the key length is 1024 bits, our second algorithm is around six to eight times faster than Kong's algorithm. As to the process of generating a partial certificate, the efficiency is greatly improved in our second algorithm. For example, when the key length is 1024 bits, our second algorithm is around 20–80 times faster. Consequently, using our second algorithm, a node can easily find enough neighbor nodes to provide the certification service, since very little effort is involved.

From empirical results, we notice that the performance of our algorithm is tightly related to the key length and the threshold. The larger the key length is, the more time we need to complete the certification service. However, compared to Kong et al.'s scheme, our second scheme is less sensitive to this parameter, and thus is more efficient. Similarly, the larger the threshold is, the more time we need to complete the certification service.

8.5. Certification services under active attacks

To compare the efficiency of certification services under active attacks, we use the success rate of certification renewals within certain rounds and the average rounds of retries before successfully assigning or renewing a certificate as the evaluation metrics. Here, we denote the success rate of certification renewals within r rounds and the wireless channel error rate as SR_r and e, respectively.

We run the simulation in a 600 m × 600 m network with 100 or 250 nodes, and the speed of

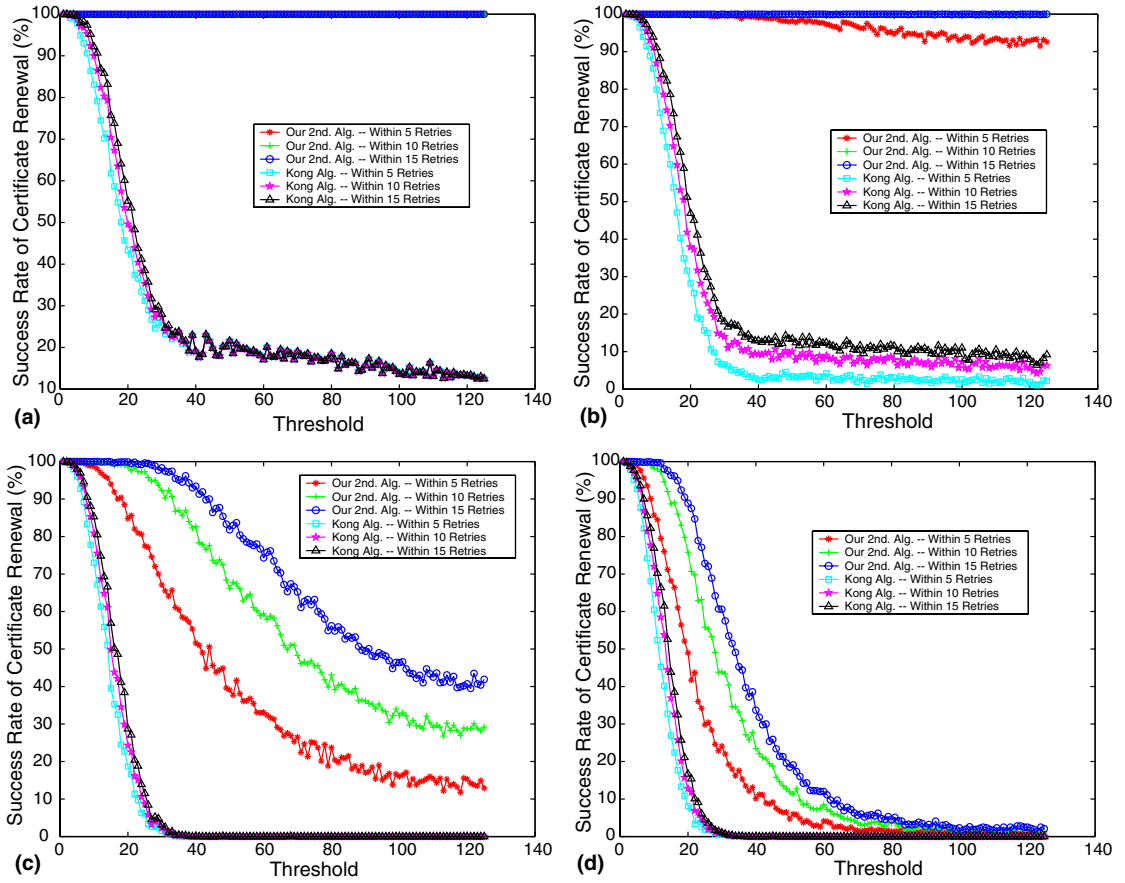

Fig. 15. Success rate of certification renewals—$n = 250$: (a) error rate = 0%; (b) error rate = 1%; (c) error rate = 5% and (d) error rate = 10%.

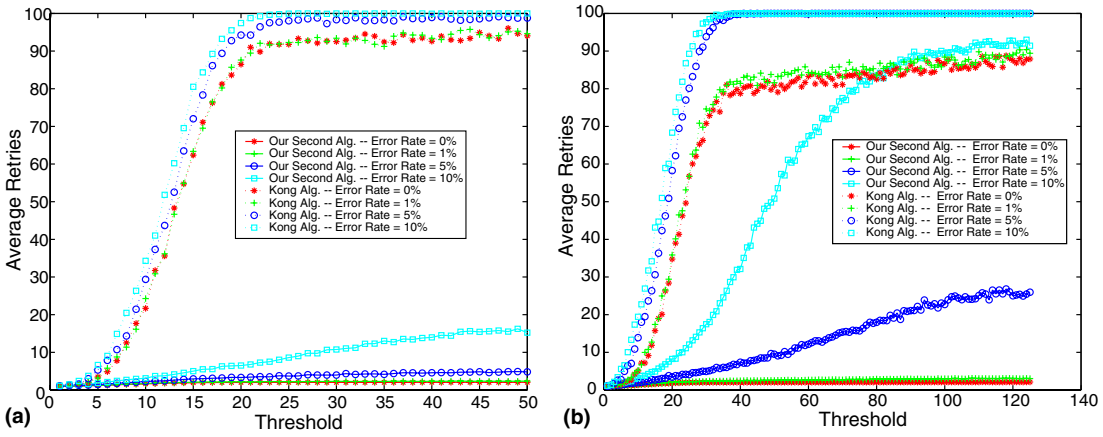

Fig. 16. Average retries of certification renewals: (a) $n = 100$ and (b) $n = 250$.

nodes ranges from 1 m/s to 20 m/s. The random way-point model is applied to emulate node mobility pattern. In the simulation, we consider the scenarios of different wireless channel error rates, from no error (0%) to high error rate (10%). For each scenario, if the certification renewal fails due to either active attacks or communication errors, the node which requests the certification service would replay the request up to a maximum number of retries. Typically, we set the maximum number of retries to 100 in the simulation.

As shown in Figs. 14 and 15, in all the simulation models, the success rate of Kong's algorithm declines very quickly when the threshold increases. For example, even when there is no communication error, for a region with size 100 and p_n is 0.01, if the threshold of this region is set to be 5, SR_2 is 81.7%. However, if the threshold increases to 10, it drops to 38.2%. In contrast, in our second algorithm, SR_2 decreases only 0.2%, i.e. from 100% to 99.8%, while the threshold increases from 5 to 10.

As shown in Figs. 14 and 15, the success rate of our second algorithm is always higher than that of Kong's algorithm. However, we also notice that in our second algorithm, the higher the wireless channel error rate is, the faster the success rate of certification renewals declines. More specifically, our second algorithm works well under low wireless channel error rates, e.g. $e = 1\%$. When e increases (e.g. under heavy communications), SR_r is more sensitive to the variations on the threshold. For instance, as shown in Fig. 14, when $e = 1\%$ and $n = 100$, SR_2 decrease by 5.4%, while the threshold increases 5 from 10. However, when $e = 10\%$ and $n = 100$, for the same variation of the threshold, SR_2 decreases by 33.4% instead. In such cases, to improve the success rate, we need to either choose a small threshold or increase the number of retries. The former is more effective. In addition, in AKM, to ensure the security, the RTC of one region should not be less than GTC. Therefore, we need to limit the size of a region, when the wireless channel error rate is high.

Fig. 16 shows the average retries of certification renewals in Kong's algorithm and our second algorithm under different wireless channel error rates. In all the cases, compared to our second algorithm, Kong's algorithm takes more rounds to complete the certification renewal. Similar to the experimental results on the success rate of certification renewals, in our second algorithm, the higher the wireless channel error rate is, the faster the average retries of certification renewals raises when the threshold of the region increases.

9. Conclusion

To overcome those challenges (e.g. security, efficiency, flexibility, and adaptivity) in key management in large MANETs, in this paper, we

provided approaches aiming at both the architecture level and the algorithm level. At the architecture level, to ensure flexibility and adaptivity, a new key management scheme based on the hierarchical structure and secret sharing was proposed to distribute cryptographic keys and provide certification services; at the algorithm level, to resist active attacks, we proposed two algorithms which are based on threshold cryptography and VSS, and can be used independently from the key management scheme proposed. Simulation results show that, compared to previous work [16,18,19], our scheme is not only much faster, but it also can resist active attacks which are difficult to defend using existing schemes. Another major contribution of our work is that certificates with different levels of assurance can be issued with the coalition of a relatively small number of nodes. Furthermore, our scheme can isolate the compromised regions and provide stronger protection to the Global Secret Key (GSK) compared to flat-structured schemes.

References

[1] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot, INRIA Rocquencourt. Optimized link state routing protocol, September 2001. Internet draft, draft-ietf-manet-olsr-06.txt.

[2] Y. Desmedt, Threshold cryptography, European Transactions on Telecommunications 5 (4) (1994) 449–457.

[3] Y. Desmedt, Y. Frankel, Threshold cryptosystems, in: Proceedings of Advances in Cryptology-CRYPTO'89, LNCS 0435, 1990, pp. 307–315.

[4] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory 31 (1985) 469–472.

[5] P. Feldman, A practical scheme for non-interactive verifiable secret sharing, in: Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science, 1987, pp. 427–437.

[6] National Institute for Standards and Technology. Digital signature standard (DSS), 1998. FIPS 186-1.

[7] Y. Frankel, Y.G. Desmedt, Parallel reliable threshold multi-signature, Technical Report TR-92-04-02, Department of EECS, University of Wisconsin-Milwaukee, 1992.

[8] Y. Frankel, P. Gemmel, P. MacKenzie, M. Yung, Optimal resilience proactive public-key cryptosystems, in: Proceedings of the 38th Symposium on Foundations of Computer Science, 1997.

[9] Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung, Proactive RSA, in: Advances in Cryptology–Crypto'97, LNCS 1294, 1997, pp. 440–454.

[10] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Secure distributed key generation for discrete-log based cryptosystem, in: Eurocrypt '99, 1999, pp. 295–310.

[11] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, Proactive public-key and signature schemes, in: Proceedings of the Fourth Annual Conference on Computer Communications Security, 1997, pp. 100–110.

[12] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, Proactive secret sharing or: how to cope with perpetual leakage, in: Advances in Cryptology—Crypto '95, LNCS 963, 1995, pp. 457–469.

[13] S. Jarecki, Proactive secret sharing and public key cryptosystems, Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, September 1995.

[14] A. Khalili, J. Katz, W. Arbaugh, Toward secure key distribution in truly ad hoc networks, in: IEEE Workshop on Security and Assurance in Ad Hoc Networks, in Conjunction with the 2003 International Symposium on Applications and the Internet, 2003.

[15] J. Kong, H. Luo, K. Xu, D.L. Gu, M. Gerla, S. Lu, Adaptive security for multi-layer ad hoc networks, Wireless Communications and Mobile Computing 2 (5) (2002) 533–547.

[16] J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang, Providing robust and ubiquitous security support for mobile ad hoc networks, in: IEEE 9th International Conference on Network Protocols (ICNP'01), 2001.

[17] B. Lehane, L. Doyle, D. O'Mahony, Shared RSA key generation in a mobile ad hoc network, in: Proceedings of IEEE Military Communications Conference (MILCOM 2003), 2003.

[18] H. Luo, J. Kong, P. Zerfos, S. Lu, L. Zhang. Self-securing ad hoc wireless networks, in: Seventh IEEE Symposium on Computers and Communications (ISCC'02), 2002.

[19] H. Luo, J. Kong, P. Zerfos, S. Lu, L. Zhang, URSA: ubiquitous and robust access control for mobile ad hoc networks, IEEE/ACM Transactions on Networking 12 (6) (2004) 1049–1063.

[20] P. McDaniel, S. Jamin, A scalable key distribution hierarchy, Technical Report CSE-TR-366-98, Electrical Engineering and Computer Science, University of Michigan, 1998.

[21] M. Narasimha, G. Tsudik, J.H. Yi, On the utility of distributed cryptography in P2P and MANETs: The case of membership control, in: Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03), November 2003, pp. 336–345.

[22] P. Papadimitratos, Z.J. Haas, Secure routing for mobile ad hoc networks, in: SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), January 2002.

[23] C. Park, K. Kurosawa, New ElGamal type threshold digital signature scheme, IEICE Trans. Fundamentals Special Section on Cryptography and Information Security E79-A (1) (1996) 86–93.

[24] T. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in: Advances in Cryptology–Crypto'91, LNCS 576, 1992, pp. 129–140.
[25] C.E. Perkins, E.M. Royer, Ad hoc on-demand distance vector routing, in: WMCSA'99, 1999.
[26] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, E.M. Belding-Royer, A secure routing protocol for ad hoc networks, in: Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP), 2002.
[27] C.P. Schnorr, Efficient signature generation for smart cards, Journal of Cryptology 4 (3) (1991) 239–252.
[28] A. Shamir, How to share a secret, Communications of the ACM 22 (11) (1979) 612–613.
[29] D.R. Stinson, R. Strobl, Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates, in: ACISP 2001, LNCS 2119, 2001, pp. 417–434.
[30] S. Čapkuny, L. Buttyán, J.-P. Hubaux, Self-organized public-key management for mobile ad hoc networks, Technical Report 2002/34, EPFL/IC, May 2002.
[31] S. Yi, R. Kravets, MOCA: mobile certificate authority for wireless ad hoc networks, in: 2nd Annual PKI Research Workshop Program (PKI 03), 2003.
[32] Y. Zhang, W. Lee, Intrusion detection in wireless ad hoc networks, in: Proceedings of The Sixth International Conference on Mobile Computing and Networking (MobiCom 2000), August 2000.
[33] L. Zhou, Z.J. Haas, Securing ad hoc networks, IEEE Network, Special Issue on Network Security 13 (6) (1999) 24–30.

Bo Zhu is a PhD. candidate with the School of Computing at National University of Singapore. He received the MSc in information system from National University of Singapore in 2002, the M.Eng and the B.Eng in Automation from Wuhan University of Hydraulic and Electric Engineering (currently Wuhan University) in 1999 and 1996, respectively. His research interests include security in ad hoc network and wireless network, secret sharing, key management, authentication, system security, and network security.

Feng Bao received his B.Sc and M.Sc from Beijing University, China, in 1984 and 1986, respectively. He was an Assistant and then Associate Professor with the Institute of Software, Chinese Academy of Sciences, China, from 1986 to 1994. He obtained his PhD from Gunma University, Japan, in 1996. He is currently a Leading Scientist with the Institute for Infocomm Research, Singapore. He has more than 80 publications in the areas of cryptography and security systems. He has served on many program committees of international conferences. He is the Program Chair of the 2004 International Workshop on Practice and Theory in Public Key Cryptography.

Robert H. Deng received his B.Eng from National University of Defense Technology, China, and his M.Sc and PhD from Illinois Institute of Technology. He is currently Professor, School of Information Systems, Singapore Management University. Prior to this, he was Principal Scientist and Manager of Infocomm Security Department, Institute for Infocomm research. He has more than 140 publications in the areas of error-control coding, digital communications, computer networks, information security, and network and distributed systems security. He served on many advisory and program committees of international conferences. He served as Program Chair of the 2002 International Conference on Information and Communications Security. He also served as the General Chair of the 2004 International Workshop on Practice and Theory in Public Key Cryptography.

Mohan S. Kankanhalli obtained his B-Tech (Electrical Engineering) from the Indian Institute of Technology, Kharagpur, in 1986 and his MS and PhD (Computer and Systems Engineering) from the Rensselaer Polytechnic Institute in 1998 and 1990, respectively. He then joined the Institute of Systems Science (ISS-now Institute for Infocomm Research) in Singapore in October 1990. He mainly worked on content-based multimedia information retrieval in the multimedia group. He left ISS in June 1997 and spent the 1997–1998 academic year at the Department of Electrical Engineering of the Indian Institute of Science, Bangalore. He has been with Department of Computer Science of the School of Computing at the National University of Singapore since May 1998. His current research interests are in Multimedia Information Systems (image/audio/video content processing, multimedia retrieval) and Information Security (multimedia watermarking, image/video authentication). More details are available at: http://www.comp.nus.edu.sg/~mohan.

Guilin Wang is currently a research scientist with the Institute for Infocomm Research, Singapore. He received his Ph.D. degree in computer science from the Institute of Software, Chinese Academy of Sciences, China, in March 2001. His research interests include the analysis, design and application of digital signature, secret sharing, and cryptographic protocol, etc.