Reliable and fully distributed trust model for mobile ad hoc networks

Mawloud Omar, Yacine Challal, Abdelmadjid Bouabdallah

Fully Distributed Trust Model based on Trust Graph for Mobile Ad hoc Networks

Mawloud Omar, Yacine Challal, and Abdelmadjid Bouabdallah

ReSyD, Bejaia University, Algeria. mawloud.omar@gmail.com
Heudiasyc Lab., Compiegne University of Technology, France. ychallal@hds.utc.fr
Heudiasyc Lab., Compiegne University of Technology, France. bouabdal@hds.utc.fr

Abstract

A mobile ad hoc network (MANET) is a wireless communication network which does not rely on a pre-existing infrastructure or any centralized management. Securing the exchanges in MANETs is compulsory to guarantee a wide spread development of services for this kind of networks. The deployment of any security policy requires the definition of a trust model that defines who trusts who and how. Our work aims to provide a fully distributed trust model for mobile ad hoc networks. In this paper, we propose a fully distributed public key certificate management system based on trust graphs and threshold cryptography. It permits users to issue public key certificates, and to perform authentication via certificates' chains without any centralized management or trusted authorities. Moreover, thanks to the use of threshold cryptography; our system resists against false public keys certification. We perform an overall evaluation of our proposed approach through simulations. The results indicate out performance of our approach while providing effective security.

Key words: MANETs, Security, Trust Model, Trust Graph, Threshold Cryptography.

1 Introduction

A mobile ad hoc network (MANET) [19] comprises a group of wireless independent nodes which temporarily forms a network without pre-existing infrastructure; all networking operations (routing, mobility management, and so on) are performed by the nodes themselves. This emerging technology seeks to provide "anytime, anywhere" networking services for mobile users. With the proliferation of mobile computing and communication devices, mobile ad hoc networking is predicted to be a key technology for the next generation of wireless communications [8]. They are mostly desired in military applications where their mobility is attractive but where their insecurity continues to slow down more widespread take-ups.

Operation in an ad hoc network introduces new security problems: ad hoc networks are generally more prone to physical security threats. The possibility of eavesdropping, spoofing, denial-of-service, and impersonation attacks increases [4]. Similar to fixed networks, security of the ad hoc networks is considered from different points such as availability, confidentiality, integrity, authentication, non-repudiation, access control and usage control [30,28]. However, security approaches used to protect the fixed networks are not feasible due to the salient characteristics of the ad hoc networks. New threats, such as attacks raised from internal malicious nodes, are hard to defend [5]. In the literature, there are several manners to introduce security in mobile ad hoc networks that can be classified into two main approaches:

(1) Models based on TTP (Trusted Third Party) where certificates and/or keys are issued by a single authority (or a group of special servers), like PKI (public key Infrastructure) [20,30] and Kerberos [11,21].
(2) Through full self-organization, where security does not rely on any trusted authority or fixed server, like models based on trust propagation through a trust graph, such as PGP [1].

Our proposal is based on the second approach, and we propose *a fully distributed trust model based on trust graph for mobile ad hoc networks*, that allows nodes to generate, store, and distribute their public key certificates without any central server or trusted party. In our system, all the nodes have a similar role, and we do not assign any special functions to a subset of nodes. The main motivation for employing this approach comes from the self-organized nature of MANETs and from the need to allow users to fully control the security settings in the network. In our system, like in PGP [1], users' public/private keys are created by the users themselves, and key authentication is performed via chains of public key certificates in the graph of trust. Also, like in [3], instead of storing certificates in centralized certificate repositories, certificates in our system are stored and distributed by nodes themselves. Our main con-

tribution is the inclusion of a threshold scheme within the graph of trust, in order to resist against false public key certificates issued by malicious nodes in the network. During network initialization, nodes share a private key, and each node holds one private share. Instead of using private keys for certificates signing, nodes will use their private shares. Our solution is developed for open networks, in which nodes can join/leave the network without any centralized administration. The joining operation is performed by a coalition of member nodes to allow access to a new node.

The remaining of this paper is structured as follows: in section 2, we give an overview of threshold cryptography. In section 3, we introduce the related works. In sections 4 and 5, we give detailed description and analysis of our trust model. In section 6, we describe and discuss simulation results. We finally conclude this work in section 7.

2 Threshold Cryptography: a Background

A (t, n) threshold scheme ($t \leq n$) is a cryptographic technique that allows to hide a secret S in n different shares S_i ($1 \leq i \leq n$), so that the knowledge of at least t shares is required to recover the initial secret S. Let us illustrate this technique with the following famous scheme: Shamir's threshold scheme [25] is based on polynomial interpolation, and the fact that a univariate polynomial $y = f(x)$ of degree $t-1$ is uniquely defined by distinct t points (x_i, y_i).

Setup: The trusted party T begins with a secret integer $S \geq 0$ it wishes to distribute among n users:

(1) T chooses a prime $p > max(S, n)$, and defines $f(0) = a_0 = S$.
(2) T selects $t-1$ random, independent coefficients $a_1, ..., a_{t-1}$, $0 \leq a_j \leq p-1$, defining the random polynomial over Z_p, $f(x) = \sum_{j=0}^{t-1} a_j x^j$.
(3) T computes $S_i = f(i)\ mod\ p$, $1 \leq i \leq n$ (or for any n distinct points i, $1 \leq i \leq p-1$), and securely transfers the share S_i to user P_i, along with public index i.

Recovering the secret: To recover the initial secret S, a subgroup of at least t users should exchange their shares. After the exchange, each user of the subgroup will get t distinct points (i, S_i) of the polynomial f. These t points allow to calculate the coefficients of the polynomial f using the Lagrange interpolation as follows:

$$f(x) = \sum_{i=1}^{t} S_i \prod_{1 \leq j \leq t, j \neq i} \frac{x-j}{i-j}$$

Since $f(0) = a_0 = S$, the shared secret may be expressed as:

$$S = \sum_{i=1}^{t} c_i S_i \, , \; where \;\; c_i = \prod_{1 \leq j \leq t, j \neq i} \frac{j}{j - i}$$

3 Related Works

In this section we survey the most interesting public key-based trust models in MANETs, which we classify into two categories: partially and completely distributed models, as illustrated in figure 1.

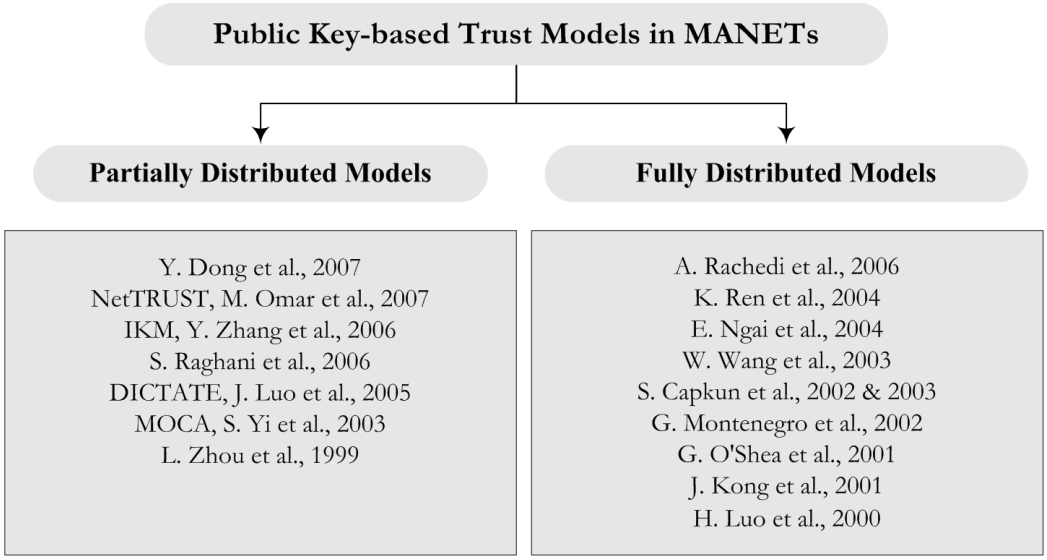


Fig. 1. Taxonomy of public key-based trust models.

3.1 Partially Distributed Models

L. Zhou et al. [30] proposed a partially distributed certification authority (CA) relying on threshold cryptography. The CA is distributed among particular nodes: servers, combiners, and a dealer. Servers and combiners sign public key certificates for users. The dealer is a special server which knows the CA's private key. For any joining node, if all partial signatures are collected, it can then compute the complete signature locally to obtain the complete public key certificate. Recently, S. Raghani et al. [23] proposed a similar solution, in which they allow to dynamically adjust the value of the threshold when required, and thereby reduces the certification delays.

S. Yi et al. [27] follows the same direction by building a distributed CA based on threshold cryptography. They improve security by secure and power selected nodes as MOCA servers (MObile Certification Authority) and reduce communication overhead by caching routes to MOCA servers. The system uses unicast instead of flooding when sufficient cached routes exist.

J. Luo et al. [14] proposed DICTATE (Distributed CerTification Authority with probabilisTic frEshness for ad hoc networks). DICTATE uses a hierarchical CA between one mCA (mother CA) in wired network, and a group of dCA (distributed CA) in ad hoc network. Nodes in ad hoc network can collectively be isolated from the mCA, but always have the need for CA's services. The mCA delegates a group of dCA during the isolation period in order to ensure the availability of security services.

Y. Zhang et al. [29] proposed an ID-based key management system using threshold cryptography. The system is a "certificateless"-based model in which nodes' public keys are directly derivable from their known identifiers (IDs) plus some common information, and eliminates the need to certificate public key distribution. The system introduces a novel construction method of ID-based public/private keys, which ensures tolerance against compromised nodes.

Y. Dong et al. [6] proposed a CA cluster-based architecture. The system organizes the network into clusters. Each cluster head (CH) maintains a CA information table (CIT), which contains a list of CA nodes in its local cluster and in the other clusters. The distributed CA information is managed among CHs, which reduces service response delay and system overhead.

M. Omar et al. [17] proposed NetTRUST (mixed NETworks Trust infrastRUcture baSed on Threshold cryptography). NetTRUST uses two particular CAs, that ensure public key management: central CAs (CCA) in wired network, and mobile CAs (MCA) in ad hoc network. MCA servers emulate the CA role by using a k-threshold cryptography scheme, and the CCA servers delegate the CA role to MCA servers by using a t-threshold cryptography scheme. The system is decentralized, supports nodes' mobility, and resist against MCA's failures.

3.2 Fully Distributed Models

PGP (Pretty Good Privacy) [1] is a completely distributed model which was created, by P. Zimmermann, initially for the Internet in order to secure emails. PGP is based on referral certification, which allows multiple users to recommend a certain user by signing certificates of its public key. PGP adopts a system, called *web of trust*. It consists of an establishment of a distributed public key management. The principle interest in this model lies in the ab-

sence of a central authority. However, this model is not perfectly secure because dishonest users may issue false certificates to cheat other users.

Based on the same principles, S. Capkun et al. [2,3] proposed a self-organized trust model for MANETs, in which trust among nodes is constituted through physical contact. In this model every node issues public key certificates to those it trusts from its own domain. Nodes can authenticate each others with chains of trust. They have also developed detailed algorithms for their model to facilitate the initialization and authentication process, and nodes are assumed to store as many certificates as possible. In this model, trust establishment is coming from "offline trust relationships", which are generated from general "social relationships".

K. Ren et al. [24] proposed a modified version of S. Capkun, and they introduced a boot server in order to initialize the system. The server computes and distributes to each node a list of bindings (nodes' identifiers and public keys) and each of them generates the corresponding certificates [1]. Thus, a web of trust relationships is formed, and the system became fully distributed, in which nodes authenticate themselves through certificates chains.

E. Ngai et al. [16] proposed another direction based on web of trust approach proposed in PGP, in which nodes act as CAs without any TTP. The system organizes the network into clusters, such that nodes are divided into different groups with unique identifiers. Nodes in the same group are assumed to know each other, where each node monitors and keeps a trust table for storing trust values (defined as a continuous value $\in [0, 1]$ interval) for the behavior of its group members [2]. Therefore, the protocol of certification between two nodes will be executed when both nodes belong to different groups. When a node n_1 needs to request for public key of a target node n_2, n_1 should request for certificates signed with some "introducing" nodes in the same group of n_2. Then, each "introducer" replies to n_1 with the public key and the trust value of n_2. The trust values are involved in the calculation of the final trust value of n_2 at n_1 with a specific formula.

A. Rachedi et al. [22] proposed a similar distributed clustering-based model which relies on trust values metric and behavior monitoring. However, each cluster is supervised by a cluster head (CH) which considered as CA in the cluster. The trust relationship is ensured by CAs among clusters (instead of "introducers" in [16]), where a CA can recommend a node, with certain trust level, belonging to its cluster to another CA. They improve the security of CAs in each cluster by using dispensable confident nodes, called *registration authorities* (RA). The role of RAs is to protect CA, by receiving requests of

[1] Lists are not assumed to include the same list of bindings.
[2] Monitoring behaviors of nodes such as security collaboration, correct packets forwarding, and so on [22].

certification, filtering and treating these demands before forwarding them to the CA.

H. Luo et al. [13] proposed a fully distributed CA based on threshold cryptography. The system distributes shares to all entities at the time when they join the network. Trust is established by the assumption that all the nodes must supervise the direct neighbors' behavior, and maintain their own certificate revocation list (CRL). If a node discovers that one of its neighbors is incorrect, it adds its certificate to the list of revocations and diffuses through the network an "accusation". If the certificate of accusatory is revoked, the accusation is ignored. Otherwise, the node is marked to be suspect by all the nodes receiving the accusation.

J. Kong et al. [12] proposed a distributed CA based on threshold cryptography. A node receives its public key from its k neighboring nodes. The authors supposed that all the nodes in the network need to be bootstrapped with their public key certificates from a trusted central management. When a new node needs to get its certificate, it sends a request to its k neighboring nodes for partial certificates. If the coalition considers that the requesting node is a "well-behaved node", they issue their partial certificates, which are then combined together by the target node to issue the complete certificate using an interpolation function.

W. Wang et al. [26] proposed a distributed CA based on threshold secret sharing. In order to handle heterogeneous CAs, each node maintains a list of CAs that it trusts. When a node requires to authenticate another node, they start by exchanging CAs lists. Then, they compare the both lists to check if there are some common CAs, and if so, they proceed to exchange the certificate signed with the common CA. If the two nodes haven't common CAs, then they try to search in their one-hop and two-hop neighbors.

Two other approaches, both originally designed for the address ownership problem in Mobile IPv6, are proposed by G. Montenegro et al. [15] and G. O'Shea et al. [18]. The main idea behind these approaches is to avoid certificates altogether and bind the IP addresses of a node to its public key by deriving the former from the latter in a cryptographically verifiable way. The public key is hashed, and then the hash value is used as part of the IP address of the node. This approach makes sense at the network layer, where IP addresses are handled by machines. But, it does not seem to be applicable at the application layer, where names often refer to and are processed by people, and cannot thus be computed by hashing a public key. If the public key of a node is compromised, then its revocation requires the node to change its IP address, which could be impractical in some applications [3].

3.3 Our Contributions

Our approach is a fully distributed public key-based trust model where certificates management is performed by nodes themselves in a fully distributed way. Namely, our approach relies on trust graphs, where public keys authentication is established via trust chains, unlike models based only on direct trust [13],[12],[26],[15],[18]. However, instead of using private keys for certificates signing, in our approach nodes use private shares to sign certificates. Hence, we introduce a novel trust graph type based on partial certificates chains. Thus, in our model the public key authentication is performed via combination of partial trust chains among nodes, unlike models described in: [1],[3],[24],[16],[22]. We have surveyed, in this paper, existing public key-based trust models proposed for MANETs, and to the best of our knowledge, our proposal is the first in which the trust graph concept is combined with threshold cryptography techniques to elaborate a fully distributed trust model. In what follows, we present the detailed description of our approach.

4 Our Approach

4.1 Motivation and Contribution

Originally in PGP [1], then in several other trust models [2][3][24], an efficient theory of trust relationship is produced which is based on: *if A trusts B and B trusts C, then A can trust C*. According to this trust relationship, the sensitive point of the trust chain is the principal B, in which if B will be compromised, all chains of trust that passe through B will be considered incorrect. Based on this consideration, we propose a more robust trust relationship concept: *if A trusts B and B trusts C, then A can trust C if some other* $(k-1)$ *trusted entities trust C* (cf. figure 2). With this manner, the principal A could control the behaviour of the principal B through the collaboration of other selected trusted entities in the system. In order to provide the trust sharing we employe a (k, n) threshold cryptography scheme, where n is the number of entities in the system and $k < n$ is the trust threshold.

The goal of our contribution is to establish a fully distributed and robust public key-based trust model for MANETs. In this work we target to put into practice our trust relationship in order to resist against malicious nodes, which sign false public key certificates for other nodes in the network.

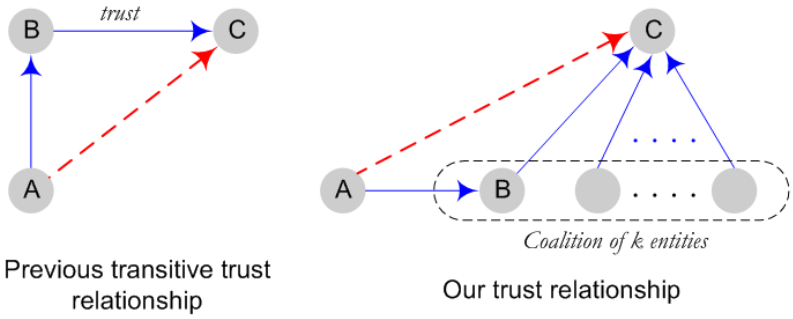


Fig. 2. The trust relationship.

4.2 Notations

Notations used in this paper are summarized in figure 3.

Notation	Description
$\boldsymbol{i}$	The node **i**'s identifier.
$\boldsymbol{K_i^{-1}/K_i}$	The node **i**'s private/public key.
$\boldsymbol{(k,n)}$	Threshold cryptography scheme, where **n** is the number of entities in the system and **k** is the threshold.
$\boldsymbol{K_{system}^{-1}/K_{system}}$	The **system**'s private/public key.
$\boldsymbol{S_i}$	The node **i**'s private share.
$\boldsymbol{S_{i\|j}}$	The node **i**'s private share constructed by **j**.
$\boldsymbol{l_j(i)}$	Lagrange interpolation.
$\boldsymbol{\Delta_j}$	The shuffling factor of the node **j**.
$\boldsymbol{\sigma(x)}$	The sign function. It equals to **1** if $\boldsymbol{x > 0}$, or to **-1** if $\boldsymbol{x < 0}$.
$\boldsymbol{PC_j}$	Partial Certificate signed by the node **j**'s private share.
$\boldsymbol{G(V,E)}$	A graph **G**, where **V** and **E** stand for the set of vertices and the set of edges, respectively.
$\boldsymbol{G(n,p)}$	An undirected graph **G** composited of **n** vertices for which the probability that a link exists between two vertices is **p**.
$\boldsymbol{\rho}$	The average vertices' degree, which represent the average number of edges for a given node; it equals to **p.(n-1)**.
$\boldsymbol{\rho^+}$	The average in-degree, which represents the average number of in-edges for a given node.
$\boldsymbol{\rho^-}$	The average out-degree, which represents the average number of out-edges for a given node.
$\boldsymbol{Pr[G(n,p)\ Connected]}$	The probability to the graph **G** will be strongly connected.

Fig. 3. Notations.

4.3 Overview

Our proposal allows nodes to generate, store, and distribute their public key certificates without any central trusted party. All nodes have a similar role, and we do not assign any special functions to a subset of nodes. Like [1,3], we assume in our approach what follows:

- Users' public/private keys are created by users themselves.
- Certificates checking is achieved via chains of public key.
- Instead of storing certificates in centralized servers, certificates in our system are stored and distributed by nodes themselves.

The main idea of our contribution is the inclusion of a (k, n) threshold cryptography scheme within the graph of trust, in order to resist against false public key certificates issued by malicious nodes in the network. During the network initialization, each node i holds a share S_i of a private key K_{system}^{-1} which is kept secret at a special node called the system dealer. Instead of using private keys to sign certificates, nodes will use their private shares [3]. We call a certificate signed using a private share a partial certificate. Our approach uses a particular graph of trust, which contains patrial certificates' chains, and key authentication among nodes is performed via combination of partial trust chains.

In our model, partial certificates chains in the system are represented by a directed graph $G(V, E)$, where V and E stand for the set of vertices and the set of edges, respectively. We note this graph by *partial trust graph*. The vertices of the graph represent nodes' identifiers and the edges represent partial certificates. As illustrated in figure 4, there is a directed edge from vertex i to vertex j, if and only if there is a partial certificate signed with S_i (i's private share) that binds K_j (j's public key) to the identity of the member j. A partial certificates chain from a node i to another node l is represented by a directed path from vertex i to vertex l in G, as illustrated in figure 4.

Now, we briefly give a general idea of the basic operations of our solution; they will be described in more detail in the following sections. Our approach includes four basic operations:

(1) **Initialization phase:** Each member i in the system is configured with a private share S_i of the system's private key K_{system}^{-1} using a (k, n) threshold cryptography scheme, where n is the number of members in the network.

[3] Unlike all related public key-based transitive trust relationship models, in which certificates are signed usually with using private keys.

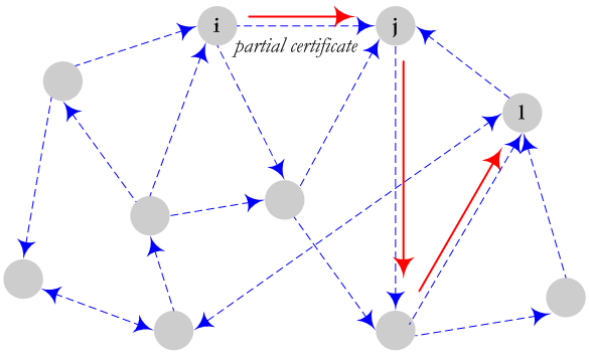


Fig. 4. Partial trust graph.

(2) **Joining the system:** Our approach is developed for open networks, where nodes can join/leave without any restriction. The joining process is performed by a group of at least k members that collaborate to allow access to a new node in the system.

(3) **Partial certificates creation and exchange:** The public and private key of each node is created locally by the corresponding user [4]. If a user i believes that a given public key K_j belongs to a given user j, then user i can issue a partial certificate in which K_j is bound to user j, signed with user i's private share S_i. Moreover, a protocol of local repositories exchange is executed systematically among neighboring nodes in the network. In this phase, the mobility helps nodes to recover a maximum of partial certificates.

(4) **Public-key authentication:** Public key authentication among nodes is performed via partial certificates chains. When a node i needs to authenticate a public key K_j of another node j, both nodes merge their partial trust graphs, and validate it with respect to our trust model based on the aforementioned threshold scheme. Then, node i tries to find a trust chain from node i toward node j in the validated partial trust graph, and if found, the authentication is performed. In the validation process phase we put into action our trust relationship.

We will now provide a detailed description of each operation in the following sections.

4.4 Initialization Phase

In the initialization phase, a system dealer is introduced. The system dealer could be a telecommunication service provider, common to all current members, which has long-term well established trust relationships with current

[4] We assume that each user owns a single mobile node. Hence, we will use the same identifier for the user and her node [3].

nodes [24]. During this phase, each node gets its own private share from the system dealer. Without loss of generality, we use Shamir's secret splitting algorithm (cf. section 2), where the system dealer generates a randomly $k-1$ degree polynomial: $f(x) = \alpha_0 + \alpha x^1 + ... + \alpha_{k-1} x^{k-1} (\bmod \eta)$, where η is a large prime number, and the private key of the system is: $K_{system}^{-1} = \alpha_0 (\bmod \eta)$. This key is kept secret at the system dealer. We assume i to be the node i's unique identifier in the network, and each private share S_i is evaluated as follows: $S_i = f(i)$.

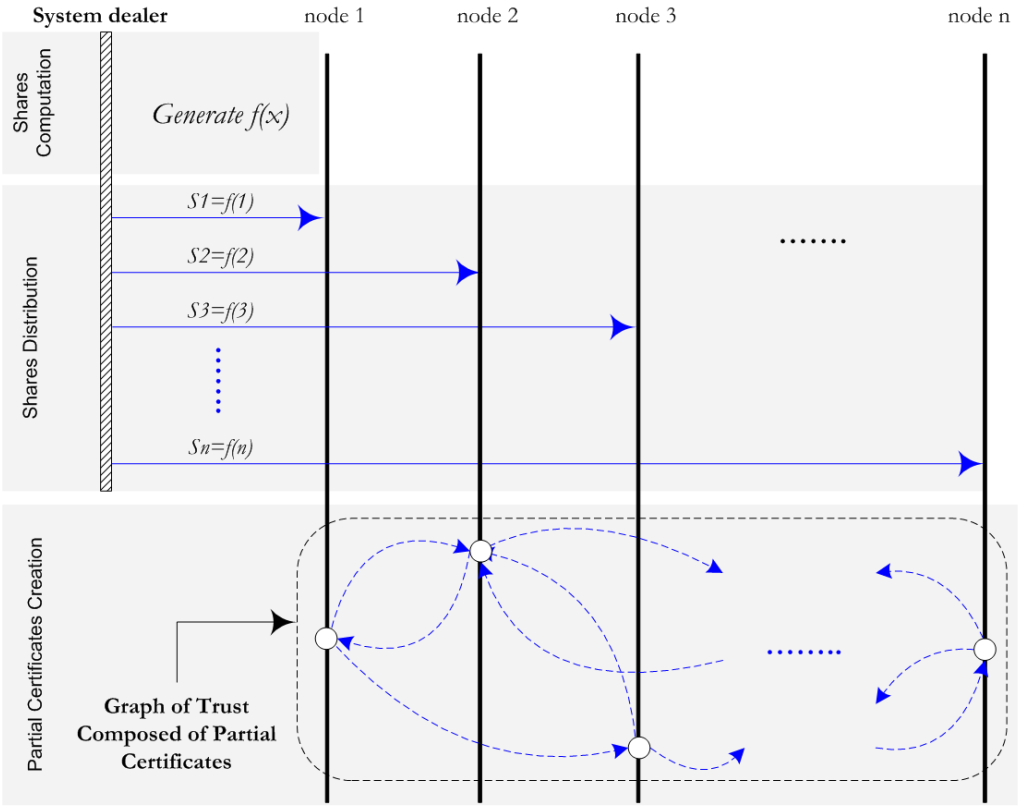


Fig. 5. Initialization phase.

The second part of the initialization phase is the creation of partial certificates. After all nodes get their own private shares, each member generates a partial certificate to each other member it trusts in the system. This allows creating a particular trust graph composed of partial certificates. We assume that there are sparse social trust relationships existing among initial nodes. After that, the system becomes fully functional, and thus no infrastructures can be expected, and the system dealer will not exist any more. The figure 5 summarizes the initialization phase.

Join operation happens when one new node wants to join the system. This operation will be performed in two steps:

- **Step 1:** The mechanism processes as follows. A new node contacts any current neighbor member node to request joining the system. The corresponding member node will handle the joining request during the process, and we call it "a delegated node". First, the new node generates its public/private keys. Then, it sends its public key and a trust evidence to the delegated node to request for a certificate signed by the the system's private key for its public key. The trust evidence is an authentication proof, like a password, for example. The delegated node broadcasts then the request to the other member nodes in order to authenticate the trust evidence provided by the new node. If the trust evidence is authenticated by a current member (let say node j), then the latter sends to the delegated node a partial certificate (PC_j) signed with this member's private share for the new node. This process is repeated until the delegated node gets at least k independent PC, and combine them to generate the complete certificate (which includes the new node public key), and returns it to the new node.

- **Step 2:** To complete the join process, the new member node must get its own private share to participate in the trust establishment. It is, in fact, the process of changing the scheme configuration from (k, n) to $(k, n+1)$. Unlike the first step, the second step must be executed only by the new member node and without using a delegated member. Messages to exchange in this step will include partial private shares of the new member node and they must be kept secret through encryption with the new member's public key. First, the new member node i broadcasts in the network its request signed with its private key. Once member node j receives the request, it verifies the signature in order to authenticate the request, and then computes for it a partial private share: $S_{i|j} = S_j l_j(i) + \Delta_j (\mathrm{mod} \eta)$, where Δ_j is the "shuffling factor" [31] of the node j, and $l_j(i)$ is the Lagrange interpolation:

$$l_j(i) = \prod_{r=1, r\neq j}^{k} \frac{i-r}{j-r} (\mathrm{mod} \eta)$$

  The shuffling factor is introduced to prevent S_j from being disclosed, because the new member node i can recover S_j from $S_{i|j}$ if there is no shuffling factor. One method to generate the shuffling factor requires that each pair of nodes (j, r) in the system exchange a number S_{jr}, and then Δ_j is computed as: $\Delta_j = \sum_{r=1, r\neq j}^{k} \sigma(j-r).S_{jr}$, where $\sigma(x)$ is the sign function:

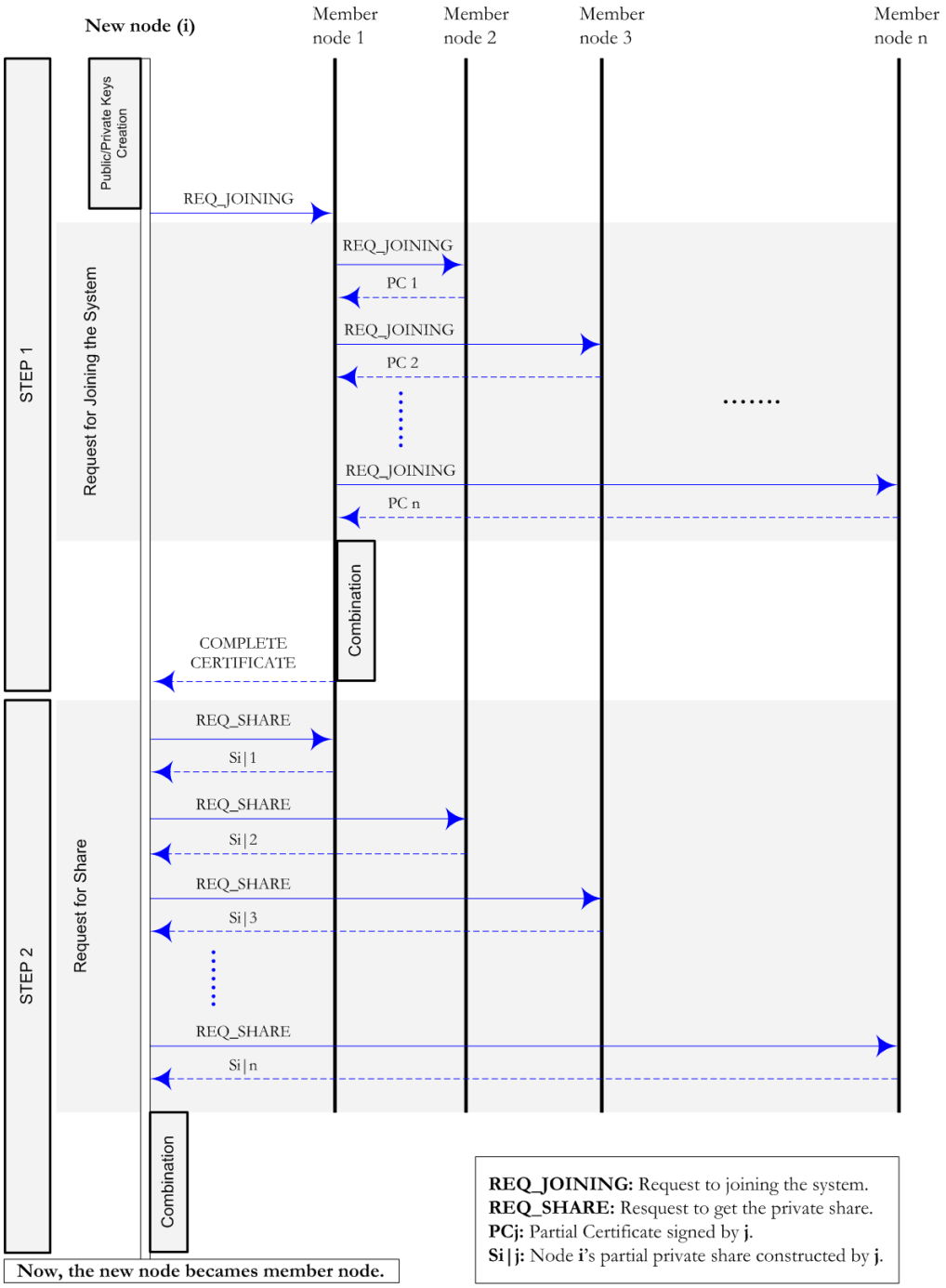


Fig. 6. Joining the system.

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

Each member node j returns a partial private share to the new member node i encrypted by the node i's public key. After receiving k partial private shares, the new node can construct its own private share: $\sum S_{i|j} = S_i(\mathrm{mod}\eta)$. Now, the new node becomes member and it must participate on the certification management.

The figure 6 summarizes all operations in the joining system phase. The next phase is the creation and exchange of partial certificates in the system.

4.6 Partial Certificates Creation and Exchange

We assume in this step that public and private key of each node is created locally by the corresponding user, and public key certificates are issued by users themselves. If user i believes that a given public key K_j belongs to a given user j, then user i creates a public key partial certificate in which K_j is bound to user j, and signs it using its private share. There may be several reasons for user i to believe that K_j belongs to user j. For example, user i and user j may have exchanged their keys through a side channel (e.g., over an infrared channel at the time of a physical contact [3]). To accomplish this point we note that creating partial certificates are the only operation performed consciously by users. All the other operations are performed automatically by the corresponding mobile nodes, without users intervention.

In our approach, partial certificates exchange is an important mechanism that enables nodes to share and distribute partial certificates that they issue and hold. The partial certificates exchange protocol is executed periodically between each node and its neighbors. In this phase, the mobility helps nodes to recover a maximum of experiences from other nodes only by their displacement. Moreover, the dynamic nature of MANETs allows nodes in the network to (1) recover more knowledge from the other nodes, (2) create a great number of partial certificates, and (3) better evaluate the pertinence of partial certificates they created compared to the others.

We illustrate in figure 7 an example of the protocol execution with a system containing five member nodes: A,B,C,D, and E, where each node holds its partial trust graph. As illustrated in the initial network topology, all neighbors' nodes (A−B, A−C, D−E) share similar partial trust graphs as a logic result of the initial exchange execution. When the network topology changes (A−C, A−E, B−D), each pair of neighbors nodes exchange their local repositories, and hence getting more experience about the global view of the graph of trust. Moreover, this mechanism allows also to forward experiences as illustrated with nodes: A,C, and E, where first A exchanges and merges with E, then the novel partial trust graph of A will be exchanged with C.

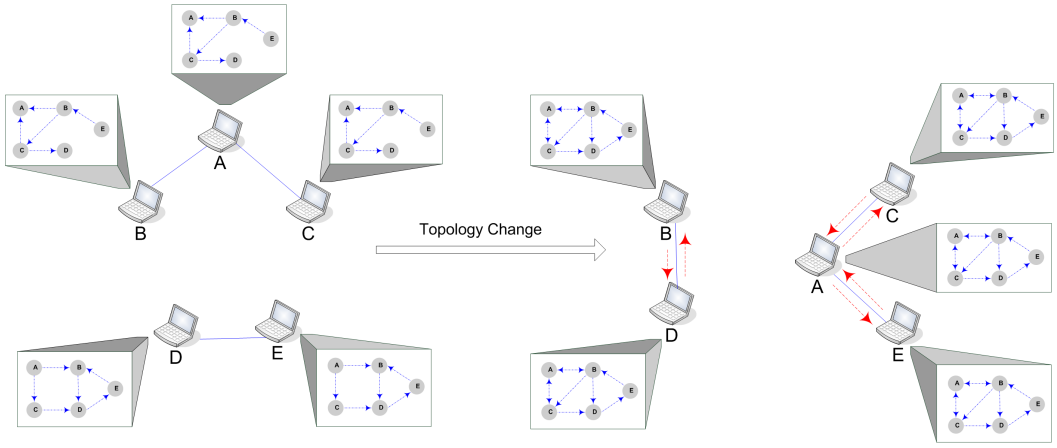


Fig. 7. Partial certificates exchange.

4.7 Public-key Authentication

As described in the previous sections, our approach uses a particular graph of trust, noted *partial trust graph*, composed of partial certificates chains. The partial trust graph is represented by a directed graph $G(V, E)$. The vertices represent nodes' identifiers and the edges represent partial certificates. A directed edge from node i to node j will exist if there is a partial certificate signed with the private share of i that binds the identifier of node j and its public key K_j. A partial certificates chain from node i to node j is represented by a directed path from vertex i to vertex j in G. So if any two nodes in G are connected, then the trust between the two nodes is partially established, and it is represented by a partial certificates chain.

Public key authentication among nodes is performed via the combination of partial certificates chains. When a node i needs to authenticate a public key of another node j, both nodes *merge* and *validate* their partial trust graphs. The validation of the merged graph is performed via the combination of partial certificate signatures. Both nodes i and j try to verify for each member node in the partial trust graph the complete certificate signature. If the verification succeeds, all nodes' incoming edges will be then marked as *trusted edges*. Otherwise, if the combination of signatures fails, all nodes' incoming edges will be deleted [5]. Then, node i tries to find a trust chain (composed of trusted edges) from node i toward node j. If such chain is found, the authentication is performed and then node i trusts node j's public key.

We consider the partial trust graph illustrated in figure 8 (a) as the merged graphs of both nodes E and B, where E requires to authenticate the public

[5] Also, if the number of incoming edges is inferior to the trust threshold k, the combination of signatures will fail, and then all nodes' incoming edges will be deleted.

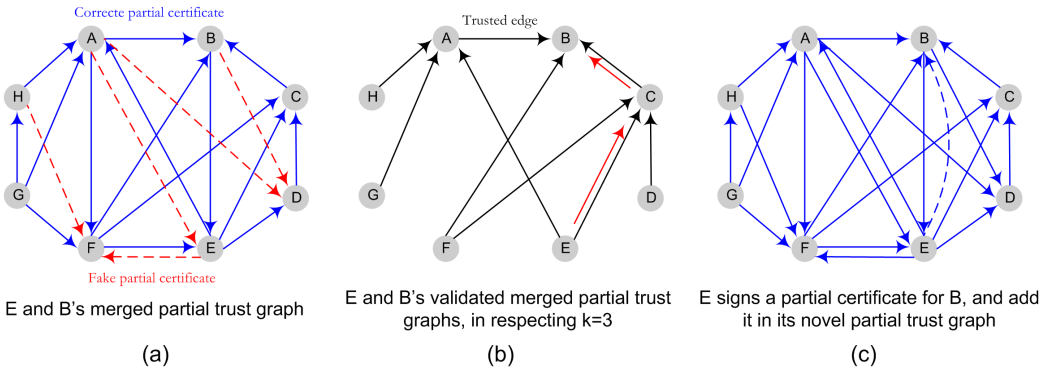


Fig. 8. Partial trust graph validation with respect to a $(3, 8)$ threshold cryptography scheme.

key of B. To perform that, E validates the merged graph and tries to find a trust chain from E toward B. We assume that the merged partial trust graph comprises some fake partial certificates. A has three correct partial certificates, thus the threshold value is respected (trust threshold $k = 3$), so all A's incoming edges are marked trusted. E has three partial certificates, among which one is faked, so the combination fails and all E's incoming edges were then deleted. The figure 8 (b) represents the validated partial trust graph, in which E can employ the trust chain E→C→B and trust the B's public key. Then, E issues a partial certificate for B (cf. figure 8 (c)).

5 Security and Performance Analysis of our Trust Model

5.1 Security of our Approach

Now, we verify the security of our approach by considering malicious nodes in the network. We distinguish between two types of adversaries: *internal* and *external* malicious nodes.

5.1.1 External Malicious Nodes

In our approach, external malicious nodes have no way to impersonate member nodes. Indeed, member nodes can verify their identities by checking the validity of their certificates with the system's public share. Moreover, to be able to sign partial certificates the node must have its private share corresponding to the threshold cryptography scheme, and so, an external node have no way to sign fake partial certificates under the identity of a valid member node.

5.1.2 Internal Malicious Nodes

An internal malicious node may issue several types of false certificates: (1) it may issue a certificate that binds a public key K_i to a node j instead of to node i in order to trick other nodes to believe in this fake binding, (2) it may issue a certificate that binds node j to a fake public key K'_j, which may then cause other nodes to believe in this fake binding, or (3) it can invent a number of node identifiers and public keys and bind them by appropriate certificates.

Our approach resists against false bindings described in (1), (2), and (3). Indeed, if a malicious member intends to invent a fake binding, it must exist at least other $k-1$ partial certificates corresponding to this false binding. However, to do that the adversary member must collect at least k private shares from compromised members and signs k fake partial certificates. Moreover, malicious members, as noted in (3), haven't any way to trick other nodes to believe on a fake partial trust graph, because the merged trust graphs will be validated by deleting all edges that fail the combination of signatures using the system's public key K_{system} of the (k, n) threshold scheme.

5.2 Availability of the Authentication Service

In this subsection we study the optimal value of the system's threshold k^* so that our model achieves the highest availability of the authentication service. Since the existence of trustiness between two nodes is probabilistic, we model the partial trust graph by a random graph. A random graph $G(n, p)$ is an undirected graph composed of n vertices for which the probability that a link exists between two vertices is p, and the vertex's degree ρ is defined as $\rho = p.(n-1)$. The vertex's degree ρ represents the number of edges linking that vertex with its neighbors, where $\rho = \rho^+ + \rho^-$. The in-degree ρ^+ represents the number of in-edges, and the out-degree ρ^- represents the number of out-edges. We set p be the probability that a partial certificate exists between any two nodes, n be the number of nodes in the network, and $\rho = p.(n-1)$ be the node's degree. In [24], K. Ren et al. discussed the case of directed graphs, and they proposed a transforming method, for which starting from a given random graph, a directed random graph can be constructed, called a semi-random graph. Each undirected edge in the random graph will be transformed to two directed edges, in order to keep a mutual directly connection in the resulting semi-random graph (as illustrated in the figure 9). Indeed, $n\rho/2$ edges will be added and the totally edges will be $n\rho$. Thus, the average node's degree ρ' will be double as that of random graph: $\rho' = 2\rho$, with $\rho'^+ = \rho'^- = \rho$.

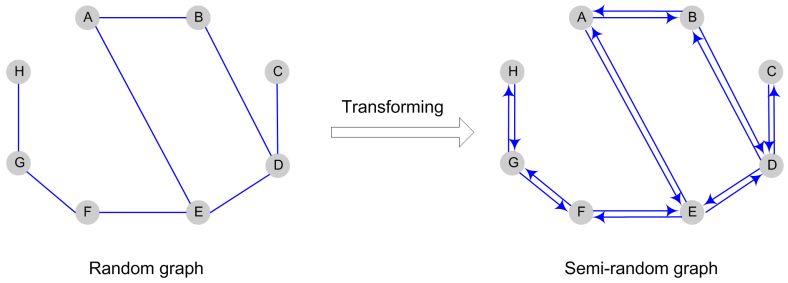


Fig. 9. Transforming a random graph to a semi-random graph.

Lemma 1. *Our trust model achieves the highest availability when the partial trust graph is strongly connected.*

Proof. The availability of our trust model depends on the probability that each two nodes in the network can authenticate each other. Formally, it is interpreted by the availability of trust chains that connect each two nodes in the partial trust graph, and so, the propriety of availability is reduced to a connectivity problem. However, to make possible that each two nodes can authenticate each other, it is required that each two vertices in the partial trust graph be connected at a least by one trust chain, which means that the graph must be strongly connected ($\Pr[G\,connected] \approx 1$). □

Lemma 2. *ρ is a function of the network size and the probability of G connection: $\rho = F(n, \Pr[G\,connected])$.*

Proof. Erdos and Renyi discussed in [7] the propriety $\prec G\,connected \succ$ of random graphs and they showed that there exists a value of p which moves this propriety from "non-existent" to "certainly-true" in a large random graph. They showed that for any real constant β:

$$\text{If } p = \ln(n)/n + \beta/n, \text{ then } \lim_{n\to+\infty} \Pr[G\,connected] = e^{-e^{-\beta}}$$

For a large number of nodes n, we can calculate β as:

$$\beta = -\ln(-\ln(\Pr[G\ connected]))$$

$$\Longrightarrow p = \ln(n)/n + \beta/n = \tfrac{1}{n}[\ln(n) - \ln(-\ln(\Pr[G\ connected]))]$$

$$\Longrightarrow \rho = p.(n-1) = \tfrac{1}{n}(n-1)[\ln(n) - \ln(-\ln(\Pr[G\ connected]))]$$

$$\Longrightarrow \rho = F(n, \Pr[G\,connected]). \square$$

Proposition. *The optimal threshold k^*, so that our trust model achieves the highest availability, can be estimated according to the network size.*

Proof. In our model, ρ'^{+} represents the number of incoming edges. Thus, it represents the needed number of partial certificates signed to nodes in the network. We can thus decide the value of the trust threshold k which equals to $\rho'^{+} = \rho$. According to lemma 1, our model achieves the highest level of availability when $\Pr[G\,connected] \approx 1$. Moreover, according to lemma 2, we can calculate the optimal value of the parameter ρ through the given value of $\Pr[G\,connected]$. Thus, $k^* = \rho = \frac{1}{n}(n-1)(\ln(n) + 4,6)$.[6] $\square$

We illustrate in the figure 10 the plot function of the node's degree ρ for various values of n and $\Pr[G\ connected]$. For example, to make sure that the graph will be connected with a probability of $0,99$ the average vertex's degree needed is only 9 with n equals to 100 vertices. So, to keep a high level of availability (with probability $0,99$) the optimal value of k is 9 in a network size of 100 nodes.

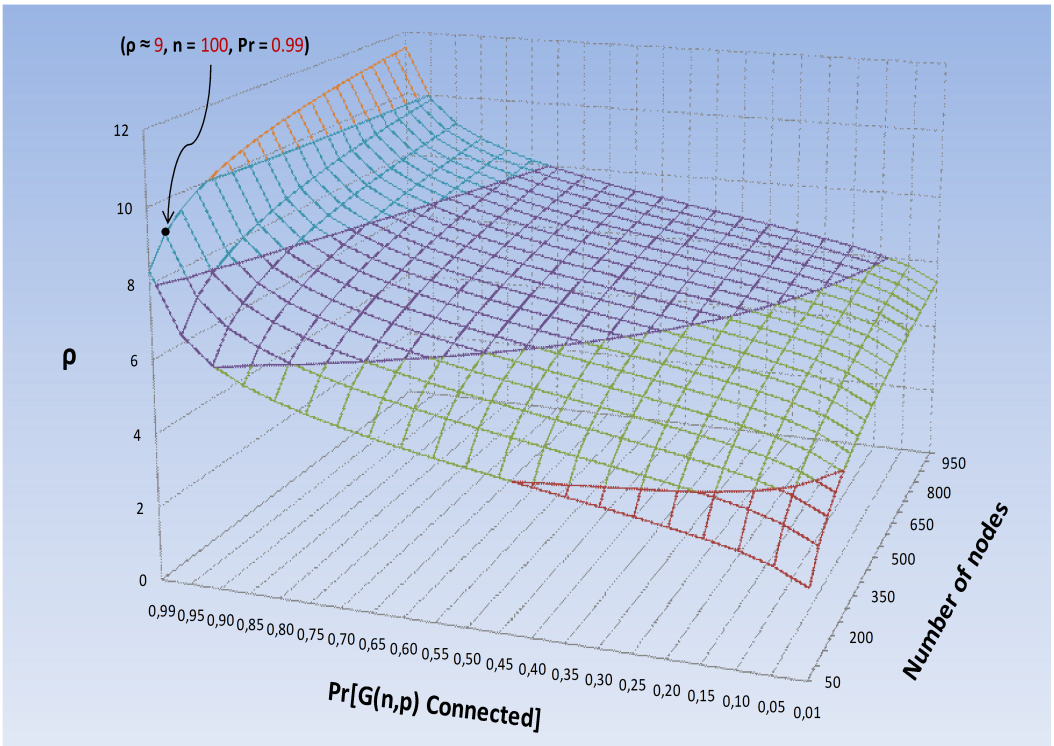


Fig. 10. The average node's degree ρ under various (n, $\Pr[G(n,p)\ Connected]$).

[6] $-\ln(-\ln(0,99)) \approx 4,6$.

6 Simulation Results

6.1 Parameters and Assumptions

In our simulations, we have used MATLAB environment [7]. We simulated a mobile ad hoc network with 100 nodes. Each node has a nominal range of $\alpha = 150$ m and move on a square area of 1km^2. The movement pattern is defined by the *random waypoint model* [10]. In the random waypoint mobility model, a mobile node moves on the area from its current position to a new location by randomly choosing its destination coordinates, its speed of movement, and the time that it will pause when it reaches the destination. After the pause time, the node chooses a new destination, speed, and pause time. This is repeated for each node, until the end of the simulation time. We pair the mobility parameters, such that each node has a maximum speed of 2 m/s, 5 m/s, 10 m/s, and 20 m/s, and a corresponding average pause time of 2 s, 5 s, 10 s, and 20 s, respectively. Our simulator estimates if a radio link exists among nodes according to the distance that separates them. The initial nodes positions are random on the surface. We assume that nodes have the same hardware characteristics and processing capabilities. We assume that certification queries arrive following a Poisson law with an average inter-arrival between queries of $\mu = 1/\lambda = 10$ s. The evaluation is focused on the measurement of the successful of certification requests and the related rate is computed as follows:

$$\frac{number\ of\ successful\ certification\ request}{number\ of\ total\ certification\ request}$$

6.2 Comparison Results

In this subsection, we provide the simulation results comparing our approach, for $k = 3$, with tow distributed systems. The first system is based on PGP with local certificate repositories at individual nodes proposed by S. Capkun et al. in [3]. In this system, a user u verifies the public key of user v by finding a certificate chain from u to v in their local repositories. The second system is the classical PGP, where several certificate repositories servers are involved in order to perform nodes' public key authentication. In this simulation, we study the impact of the presence of malicious nodes and network partitioning.

[7] MATLAB is an abbreviation of MATrix LABoratory. MATLAB is an interactive, matrix-based system for scientific and engineering calculations, designated to solve complex numerical problems.

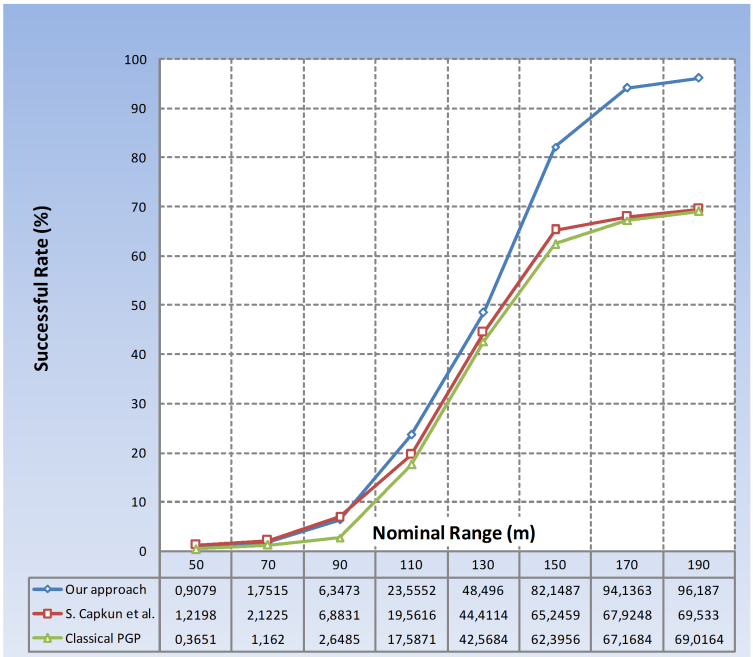
Firstly, we were interested in comparing the successful rate of certification, by measuring the impact of the malicious nodes on the performance of the three systems. We varied the percentage of malicious nodes in the network from 10% to 80%, and we observed the successful rate of requests. The figure 11 compares successful rates among the three systems with the presence of malicious nodes in the network. The two PGP-based systems achieve a low performance compared to our system. In PGP-based systems, a node i requests for public key certificates of another node j by selecting a given certificate path from i to j. However, a such path can contain eventually a false public key certificate signed for node j, so their successful certification rates are low. In our system, on the other hand, for each node we combine all partial certificates, which allow eliminating fake public key certificates, and so it maintains high successful rate.

	10	20	30	40	50	60	70	80
Our approach	86,0359	84,4739	82,1487	75,3952	66,6549	52,4269	35,9162	25,2325
S. Capkun et al.	83,4125	74,2455	65,2459	54,649	46,8246	37,8594	29,2346	20,1368
Classical PGP	79,5133	71,2185	62,3956	53,272	45,5362	36,2857	27,948	18,8724

Fig. 11. Impact of malicious nodes on the successful rate.

Then, we were interested in comparing the impact of the nodes disconnections on successful rate of certificate issuing. We fixed the percentage of malicious nodes at 30% and we varied the nominal range α from 50 m to 190 m. The figure 12 represents the impact of nominal range α on the average rate of successful requests. For reduced α, the performances of the three systems are bad. Starting from nominal range $\alpha = 100$ m our system demonstrates appreciably better performance opposite to the two PGP-based systems. Starting from nominal range $\alpha = 150$ m the performance of our system becomes very interesting. So, practically our system is more powerful in the application areas where nodes are configured with wireless interfaces beyond 150 m. This is not a strong assumption, since most of the actual technologies provide communication ranges greater than 150 m. A great value of nominal range generates a

large connectivity among nodes in the network, and it allows malicious nodes well in diffusing fake public key certificates. Consequently, the success rate is low for the two PGP-based systems. However, our system exploits well the replica mechanism (performed via neighbors exchange protocol) to get a maximum of partial certificates and to eliminate a maximum of fake certificates, so the success rate is kept high.



	50	70	90	110	130	150	170	190
Our approach	0,9079	1,7515	6,3473	23,5552	48,496	82,1487	94,1363	96,187
S. Capkun et al.	1,2198	2,1225	6,8831	19,5616	44,4114	65,2459	67,9248	69,533
Classical PGP	0,3651	1,162	2,6485	17,5871	42,5684	62,3956	67,1684	69,0164

Fig. 12. Impact of network partitioning on the successful rate.

6.3 Practical Analysis of our System

In this subsection, we analysis the practical adaptation of our system with a high level of availability of the authentication service (described in section 5.2). Therefore, we discuss the hardware performance required to put our system in the real practice with a high level of availability. According to this, each node must maintain a partial trust graph which contains at least $2n\rho$ *pc* (Partial Certificates). Thus, in order to manage this quantity of partial certificates, three criteria will be necessarily involved: the storage, the transmission, and the calculus capacities.

(1) **Storage Requirement:** In our system, nodes public key certificates are stored by nodes themselves, and each node maintains a partial trust graph. Indeed, the number of edges in the graph represents the real number of partial certificates stored at each node, which equals to $2n\rho$ *pc*. In practice, the average size approximation of a public key certificate is $\psi = 5$ KB. Thus, the minimal capacity of storage required is $2n\rho\psi =$

$(n-1)(10.\ln(n)+46)$ KB. The figure 13 illustrates the storage capacity required according to various network sizes. For example, with 1000 nodes, we can achieve a high level of availability only by 115 MB of storage capacity at each node (for example, only by a card memory of 128 MB size). Therefore, we note that our system presents no constraints in term of storage capacity requirements.

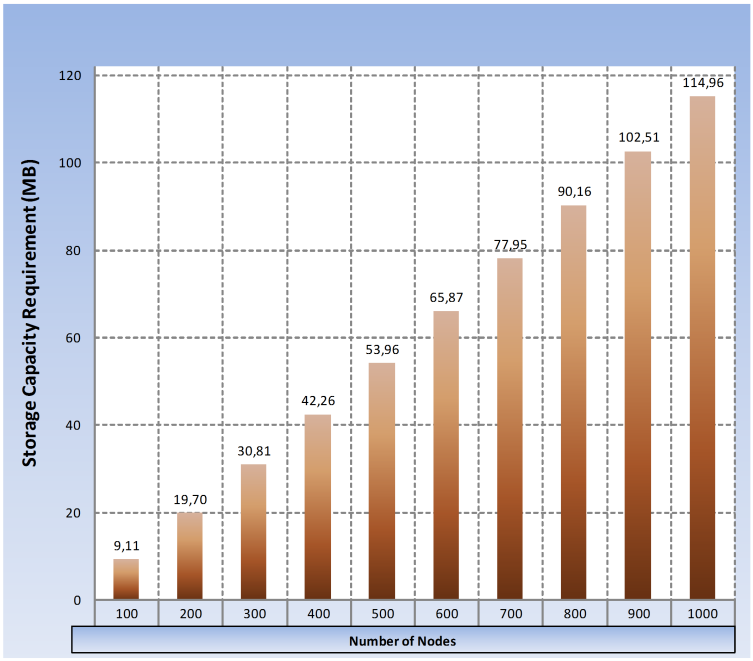


Fig. 13. Storage requirement.

(2) **Transmission Requirement:** In our system, the partial certificates exchange is the expensive step in term of transmission. Indeed, that enables nodes to distribute partial certificates that they issue and hold. The exchange protocol is executed periodically between each node and its neighbors. Each node i tries to find in its partial trust graph the partial certificates missing to its neighbor j and sends them to it. The process consists of improving the partial trust graph G_j (that hold the node j) with new partial certificates that will be extracted from G_i (that hold the node i). We denote the number of partial certificates which will be sent to the node j with $Card\ \{pc \in G_i / pc \notin G_j\}$, and so the rate number of partial certificates is defined by the parameter τ, where:

$$\tau = \frac{Card\ \{pc \in G_i / pc \notin G_j\}}{Card\ \{pc \in G_i\}}$$

In the worst case, when $\tau = 1$ the node i must send $2n\rho\ pc$ to its neighbor. In the figure 14, we illustrate the simulation result of our approach by observing the parameter τ. According to the result, initially τ begins at 20%, and is reduced in the rest of time through the exchange protocol execution among nodes. The parameter τ converges to zero when

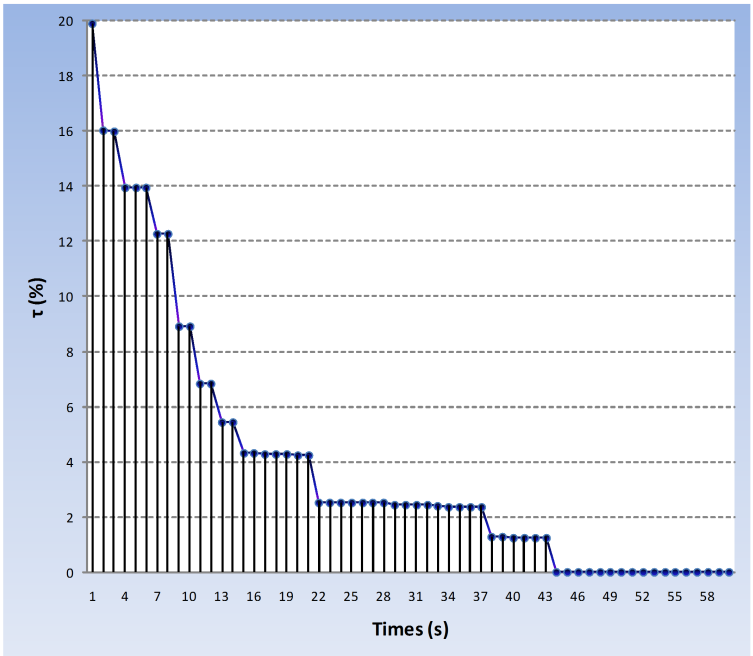


Fig. 14. The average of the parameter τ.

$t = \infty$, i.e. that in the future times all nodes will share, approximatively, the same partial trust graph, and so, the number of partial certificates to be sent converge, approximatively, to 1 pc. Therefore, now we discuss only the worst case when the system is in its initial times, and we put $\tau = 20\%$. We assume that in the initial times the system have a tolerable marge of transmission execution time when the node can achieve the transmission of partial certificates in an average time of $\theta = 10\ s$ with its neighbor. Thus, the node will send $\frac{2n\rho\tau}{\theta}\ pc/s$, so then $8\frac{2n\rho\tau\psi}{\theta}\ Kbit/s$. The figure 15 illustrates the transmission speed needed according to various network sizes. For example, with 1000 nodes, we can achieve a high level of availability only by nodes configured by wireless communication interfaces with 19 $Mbit/s$ of transmission rate. This criteria is well adapted according to the actual wireless communication interfaces equipments.

(3) **Calculus Requirement:** In our system, the phase which contains an important quantity of calculuses is the public key authentication step. We assume that the system have an acceptable level of treatments rapidity only when each node can achieve the authentication process in $\theta = 100\ ms$. Indeed, in order to achieve the authentication process, the node must validate the partial trust graph by combining all partial certificates. So, it must combine n ensembles of partial certificates in the period of θ. The complexity of computational overhead of partial certificates combination according to a (k, n) threshold scheme is: $5n+k^2+4k+5$ [9]. The node must calculate n complete certificates, so it must achieve in

totaly $5n^2 + n\rho^2 + 4n\rho + 5n$ operations. The figure 16 illustrates the CPU speed needed according to various network sizes. For example, with 1000 nodes, we can achieve a high level of availability only by nodes configured with 52 Mhz of speed capacity. Once again, we note that our system presents no constraints according to the actual development of processor technology.

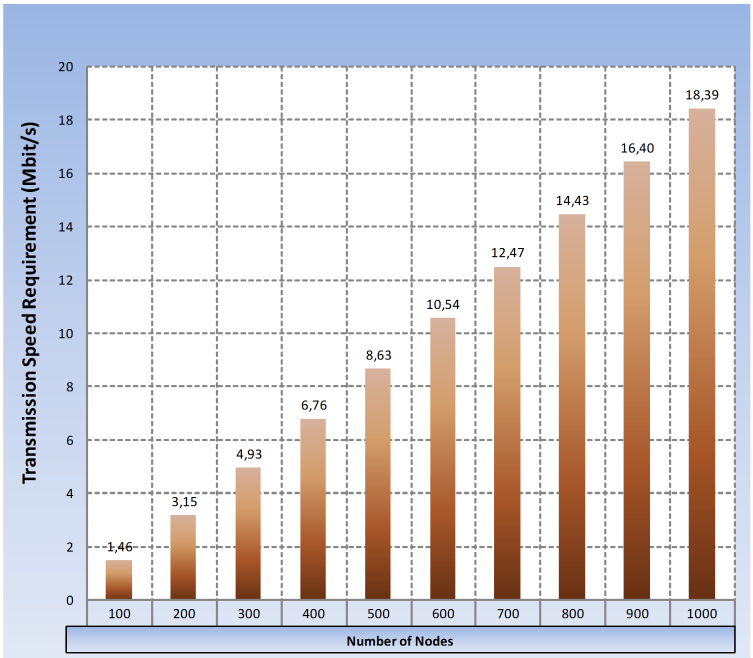


Fig. 15. Transmission requirement.

7 Conclusion

In this paper, we have focused on the trust models and infrastructures in mobile ad hoc networks. We have presented the related models, where we have classified them to two principal categories: partially and completely distributed models. We have then proposed a fully distributed public key management system that does not rely on any trusted authority. Our system is based on a trust graph, in which we have used a threshold cryptography scheme in order to resist against malicious nodes, that issue false public key certificates to cheat the service of certification. To the best of our knowledge, our proposal is the first in which the trust graph concept is combined with threshold cryptography techniques in a fully distributed way. In our system, the detection of false certificates is enabled through the threshold scheme that allows nodes to combine partial certificates and eliminate any conflicting certificates.

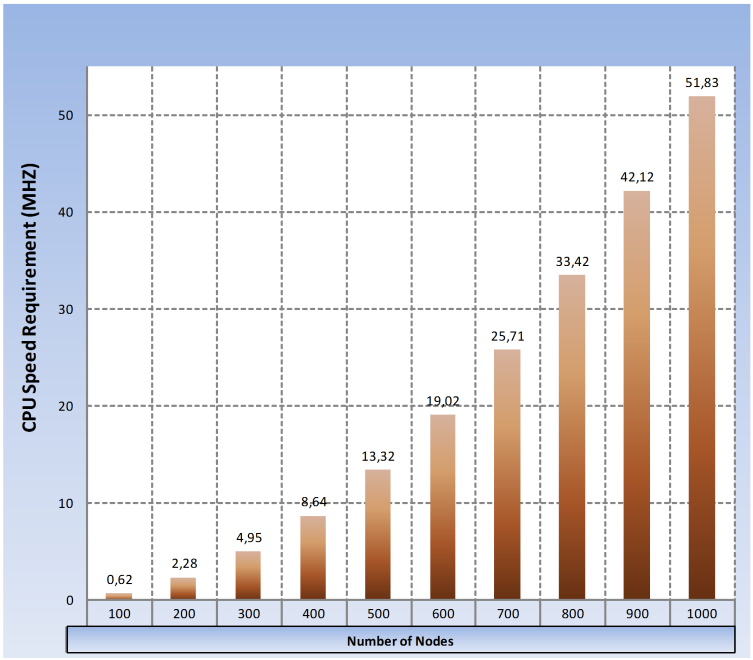


Fig. 16. Calculus requirement.

Our solution is decentralized and completely distributed. It is designed primarily for use in mobile ad hoc networks. An important feature of our system is that public key authentication is still possible even when the network is partitioned and nodes can communicate with only a subset of other nodes. In addition, we conducted the evaluation of three different approaches in public key authentication to observe their performance and characteristics in providing network security against malicious nodes. We have compared two PGP based models with our model, in which our solution demonstrated better resistance against malicious nodes in the network.

References

[1] A. Abdulrahman. The pgp trust model. *The Journal of Electronic Commerce*, 1997.

[2] S. Capkun, L. Buttyan, and J. Hubaux. Small worlds in security systems: an analysis of the pgp certificate graph. *New Security Paradigms Workshop 2002*, 2002.

[3] S. Capkun, L. Buttyan, and J. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, March 2003.

[4] S. Corson and J. Macker. Mobile ad hoc networking: Routing protocol performance issues and evaluation considerations. *IETF RFC-2501*, 1999.

[5] H. Deng, W. Li, and D. P. Agrawal. Routing security in wireless ad hoc networks. *IEEE Communications Magazine*, pages 70–75, October 2002.

[6] Y. Dong, A.-F. Sui, S. Yiu, V. O. Li, and L. C. Hui. Providing distributed certificate authority service in cluster-based mobile ad hoc networks. *Elsevier, Computer Communications*, May 2007.

[7] P. Erdos and A. Rényi. On the evolution of random graphs. *Acta Mathematica Hungarica*, pages 17–61, 1960.

[8] S. Giordano. *Mobile Ad-Hoc Networks, Handbook of Wireless Networks and Mobile Computing*. John Wiley and Sons, stojmenovic edition, 2001.

[9] M.-S. Hwang, E. J.-L. Lu, and I.-C. Lin. A practical (t, n) threshold proxy signature scheme based on the rsa cryptosystem. *IEEE Transactions on Knowledge and Data Engineering*, 15:1552–1560, November/December 2003.

[10] D. Johnson and D. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks, Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[11] J. Kohl and B. Neuman. The kerberos network authentication service version 5. *RFC-1510*, 1991.

[12] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad hoc networks. *In Proceedings of the 9th International conference on Network Protocols*, November 2001.

[13] H. Luo and S. Lu. Ubiquitous and robust authentication services for ad hoc wireless networks. Technical report, UCLA Computer, 2000.

[14] J. Luo, J. Hubaux, and P. Eugster. Dictate: Distributed certification authority with probabilistic freshness for ad hoc networks. *IEEE Transactions on Dependable and Secure Computing*, 2005.

[15] G. Montenegro and C. Castelluccia. Statistically unique and cryptographically verifiable identifiers and addresses. *In Proceedings Ninth Ann. Network and Distributed System Security Symp.*, 2002.

[16] E. Ngai and M. Lyu. Trust- and clustering-based authentication services in mobile ad hoc networks. *In the 24th International Conference on Distributed Computing Systems Workshops*, 2004.

[17] M. Omar, Y. Challal, and A. Bouabdallah. Nettrust: mixed networks trust infrastructure based on threshold cryptography. *In proceedings of IEEE Securecom'07/SECOVAL*, September 2007.

[18] G. O'Shea and M. Roe. Child-proof authentication for mipv6. *ACM Computer Comm. Rev.*, April 2001.

[19] C. Perkins. *Ad Hoc Networking*. Addison Wesley Professional, december 2000.

[20] R. Perlman. An overview of pki trusts models. *IEEE Network*, 1999.

[21] A. Pirzada and C. McDonald. Kerberos assisted authentication in mobile ad-hoc networks. *In Proceedings of 27th Australasian Computer Science Conference*, 2004.

[22] A. Rachedi and A. Benslimane. Trust and mobility-based clustering algorithm for secure mobile ad hoc networks. *International Conference on Systems and Networks Communication*, 2006.

[23] S. Raghani, D. Toshniwal, and R. Joshi. Dynamic support for distributed certification authority in mobile ad hoc networks. *In IEEE International Conference on Hybrid Information Technology*, 2006.

[24] K. Ren, T. Li, Z. Wan, F. Bao, R. H. Deng, and K. Kim. Highly reliable trust establishment scheme in ad hoc networks. *Elsevier, Computer Networks*, April 2004.

[25] A. Shamir. How to share a secret. *Communication of the ACM*, 1979.

[26] W. Wang, Y. Zhu, and B. Li. Self-managed heterogeneous certification in mobile ad hoc networks. *In the Proceedings of IEEE Vehicular Technology Conference*, 2003.

[27] S. Yi and R. Kravets. Moca: Mobile certificate authority for wireless ad hoc networks. *In Proceedings of 2nd Annual PKI Research Workshop*, 2003.

[28] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. *Proceedings of MobiCom 2000, Sixth Annual International Conference on Mobile Computing and Networking*, pages 6–11, August 2000.

[29] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Securing mobile ad hoc networks with certificateless public keys. *IEEE Transactions on Dependable and Secure Computing*, 2006.

[30] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network*, 6(13):24–30, November-December 1999.

[31] B. Zhu, F. Bao, R. Deng, M. Kankanhalli, and G. Wang. Efficient and robust key management for large mobile ad hoc networks. *Elsevier, Computer Networks*, 48:657–682, December 2005.