Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 1498–1512
computer communications
www.elsevier.com/locate/comcom

Building a virtual hierarchy to simplify certification path discovery in mobile ad-hoc networks

Cristina Satizábal [a,b], Juan Hernández-Serrano [a], Jordi Forné [a], Josep Pegueroles [a,*]

[a] Department of Telematics Engineering, Technical University of Catalonia, Jordi Girona 1-3 C3, 08034 Barcelona, Spain
[b] Department of Engineering and Architecture, Pamplona University, Km 1 via Bucaramanga, Pamplona, Colombia

Received 25 July 2006; received in revised form 17 December 2006; accepted 29 December 2006
Available online 9 January 2007

Abstract

The ease with which nodes may join or leave a Mobile Ad-hoc Network (MANET) implies changing trust relationships among them and problems to build certification paths. Peer-to-peer Public Key Infrastructures (PKIs) are quite dynamic and certification paths can be built although part of the infrastructure is temporarily unreachable. However, path discovery is difficult because trust relationships are bidirectional. On the contrary, in hierarchical PKIs, there is only one path between two entities and certification paths are easy to find. We propose a protocol that establishes a virtual hierarchy in a peer-to-peer PKI. This protocol is suitable for dynamic environments such as MANETs since it is executed in a short time. In addition, our protocol does not require to issue new certificates among PKI entities, facilitates the certification path discovery process and the maximum path length can be adapted to the characteristics of users with limited processing and storage capacity.

Keywords: PKI; MANET; Hierarchical trust model; Peer-to-peer trust model

1. Introduction

Advances in wireless technology and portable computing along with demands for greater user mobility have provided a major impetus toward ubiquitous computing. Wireless networks provide mobile users with ubiquitous communicating capability and access to the information regardless of their location.

If we restrict ourselves to ground radio networks, we can define two basic types of wireless networks: cellular and multi-hop. In cellular radio networks mobile users communicate via a single hop wireless channel with a base station which is in turn connected to a wired backbone. In a multi-hop wireless network, in contrast, there are not fixed base stations connected to a wired network. Thus, all nodes communicate via the wireless channel with possible multi-hopping over several mobile stations. Examples of such networks are ad-hoc networks and packet radio networks.

Mobile ad-hoc networks (MANETs) are dynamic peer-to-peer networks that do not have a pre-existing infrastructure. This lack of fixed backbone means the absence of central administration, stable connection, control over the network principals and such. Consequently, mobile end-systems in an ad-hoc network are expected to act cooperatively to support all network functionalities. As a first approach, such functionalities are traffic routing and adaptation to the highly dynamic state of network links. These functionalities can also be extended to any service over the network, such as security. Since mobile devices are capable of forming an ad-hoc network on the fly, the existing security infrastructure of wired networks fails and demands dynamic security considerations, requiring as few assumptions as possible about the nature of the network.

Public Key Infrastructure (PKI) [1] can be used to provide the secure interchange of the information since it

* Corresponding author. Tel.: +34 934016012; fax: +34 934054264.
E-mail addresses: isabelcs@entel.upc.edu (C. Satizábal), jserrano@entel.upc.edu (J. Hernández-Serrano), jforne@entel.upc.edu (J. Forné), josep.pegueroles@entel.upc.edu (J. Pegueroles).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.12.024

supports data integrity, confidentiality, strong authentication and non-repudiation. PKI uses Trusted Third Parties (TTPs), known as Certification Authorities (CAs) to digitally sign Public Key Certificates (PKCs), ensuring that a particular public key belongs to a certain user. In addition, the certificates allow establishing different trust relationships among the entities of the PKI. Thus, an entity can build certificate trust chains, known as certification paths, from its trusted CA to the other entities and verify the validity of the certificates.

However, PKI is mostly designed for centralized, wired and well-connected networks, so adopting PKI in MANETs is not an easy task since the network topology and resources can change frequently. Also, there is a high possibility that a node is compromised therefore some precautionary steps are needed to preserve the security of public key certificates. In addition, when CAs are not reachable, several PKI services, such as certification path processing and certificate status checking are not available.

Previous drawbacks are mainly overcome in MANETs by decentralizing CA functionalities and self-organized PKIs. In the latter, management of multiple trust relationships among PKI entities has to be taken into account, since nodes can dynamically create bidirectional trust relationships among them. In this scenario, certification path discovery becomes a difficult task because they can have multiple paths between two entities and all the options do not lead to the target entity.

Validation of long paths can also be hard for mobile devices with limited capacities from the computational point of view, since the public key algorithms require complex mathematics calculations, and considering the set of resources necessary to obtain, store and verify the certificates.

The purpose of our proposal is to take advantage of the efficiency in the path discovery process offered by hierarchical PKIs, where trust relationships are unidirectional and paths are easy to find. For that reason, our protocol establishes a virtual hierarchy among the entities of a peer-to-peer PKI and contributes to simplify the path discovery process. In a hierarchical model, there is only one path between two entities so the verifier must carry out less search operations to find a path than in a peer-to-peer architecture. Additionally, services based on hierarchical PKIs can be easily used. Also, since validation of long paths is difficult for verifiers with limited capacities, the hierarchy is built considering a maximum path length, whose value can be established taking into account the features of the users' terminals.

The different approaches that use PKI in MANETs are put forward in Section 2. Section 3 compares hierarchical and peer-to-peer trust models, and describes the certification path validation process. In Section 4, we present an example that shows the advantages of establishing a virtual hierarchy in a peer-to-peer PKI. Then, we describe the operation of our protocol in Section 5. Section 6 explains the reason for constructing the hierarchy from the leaves to the root instead of from the root to the leaves. In Section 7 we talk about the tools used to implement our protocol and the characteristics of the simulated environment. Section 8 contains the outcomes of the simulation. Finally, Section 9 concludes.

2. MANETS: Approaches to PKI

Node connectivity is not guaranteed in MANETs due to the limited-range and unreliability of wireless links. The absence of a routing infrastructure that assures connectivity of nodes precludes supporting a stable and long-term trust infrastructure, such as a hierarchy among subsets of network nodes. In addition, a single mobile node functioning as a CA will come to a halt the entire MANET if it moves out of the network and also can act as a single point of failure if it becomes compromised. Replicated CAs may be used to avoid this security bottleneck, but this solution is not scalable from administration perspectives and creates several points of failure if any CA node is compromised.

There are essentially two families of approaches for eliminating a centralized certification authority in a mobile ad-hoc network. The first consists in distributing the functionality of the certification authority among several nodes by threshold cryptography; and the second is based on a self-organized public-key infrastructure.

2.1. Distributed PKI with threshold cryptography

The ease with which nodes may join or leave a MANET shows that it is advisable to distribute the CA functionality among several nodes in the network rather than assign it to a single node. Using threshold cryptography, multiple trusted nodes are required to sign a certificate and an adversary would need to corrupt a large number of them before he/she can forge a certificate.

The fundamental problem of threshold cryptography is the secure sharing of a secret. A secret sharing scheme allows one to distribute a piece of secret information among several servers in a way that meets the following requirements: (1) no group of corrupt servers (smaller than a given threshold) can figure out what the secret is, even if they cooperate; (2) when it becomes necessary that the secret information be reconstructed, a large enough number of servers (a number larger than the above threshold) can always do it. Many existing threshold cryptosystems are based on Shamir's (k,n) secret sharing [2]. In these approaches, threshold cryptography is used for distributing the CA private key among n nodes, but the CA functionalities can be assumed by only k nodes ($k < n$). Thus, each time, k nodes collaborate to generate a signature. As these functionalities rely on the CA signature,[1] the goal is to distribute the signing power.

[1] Notice that the main functionality of a CA is to sign certificates.

The contributions for distributed CAs in MANETs can be classified in two types: "Partially-Distributed CA" [3–5] and "Fully Distributed CA" [6,7]. In the first case, the signing power is distributed to a set of servers; and in the second, the signing power is distributed to the whole group of members.

2.1.1. Partially-Distributed CA

Yi and Kravets [3] propose the use of mobile CAs (MOCAs). A MOCA is a mobile node within the ad-hoc network selected to provide distributed CA functionality. Typically, the more powerful and more secure nodes are chosen to provide CA service to other nodes. MOCA nodes apply threshold cryptography to share the responsibility and increase availability of the ad-hoc network. When a client wants to obtain a certificate, it sends Certificate Request (CREQ) packets to at least k MOCA servers and waits for a reply. Any MOCA that receives a CREQ will send a Certification Reply (CREP) packet containing its partial signature. Once the client receives k valid CREPs, it can reconstruct the whole certificate.

Budakoglu and Gulliver [4] use threshold cryptography and threshold function sharing to distribute the certification authority functionality among a specific number of mobile nodes. They set several threshold levels to offer users flexibility in choosing an appropriate security level for a given application. Thus, they provide distributed, fault tolerant and hierarchical key management services.

Dong et al. [5] also propose an architecture for providing a partially distributed CA service in MANETs based on threshold cryptography. They make use of the cluster structure to provide CA service and design a scheme for locating CA server nodes in MANETs. In addition, they provide a proactive secret share update protocol, which periodically updates CA secret shares with low system overhead. Compared with other approaches, this CA architecture provides faster CA services to user nodes reducing system overhead.

2.1.2. Fully Distributed CA

The goal of Kong et al. [6] is to provide pervasive CA services by making all n nodes in the network share CAs functionality. However, since sharing the private key by all participants is obviously unfeasible, especially in MANETs, Kong's scheme distributes the private key into coalition of nodes. Each coalition has at least k nodes, and they are one hop apart so that a client needs only contact k instead of n nodes by sending a local broadcast request to obtain the service. Once a certificate is issued, it is trusted by the whole network since it has been signed by the private key. Thus, a well-defined issuing policy is needed to control certificate issuing. Kong assumes that k one-hop neighbors or more usually exist. Nevertheless, in a mobile environment, this will not be the case, so Kong suggests that if the client does not receive sufficient responses within some time limit, it should move to another location and try again.

Dhillon et al. [7] propose an approach to integrate a fully distributed certification authority in OLSR (Optimal Link State Routing). This approach enables an OLSR MANET to autonomously self-secure itself without any external administration. In addition, the proposed approach minimizes the signaling overhead by supporting security at the network layer level.

2.2. Self-organized PKI

In Pretty Good Privacy (PGP), any user can sign another user's key. These signatures form a network of peer trust relationships, often described as a web of trust [8]. However, the distribution of the certificates is based on publicly accessible certificate directories that reside on centrally managed servers. Therefore, this approach is not fully self-organized.

Hubaux et al. [9] proposed a distributed implementation of PGP where certificates are stored and distributed by the users. Thus, each user maintains a local certificate repository that contains a limited number of certificates. When a user wants to obtain the public key of another one, they merge their local certificate repositories to find an appropriate certification path. The authors present two algorithms that users can use to build their local certificate repositories and show that the algorithms achieve a high performance on real PGP trust graphs, which means that any pair of users can find certificate chains to each other using only their local certificate repositories with a high probability. In addition, the size of the local certificate repositories is small compared to the total number of users in the system.

Our protocol does not use threshold cryptography but assumes a self-organized PKI among the nodes of a MANET as proposed by Hubaux et al. [9]. However, our objective is to simplify the certification path discovery establishing a virtual hierarchy among the nodes of the network. Our protocol is fast, so it can adapt to the dynamic MANET environment. In addition, this does not require issuing new certificates among the participants in the virtual hierarchy and the maximum path length can be adapted to the sometimes limited capacities of users' terminals.

3. Toward PKI trust models in MANET

3.1. Certification path validation process

A CA's *certification domain* defines the organizational or geographical boundary within which the CA is considered trustworthy. Thus, all the PKI users in a CA's certification domain consider this authority like their trust anchor.

A *trust anchor* is a Certification Authority (CA) that a PKI user explicitly trusts under all circumstances. It is used by the client application as the starting point for all certificate validation. Each user receives the public key of its trust anchor when it is registered in the PKI.

When two users belong to the same certification domain and they want to communicate each other, one can obtain easily the other's public key, since they know the public key of their trust anchor. But when users belong to different certification domains their communication is only possible if there is an uninterrupted chain of trust points between them, which supposes the intervention of several CAs and an agreement among their policies. CAs use cross-certification to allow users building trust chains from one point to another, called certification paths.

Cross-certification is the establishment of a trust relationship between two certification authorities through a certificate signed by a CA and that contains the public key of another CA, referred to as *cross-certificate* [1].

A *certification path* [10] is a chain of public key certificates through which a user can obtain the public key of another user. The path is traced from the verifier's trust anchor to the CA public key required to validate the target entity's certificate. Thus, the certification path length is equal to the number of CAs in the path plus one: a certificate for each CA and the target entity's certificate.

The primary goal of a path validation process is to verify the binding between a subject and a public key. Then, the verifier must check the signature of each certificate in the path in order to trust the public key of the target entity. In general, a path validation process involves the following steps:

- *Discovering a certification path*: It is to build a trusted path between the verifier's trust anchor and the target entity based on the trust relationship among the CAs of the PKI. When a certification path is built from the target entity to a trust anchor, this is called building in the *forward* direction. When a certification path is built from a trust anchor to the target entity, this is called building in the *reverse* direction [11].
- *Retrieving the certificates*: It is to retrieve each certificate in the path from the place(s) where they are stored. The most common method for the distribution of certificates and certificate revocation information in the enterprise domain is publication. The idea behind publication is that PKI information is posted in a widely known, publicly available, and easily accessible location. Publication is particularly attractive for large communities of users who in general are personally unknown to one another (that is, the PKI information does not have to be distributed directly to each individual). In today's enterprise, it is common practice to post (or publish) certificates and certificate revocation information (particularly revocation information based on CRLs [1]) to a repository. A *repository* is a generic term used to denote any logically centralized database capable of storing information and disseminating that information when requested to do so [12].
- *Verifying the digital signatures*: It is to verify the validity of the digital signature of each certificate in the path. It involves:
  1. Decrypting the signed part of the certificate with its issuer's public key.
  2. Calculating a hash of the certificate's content.
  3. Comparing the results of 1 and 2. If they are the same then the signature is valid.
- *Verifying the validity of the certificates*: It is to determine if the certificates have expired or have been revoked. The certificate validity period is used to verify the expiration, while the revocation status depends on the revocation mechanism used. Certificate revocation is the mechanism under which an issuer can revoke the binding of an identity with a public-key before the expiration of the corresponding certificate. Issuer can use periodic publication mechanisms such as Certificate Revocation Lists (CRLs) [1], or online query mechanisms such as the Online Certificate Status Protocol (OCSP) [13].

3.2. Hierarchical vs. Peer-to-peer Trust models

Certification architectures or trust models provide a technological framework for creating and managing trust relationships among the different entities of a PKI. Thus, certification architectures describe how the trust relationships and the necessary rules to find and to cross the certification paths are built. The most popular PKI trust models are hierarchical and peer-to-peer (see [14,15]).

3.2.1. Hierarchical Model

This is the most common model. In this configuration, all users trust the same root CA (RCA). That is, all the users of a hierarchical PKI begin their certification paths with the RCA's public key. In general, the root CA does not issue certificates to users but only issues certificates to subordinate CAs. Each subordinate CA may issue certificates to users or another level of subordinate CAs, if it is permitted by the certification policies. In a hierarchical PKI, trust relationships are unidirectional (Fig. 1).

Hierarchical PKIs are scalable. Certification paths are easy to develop because they are unidirectional and the longest path is equal to the depth of the tree less one, since

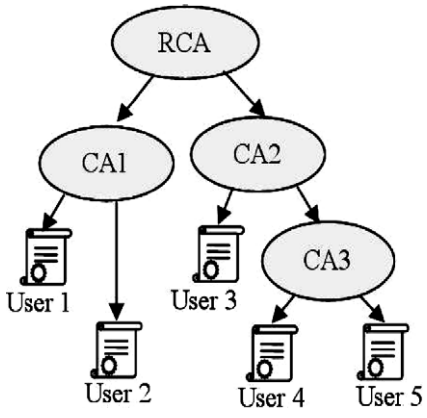

Fig. 1. Hierarchical model.

RCA's certificate is not part of the path because it is known by all entities in the architecture. In addition, users of a hierarchy know implicitly which applications a certificate may be used for, based on the position of the CA within the hierarchy.

The drawbacks of the hierarchical model result from the reliance on a single trust point. The compromise of the RCA's private key results in a compromise of the entire PKI. In addition, transition from a set of isolated CAs to a hierarchical PKI may be logistically impractical because all users must adjust their trust points.

3.2.2. Peer-to-Peer Model

It is also known as mesh or cross-certificate architecture. Here, the user's trust anchor is its local CA and all the CAs can be trust points because they are autonomous. Autonomy refers to the fact that the CA does not rely on a superior CA in a hierarchy. An autonomous CA can perform peer-to-peer cross-certification with other autonomous CAs. Thus, a pair of certificates describes their bidirectional trust relationship (Fig. 2). However, trust relationships may not be unconditional. If a CA wishes to limit its trust, it must specify these limitations in the certificates issued to its peers. All certificate validation, by clients within an autonomous CA, starts with the local CA's self-signed certificate.

Peer-to-peer PKI can easily incorporate a new community of users and although the management cost is high, there is not a single point of failure since it counts on different trust points and they can have multiple paths between two users. In addition, a peer-to-peer PKI can easily be constructed from a set of isolated CAs because users do not need to change their trust points. This model serves to represent the dynamic changes of the organizational structures or environments where communicating entities are not related hierarchically.

The drawback of this model is that the number of trust relationships is directly proportional to the number of CAs (n), that is, the number of trust relationships is equal to $n^*(n-1)$, what causes scalability problems. In addition, this model requires larger certificates because the users must determine which applications a certificate may be used for based on the content of the certificates. Thus, certificates have more extensions and the validation process is more complex. The maximum length of a certification path in a peer-to-peer PKI is the number of CAs in the infrastructure.

Table 1 compares the characteristics of hierarchical and peer-to-peer trust models.

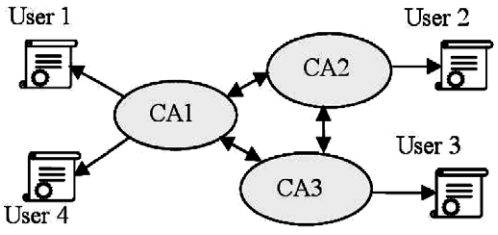

Fig. 2. Peer-to-peer model.

Table 1
Hierarchical model vs. peer-to-peer model

Characteristic	Hierarchical	Peer-to-peer
Trust anchor	Root CA	Local CA
Trust relationships	Unidirectional	Bidirectional
Scalable	Yes	No
Number of paths between two entities	One	Multiple
Path discovery	Easy	Difficult
Longest path	Depth of the tree less one	Number of CAs in the infrastructure
Usage of a certificate	Implicitly, based on the position of the CA within the hierarchy	Based on the content of the certificate
Points of failure	Single trust point	Multiple trust points

3.2.3. PKI trust models in MANET

Both previously explained approaches for PKI trust models can also be extended to MANETs.

Hierarchical PKI models can be handled by only assuring CAs availability. This can be achieved just distributing their functionalities as described in Section 2.

On the other hand, peer-to-peer models can be directly applied to MANET, although, as noted in Section 3.2.2, they can be computationally costly and do not support scalability.

This work takes advantage of the benefits of hierarchical and peer-to-peer models described in Table 1. Our goal is to provide a simple protocol with multiple trust points, easy path discovery and scalable. The proposed protocol builds a virtual hierarchical model on the fly to facilitate path discovery. Moreover, the use of trust relationships from a peer-to-peer model provides multiple trust points and avoids the distribution of CA functionalities between different nodes.

4. How Hierarchical Model simplify path discovery

In this section we show through an example how the certification path discovery is easier in a hierarchical model than in a peer-to-peer model. Fig. 3 shows a peer-to-peer PKI with 10 entities. Arrows represent the certificates issued from one node to another.

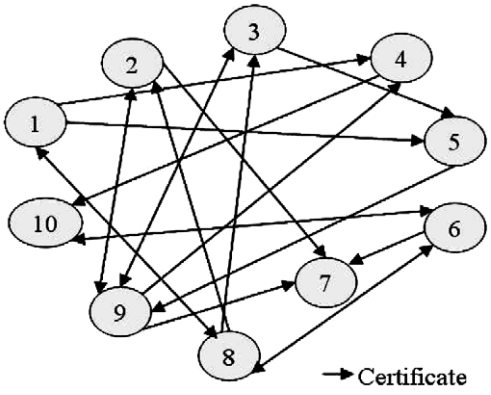

Fig. 3. Peer-to-peer PKI.

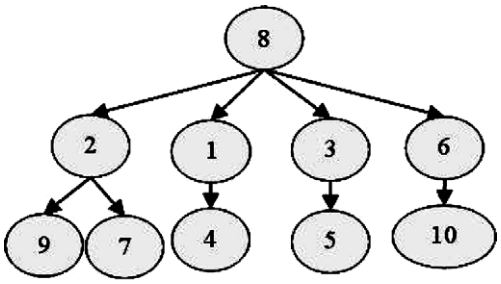

Fig. 4. Hierarchical PKI.

Entity 5 wants to discover a certification path to entity 10. In a peer-to-peer PKI building in the reverse direction (from a trust anchor to the target entity) is more effective because it allows rejecting more quickly the certificates that are not useful in constructing a valid certification path [11], so entity 5 starts the path discovery process from its trust anchors (entities 1 and 3). Entity 5 will verify the signature and validity of the certificates issued by its trust anchors. Then, it will look for the certificates issued by the subjects of such certificates and will continue the search until find a valid path to entity 10.

Entity 3 issues certificates to entities 9 and 5. Thus, the possible paths found by entity 5 beginning from entity 3 will be: 3-5/3-9-2-9/3-9-2-7/3-9-3/3-9-4-10/3-9-7. On the other hand, the possible paths found by entity 5 beginning from entity 1 will be: 1-4-10/1-8-6-7/1-8-6-8/1-8-6-10/1-8-3-5/1-8-3-9-7/1-8-3-9-4-10/1-8-3-9-3/1-8-3-9-2-7/1-8-3-9-2-9/1-8-1/1-8-2-7/1-8-2-9-2/1-8-2-9-3-5/1-8-2-9-3-9/1-8-2-9-4-10/1-8-2-9-7. Therefore, entity 5 has 23 options of which 5 are successful paths.

Considering a set of certificates among the entities of Fig. 3, we can build the hierarchy shown in Fig. 4.

In hierarchical PKIs, building in the forward direction (from target entity to a trust anchor) is more effective because each entity has a certificate issued by its superior CA and there is only one path between two entities. Thus, if entity 5 in Fig. 4 wants to discover the path to entity 10, it must retrieve the certificate of entity 10 issued by entity 6 and the certificate of entity 6 issued by entity 8. Since entity 8 is the trust anchor of all the entities in the hierarchy because it is the root CA, entity 5 can verify the signature of the entity 6's certificate with the public key of entity 8 and then verify the signature of entity 10's certificate with the public key of entity 6. Therefore, in this case, entity 5 can find only one path to entity 10: 8-6-10. Thus, the certification path discovery in a hierarchical PKI is simple compared to the path discovery in a peer-to-peer trust model.

5. PROSEARCH (Protocol to Simplify the Certification Path Discovery Constructing a Hierarchy)

5.1. Protocol description

In this section, we describe PROSEARCH operation. This protocol establishes a virtual hierarchy in a peer-to-peer PKI, based on the trustworthiness level of the participant entities. The hierarchy is built from the leaves to the root (upwards). This protocol facilitates the certification path discovery process and can be adapted to users with limited capacities.

Some aspects of our protocol are inspired on the algorithm proposed by J. Hernandez-Serrano et al. in [16], although the application area is different. Table 2 shows the notation used in this paper.

We divide our protocol in two phases to understand it better:

- *Trustworthiness order among entities*: In this phase, the neighboring entities are arranged from the less trustworthy to the most trustworthy.
- *Construction of the hierarchy*: In this phase, it is established a hierarchical trust relationship among the entities of the peer-to-peer PKI.

5.1.1. Trustworthiness Order among Entities

The protocol begins when an entity E_0 declares to its neighbors (entities that issued a certificate to E_0 and entities that E_0 issued a certificate) that it wants to establish a hierarchical trust relationship with them. In addition, E_0 propose a maximum certification path length (L_{MAX}) based on its processing and storage capacity. Thus, E_0 sends a request message to its neighbors containing the value of L_{MAX}. These messages and all the messages sent among the entities along the protocol must be authenticated by the receiver.

Each neighbor can accept or refuse to collaborate in the establishment of that hierarchy, sending to the demanding entity an acceptance or rejection message.

Once entity E_0 receives the responses from all its neighbors, it determines the number of entities that want to be part of the hierarchy and issued a certificate to E_0 (IN_0), and the number of entities that want to participate in the hierarchy and received a certificate from E_0 (OUT_0). Then, E_0 sends these values to its participant neighbors in an information message and these neighbors send to E_0 their own parameters IN_i and OUT_i.

Later, E_0 compares OUT_0 with the received OUT_i values and puts them in order from the lowest to the highest. The entity with the lowest OUT_i is the less trustworthy,

Table 2
Notation

Notation	Meaning
L_{MAX}	Maximum path length allowed
E_i	Entity i
L_i	Number of certificates from the leaves to entity i
IN_i	Number of entities which E_i trusts (received certificates)
OUT_i	Number of entities that trust E_i (issued certificates)
E_0	Current entity
N_0	Number of participant neighbors of E_0
Order$[N_0+1]$	Array that contains E_0 and its participant neighbors ordered from the less trustworthy to the most trustworthy
pos	Position of E_0 inside the Order array

that is, the neighbor that less the other participants trust. If there are two or more entities with the same OUT_i, they are arranged in accordance with the IN_i value from the lowest to the highest too. For the sake of simplicity, we have not considered other parameters to put in order the entities such as existing policy mapping or distance between entities, but these can be considered if parameters OUT_i and IN_i are the same for two or more entities. We take into account the identifier of each entity in these cases. Thus, each entity put in order its neighbors, from the less trustworthy to the most trustworthy, determining which of its neighbors are less trustworthy and more trustworthy than itself. At the beginning of the protocol, $L_i = 0$ for all the entities.

The following algorithm describes the procedure done by each authority in the first phase of the protocol.

```
BEGIN
    IF E0 begins the protocol THEN
      E0 CHOOSE LMAX
      E0 SEND request message TO all its neighbors
      E0 RECEIVE acceptance/rejection messages FROM
      its neighbors
    ELSE
      E0 RECEIVE request message FROM some neighbor
      IF E0 does not want to be part of the hierarchy
      THEN
        E0 SEND rejection message TO its demanding
        neighbor
        E0 finishes the protocol
        GO TO END
      ELSE
        E0 SEND acceptance message TO its demanding
        neighbor
        E0 SEND request message TO its other neighbors
        E0 RECEIVE acceptance/rejection messages FROM
        its neighbors
      END IF
    END IF
    E0 COMPUTE IN0 and OUT0
    E0 SEND IN0, OUT0 TO its participant neighbors
    E0 RECEIVE INj, OUTj FROM its participant neighbors
    Order[1]=0 /*E0 in position 1 of Order array*/
    pos=1
    FOR j=1 TO N0
      IF OUT0 < OUTj THEN
        GO TO POST
      ELSE IF OUT0 = OUTj THEN
        IF IN0 <= INj THEN
POST:     E0 is less trustworthy than Ej
          FOR p=N0+1 TO pos+2
            Order[p]=Order[p-1]
          END FOR
          Order[pos+1] = j
          FOR k=(pos+1) TO N0
            p1=Order[k]
            p2=Order[k+1]
            IF OUTp1 > OUTp2 THEN
              p1 is more trustworthy than p2
              Order[k]=p2
              Order[k+1]=p1
            ELSE IF OUTp1 = OUTp2 THEN
              IF INp1 > INp2 THEN
                p1 is more trustworthy than p2
                Order[k]=p2
                Order[k+1]=p1
              END IF
            END IF
          END FOR
        ELSE
          GO TO PREV
        END IF
      ELSE IF OUT0 > OUTj THEN
PREV:   E0 is more trustworthy than Ej
        pos = pos+1
        FOR p=N0+1 TO pos
          Order[p]=Order[p-1]
        END FOR
        Order[pos-1] = j
        IF (pos-1)>1
          FOR k=(pos-1) TO 2
            p1=Order[k]
            p2=Order[k-1]
            IF OUTp1 < OUTp2 THEN
              p1 is less trustworthy than p2
              Order[k]=p2
              Order[k-1]=p1
            ELSE IF OUTp1 = OUTp2 THEN
              IF INp1 <= INp2 THEN
                p1 is less trustworthy than p2
                Order[k]=p2
                Order[k-1]=p1
              END IF
            END IF
          END FOR
        END IF
      END IF
    END FOR
END
```

5.1.2. Construction of the Hierarchy

In this phase of the protocol, the entities act from the less trustworthy to the most trustworthy in accordance with the order established at the first phase. Therefore, the less trustworthy entity in the neighborhood acts first and the other entities must wait for the intervention of their less trustworthy neighbors.

The objective of the second phase is that each entity chooses a superior CA among the participant neighbors that issued it a certificate (trusted neighbors). Thus, when an entity E_0 acts, it looks for the most trustworthy entity of its trusted neighbors, based on the trustworthiness order established at the first phase of the protocol, and chooses this neighbor like superior CA. If L_0 is higher than L_i of the superior CA and $(L_0 + 1)$ is less than or equal to $(L_{\mathrm{MAX}} - 1)$, L_i of superior CA takes the value of $(L_0 + 1)$. In case that $(L_0 + 1)$ is higher than $(L_{\mathrm{MAX}} - 1)$, the chosen superior CA is not appropriate and E_0 must choose the next trusted neighbor like superior CA provided that this neighbor is more trustworthy than E_0. E_0 checks again if L_0 is higher than L_i of the new superior CA and so on until E_0 finds a suitable superior CA. Nevertheless, it can be possible that none of the trusted neighbors that are more trustworthy than E_0 can be used like superior CA. Thus, when E_0 concludes this procedure, it sends an association message to its neighbors informing the identity of its superior CA or a failure message indicating that it was not possible to choose a superior CA.

Later, the following less trustworthy entity in the neighborhood, according to the order established in the first phase, repeats the procedure and so on until all entities act, except for the most trustworthy entity that must not carry out this procedure because there is not a neighbor more trustworthy than it. This entity only sends a root_CA message to all its neighbors.

The entities that did not choose a superior CA in this phase of the protocol, including the most trustworthy

entity, are considered root CAs. If there are more than one root CA at the end of the second phase, the protocol must be repeated with the resulting root CAs, considering only the certificates issued among them to determine the new value of OUT_i and IN_i parameters. L_i maintain the value that they obtained during protocol execution. In addition, when the protocol is repeated, the value of L_i can be less than or equal to L_{MAX} in the second phase.

Even so, hierarchy can have more than one root CA after the repetition of the protocol. In this case, the root CAs must find the shortest path among them using an alternative method.

Root CAs send their public key to all the entities below them once the protocol has concluded in a root_CERT message.

The following algorithm describes the procedure done by each authority in the second phase of the protocol.

```
BEGIN
    IF protocol repetition THEN
      L_M = L_MAX
    ELSE
      L_M=L_MAX-1
    END IF
    IF E_0 is the most trustworthy neighbor THEN
      E_0 SEND root_CA message To its neighbors
      IF there are more than one root CA THEN
        root CAs REPEAT the protocol
      ELSE
        E_0 is the root of the hierarchy
        E_0 SEND root_CERT message TO all PKI entities
      END IF
    ELSE IF E_0 is the less trustworthy neighbor THEN
      GO TO ACT
    ELSE
      E_0 RECEIVE association message FROM its less
      trustworthy neighbors
ACT:  FOR j = N_0+1 TO pos+1
        t=Order[j]
        IF E_0 trust E_t THEN
          n = L_t
          IF L_t <= L_0 THEN
            L_t=L_0+1
          END IF
          IF L_t <= L_M THEN
            E_0 CHOOSE E_t like Superior CA
            BREAK /*exit FOR*/
          ELSE
            L_t = n
            E_0 must choose another superior CA
          END IF
        END IF
      END FOR
      IF E_0 did not choose a superior CA THEN
        E_0 SEND failure message TO all its neighbors
      ELSE
        E_0 SEND association message TO all its
        neighbors END IF
    END IF
END
```

5.2. Practical example

Fig. 5 shows a MANET with six nodes that establish peer-to-peer trust relationships among them. Arrows represent the certificates issued from one node to another.

5.2.1. Trustworthiness order among entities

If node 1 wants to be part of a hierarchy, it sends a request message to its neighbors (2, 3, 4 and 6). In addition, it proposes the maximum certification path length $L_{MAX} = 3$.

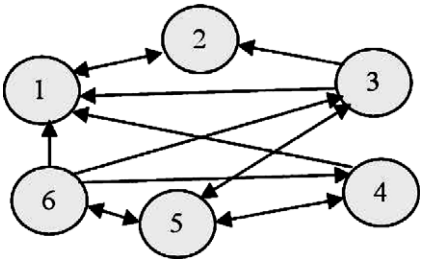

Fig. 5. Mobile ad-hoc network.

If node 2 wants to collaborate with node 1 then, it sends an acceptance message to 1 and a request message to its other neighbors (node 3), with the value of L_{MAX}.

We suppose that all the nodes want to be part of the hierarchy. Thus, node 3 receives request messages from nodes 1 and 2, so it returns them an acceptance message, and sends a request message to nodes 5 and 6. On the other hand, node 4 sends an acceptance message to node 1 and a request message to nodes 5 and 6. At this point, node 6 has already received request messages from nodes 1, 3 and 4, so it sends acceptance messages to these demanding nodes and a request message to node 5. Finally, node 5 has received request messages from all its neighbors (3, 4 and 6). Therefore, it returns them an acceptance message.

Now, all the authorities must determine their IN_i and OUT_i values and send them in an information message to their neighboring CAs. Fig. 6 shows the shared data.

Once each node obtains the data from all its participant neighbors, puts them in order from the less trustworthy to the most trustworthy. Thus, a node knows after which nodes must act in the second phase of the protocol. For example, node 1 compares its parameter OUT_1 with the one of all its participant neighbors and puts them in order: **1**, 2, 4, 3, 6. But $OUT_1 = OUT_2$, therefore, node 1 compares IN_1 and IN_2, determining that node 2 is less trustworthy than itself. Thus, the trustworthiness order for node1 is: 2, **1**, 4, 3, 6.

Likewise, the other nodes determine their trustworthiness order, that for node 2 is: **2**, 1, 3; for node 3 is: 2, 1,

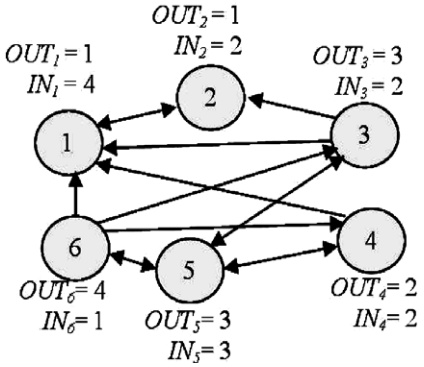

Fig. 6. Shared data.

3, 5, 6; for node 4 is: 1, **4**, 5, 6; for node 5 is: 4, 3, **5**, 6 and for node 6 is: 1, 4, 3, 5, **6**.

According to this order, node 2 must act first, later node 1 will act, then nodes 3 and 4, next node 5 and finally node 6 is the most trustworthy.

5.2.2. Construction of the hierarchy

Node 2 between its trusted neighbors (nodes 1 and 3) must look for the one that has the higher OUT_i (that is to say, node 3) and chooses it like superior CA. But L_3 is equal to L_2, therefore, now $L_3 = L_2 + 1 = 1$, that is less than $(L_{MAX} - 1)$ so the superior CA of node 2 is suitable and the trust relationship between nodes 2 and 3 is reinforced (Fig. 7). Finally, 2 sends to its participant neighbors an association message that indicates it has chosen node 3 like superior CA, so node 3 must update L_3 parameter.

Node 1 acts now. From its trusted neighbors (2, 3, 4 and 6), node 6 has the higher OUT_i. But L_6 is equal to L_1, therefore, now $L_6 = L_1 + 1 = 1$, that is less than $(L_{MAX} - 1)$, so node 1 has chosen a suitable superior CA. Fig. 8 shows the trust relationship reinforced in this case.

Now, the nodes 3 and 4 must act. Node 3 trust nodes 6 and 5, but OUT_6 is higher than OUT_5, therefore, node 3 chooses 6 like superior CA. L_6 is equal to L_3, therefore, $L_6 = L_3 + 1 = 2$, that is equal to $(L_{MAX} - 1)$, so node 6 is now the superior CA of node 3. Node 4 trusts nodes 5 and 6 and $L_4 = 0$ so it chooses node 6 like superior CA and L_6 is not modified ($L_6 = 2$).

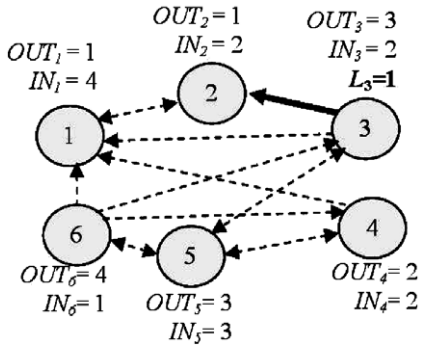

Fig. 7. Node 2 chooses 3 like superior CA.

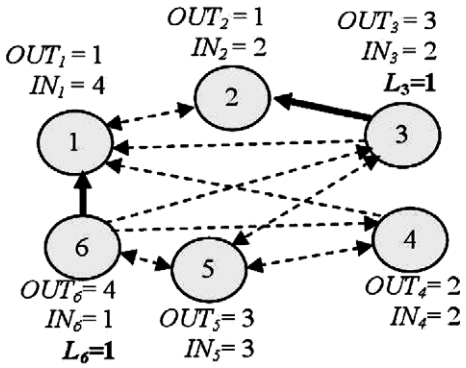

Fig. 8. Node 1 chooses 6 like superior CA.

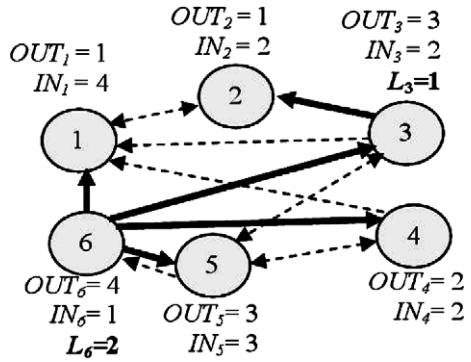

Fig. 9. Node 6 is the superior CA of nodes 3, 4 and 5.

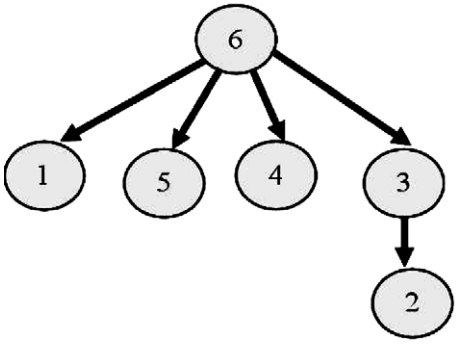

Fig. 10. Established hierarchical architecture.

It is the turn of node 5. It trusts nodes 3, 4 and 6, of which node 6 has the higher OUT_i and $L_5 = 0$, therefore, $L_6 = 2$ again. Fig. 9 shows the three trust relationships that are reinforced.

Fig. 10 shows the established hierarchy, where node 6 is now the root CA. Thus, we have obtained a virtual hierarchy whose maximum path length is 2.

6. Argumentation of PROSEARCH upwards approach

During the creation of the hierarchical PKI, entities associate among themselves by choosing root CAs. Association process can be started in two ways: started by the principal with more trust; or started by the principal with less one. That is what we call downwards and upwards approaches.

PROSEARCH protocol, explained in Section 5, makes use of an upwards approach. Now we detail why such option is adopted.

By protocol nature, a principal never rejects association with other entity with less trust. Thus many entities can associate with only one more trusted entity, but not the other way around. If the association is started by the less trust principals (upwards), the association is always successful and it is never rejected. On the other hand, if the process is started by the most trusted entities (downwards), sometimes the association will be rejected, since the less trusted entity would have been associated to other entity. This will cause a waste of messages.

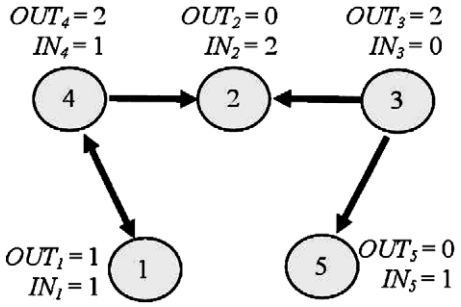

Fig. 11. Example of network.

Let us follow the example in Fig. 11 with both approaches: downwards and upwards.

With downwards approach, the most trusted entities start the protocol and request association to their less trusted neighbors with less trustworthiness level than themselves. Thus, node 4 requests association to nodes 1 and 2; and node 3 requests association to 2 and 5. After that, 2 and 1 accept association request from 4; 2 rejects association to 3; and 5 accepts association request from 3.

With an upwards approach the less trusted entities start the protocol and request association to their most trusted neighbors with more trustworthiness level than themselves. Thus, nodes 1 and 2 request association to 4; and node 5 request association to 3. No rejection is possible since a principal never rejects association with other less trusted entity. As a result neither acceptance messages nor rejection ones are needed and the overall protocol performance is clearly increased. This is the reason why an upwards approach is chosen in this contribution.

7. Implementation

Since the whole hierarchical architecture must be first fully established before starting to operate, it is very important to obtain the needed amount of time spent during the initial establishment.

We have implemented an API of PROSEARCH with JAVA code. Thus it can be directly used with any java enabled device, such as PDAs, mobile phones or laptops.

Anyway, in order to simulate PROSEARCH in a big environment with many devices, we have also developed a Java-based test environment. This simulated environment allows us to obtain simulation results of the initial establishment of the hierarchical structure.

The environment is based on the following assumptions:

- IN/OUT relations between entities are randomly chosen.
- The spent time in order to establish the secure group is measured in rounds. We define a round as a set of messages that are sent or received at the same time slot. Round time will differ from some principals to other ones, since it depends on process time, network speed, latency, etc.

8. Evaluation

The outcomes shown in the figures of this section have been obtained, calculating the average of 30 iterations by each combination of the following parameters:

- Number of entities: from 10 to 100, ten at a time.
- Relation ratio values: 0.1; 0.3; 0.5; 0.7 and 1.
- Maximum path length allowed: 3. ($L_{MAX} = 3$).

Where *number of entities* is the total amount of nodes considered in the simulation. Notice that as every node in the system is a CA, we have limited the maximum value to 100. However, although our simulator can handle the order of thousand nodes, it is showed that the evaluation figures tend to behave similarly when *number of entities* is greater than 100.

We define *relation ratio* as the probability of two randomly chosen entities to have a trust relationship. That is to say, considering two entities, A and B, there is a probability equal to relation ratio of A certifying B and the same probability of B certifying A. Particularly when *relation ratio* equals 1 it happens that every entity is certifying every other entity.

Finally, as previously stated, *maximum path length* allowed is the maximum size of a certification path allowed between two entities. We initially limited this value to 3 as we consider it a reasonable maximum value for latency when checking a certificate. However, after the simulation runs we noticed that this value is not always achieved in the different tests.

Our protocol tends to find one root CA when the relation ratio is larger than 0.5 (Fig. 12). In addition, if the relation ratio is 0.3, the average number of root CAs is 2. The number of root CAs increases with a less relation ratio value possibly because of the difficulty to find a short path between two entities. In these cases, the problem can be solved increasing the maximum path length (L_{MAX}). However it is possible that a path between two certain entities does not exist. Fig. 12 also shows that the number of root CAs is independent of the number of entities since it slightly changes insofar as the number of entities increases.

Fig. 13 shows that the certification path length is not larger than the maximum path length allowed ($L_{MAX} = 3$). When the relation ratio is 1, the average path length is minimum because there is one root CA and the other entities are associated to that root. Thus, the number of certificates from the root to the leaves is 1. Insofar as the relation ratio decreases from 1 to 0.3 the average path length increases. However, when the relation ratio is less than 0.3, the average path length decreases because it is more difficult to find a short path between two entities and several entities become root CAs.

Also, we evaluate the probability of failure (Pe) of the established hierarchy, when some entity wants to find a path to any other entity. Thus, when there is only one root CA in the hierarchy, the probability of failure of our

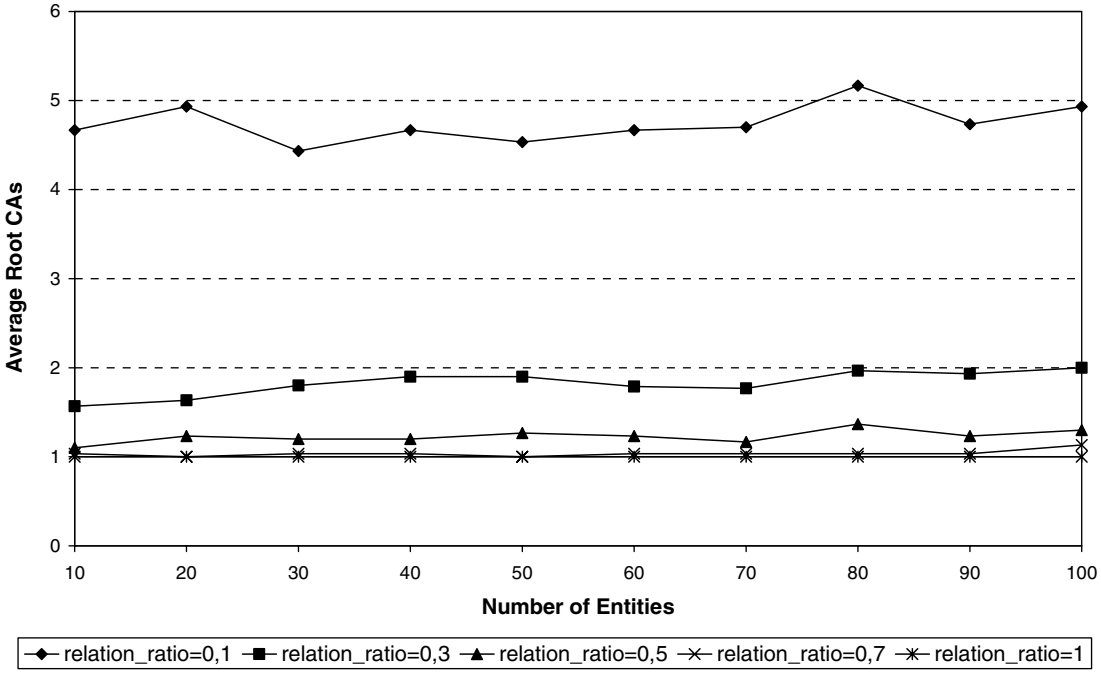

Fig. 12. Number of entities vs. average root CA.

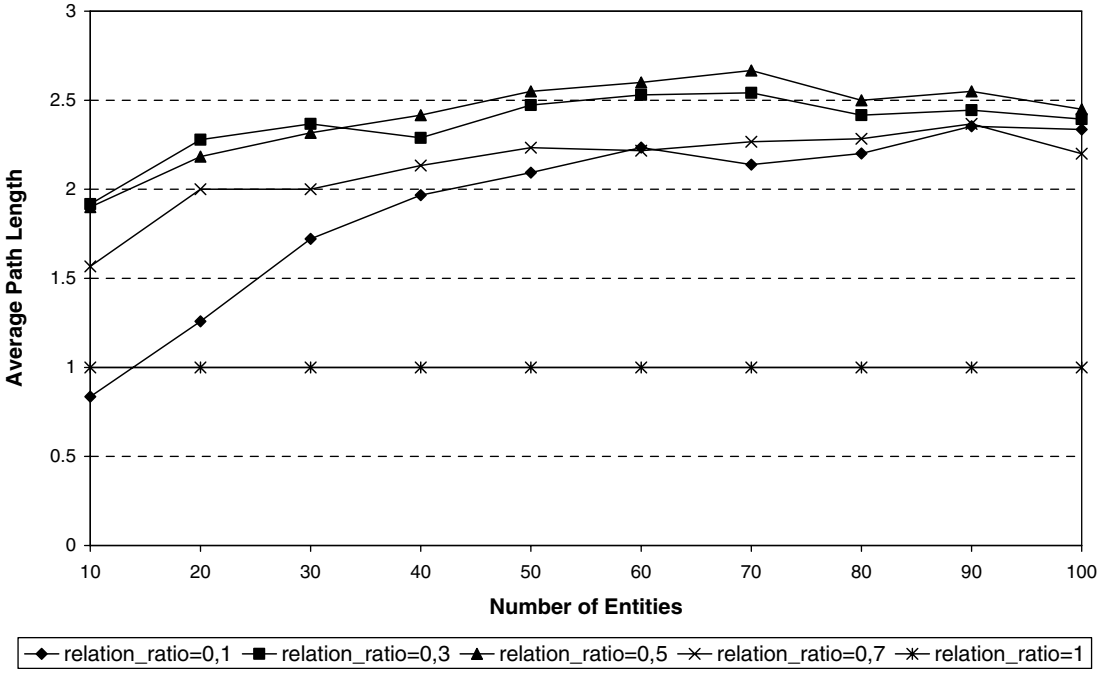

Fig. 13. Number entities vs. average path length.

protocol is zero since there is one path between each pair of entities. On the other hand, when there are several root CAs, the probability of failure will depend on the number of CAs in each subtree, reason why the probability of failure could be defined as the sum of the probability of belonging to a subtree multiplied by the probability of not belonging to the same one (see Eq. (1)).

$$P_e = \sum_{s=1}^{r} P_s(1 - P_s) \tag{1}$$

where Pe, Probability of failure; r, Number of CAs root; s, subtree identifier.

For example, if there are 3 root CAs in a network of 10 entities and the first root CA has 4 subordinated entities

(subtree 1), the second root CA has 2 subordinated entities (subtree 2) and the third root CA has 1 subordinate entity (subtree 3), the probability of failure of our protocol will be the probability that one entity in subtree 1 wants to find a path to some entity in subtrees 2 and 3 plus the probability that one entity in subtree 2 wants to find a path to some entity in subtrees 1 and 3 plus the probability that one entity in subtree 3 wants to find a path to some entity in subtrees 1 and 2 (Eq. (2)):

$$\mathrm{Pe} = 5/10(3/10 + 2/10) + 3/10(5/10 + 2/10) + 2/10(5/10 + 3/10) = 0.62 \quad (2)$$

Fig. 14 shows that the probability of failure is less than 10% for relation ratio values larger than 0.5 independent

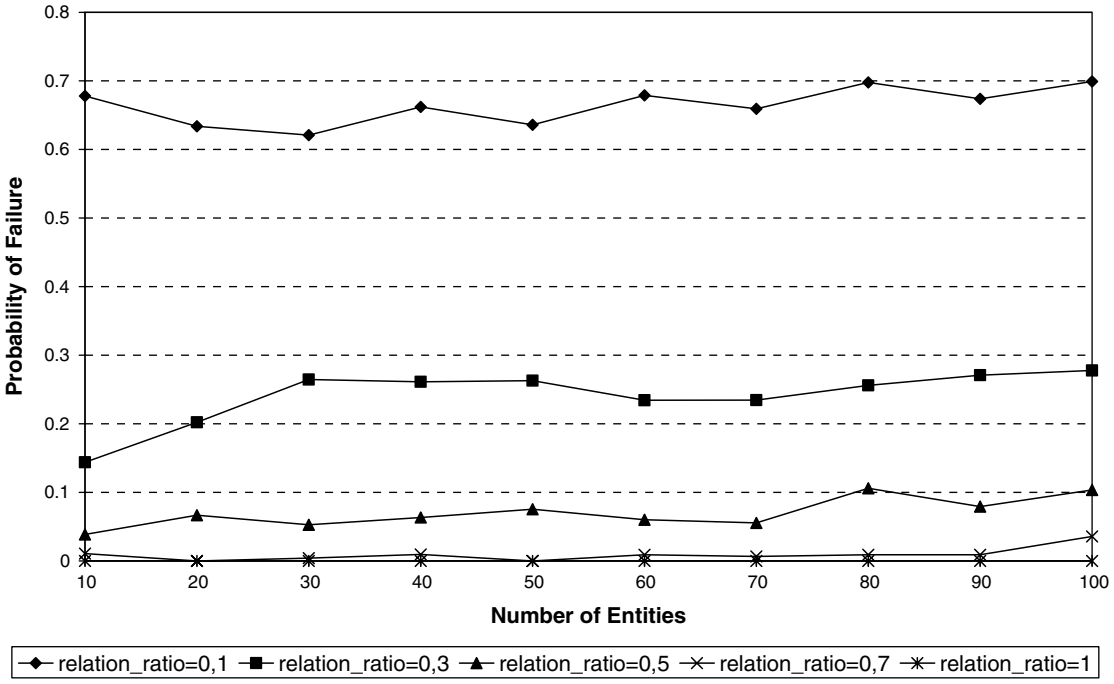

Fig. 14. Number of entities vs. probability of failure.

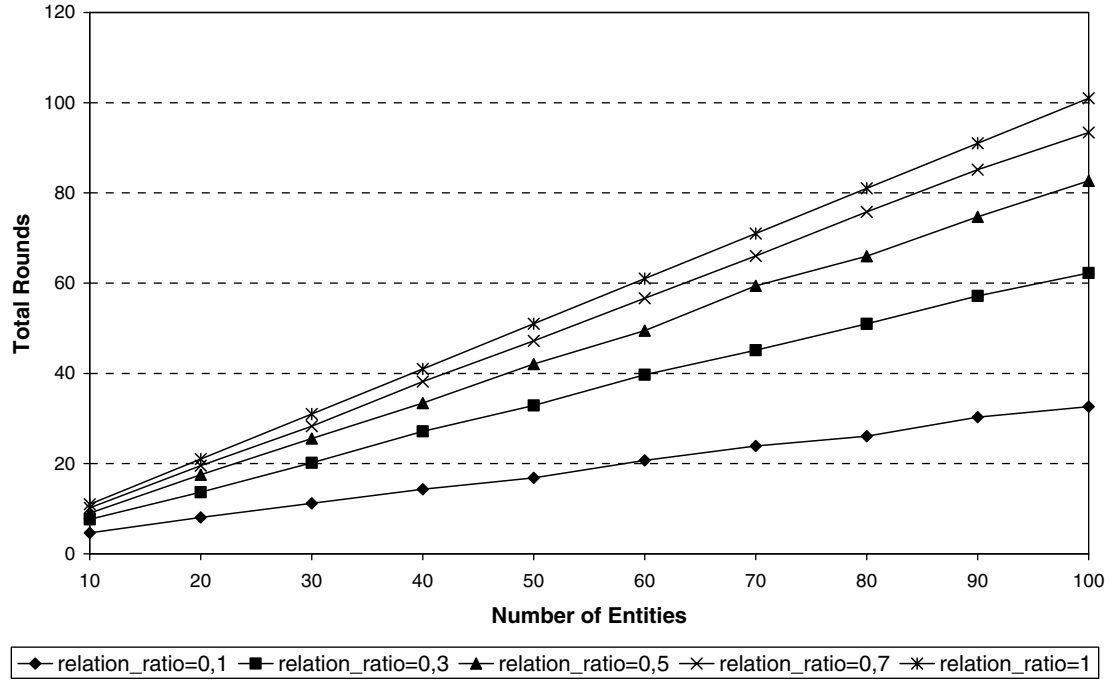

Fig. 15. Number of entities vs. average total rounds.

Table 3
DSSS system parameters used in 802.11b standard

Parameter	Value
MAC header, MAC_{hdr}	272 bits
PHY header (long), PHY_{hdr}	192 μs
RTS packet, l_{RTS}	160 bits + PHY_{hdr}
CTS packet, l_{CTS}	112 bits + PHY_{hdr}
ACK packet, l_{ACK}	112 bits + PHY_{hdr}
DIFS	50 μs
SIFS	10 μs
Control rate, $C_{control}$	2Mbit/s

of the number of entities. However, with a relation ratio of 0.3, the probability of failure is less than 30%. This means that 70% of the entities can discover easily a path among them using the hierarchy established by our protocol. On the other hand, when relation ratio is equal to 0.1, the probability of failure is too large, since there are several root CAs isolated among them.

As we mentioned before, our proposal is a mix of P2P and hierarchical models and, for this reason, it should seem that comparison between our proposal and state-of-the-art methods is straight forward. In fact, the Probability of failure parameter (provided in Fig. 14) gives the notion of the percentage of time that our protocol will behave as a hierarchical or P2P model.

In this sense, a value of the Probability of failure equal to 0.25 means that our protocol behaves as a hierarchical model 0.75% of the time. It is commonly accepted in the literature that hierarchical models are much more efficient in terms of latency checking certificate chains. Thus, Fig. 14 together with results in Refs. [4,5,7,15] can be useful for comparing our protocol to existing ones.

Fig. 15 shows that the total rounds needed to carry out our protocol increases with the relation ratio value. When the relation ratio is 1, the number of rounds is proportional to the number of entities, since the parameters OUT_i and IN_i are the same for all the entities and they are put in order in accordance with their identifier. Thus, entities must act one by one in the second phase of the protocol.

We can calculate the time needed to carry out PROSEARCH in the worst considered case, that is to say, a network of 100 entities when the relation ratio is equal to 1. The time of a successful transmission in a WLAN 802.11b (11Mbps) using RTS/CTS scheme can be determined with the values in Table 3 and Eqs. (3)–(7), taken from [17]. In addition, the average real rate (C) for this type of network, when there is a high concurrence level is 2739Mbps, according to tables in [18]. Therefore, if we assume that the maximum length of a message is 200 bytes ($l = 1600$ bits) and a propagation delay $\delta = 1$ μs, the time of a successful transmission T_S^{RTS} is 1.34 ms. Thus, the time needed to carry out 101 rounds is 135.34 ms, that is very short.

However, when relation ratio is equal to 1, it has not sense to carry out our protocol because the path from one entity to another is easy to find.

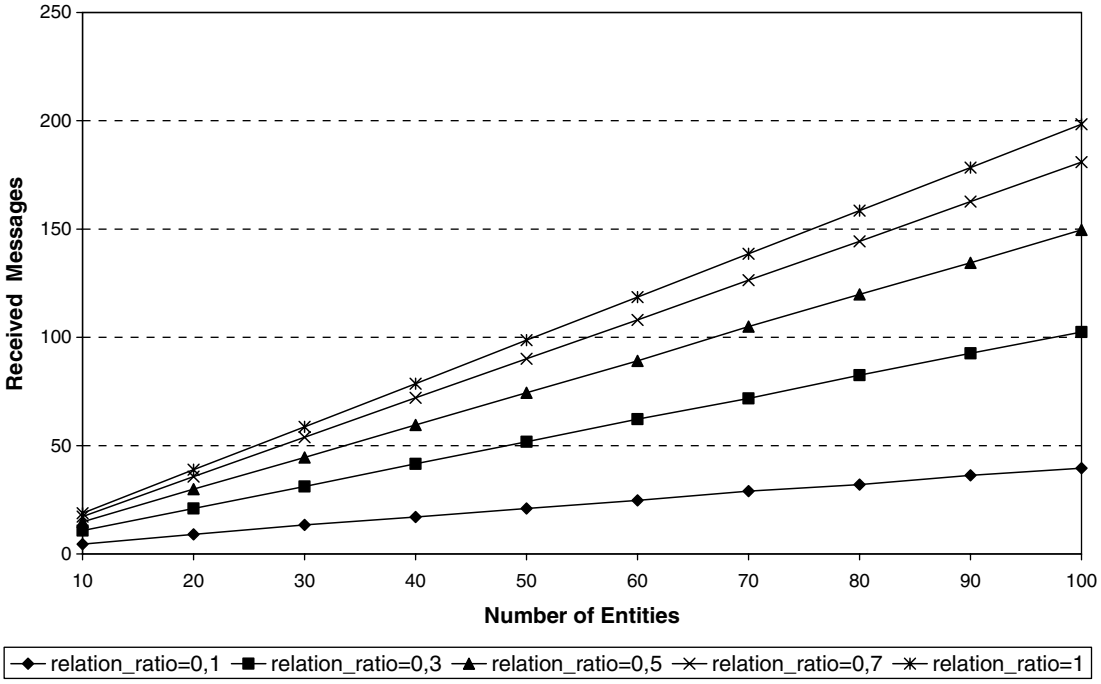

Fig. 16. Number of entities vs. average number of received messages.

$$T_S^{RTS} = \mathrm{DIFS} + T_{RTS} + \mathrm{SIFS} + T_{CTS} + \mathrm{SIFS} + T_{header} + (l/C) + \mathrm{SIFS} + T_{ACK} + 4\delta \quad (3)$$

$$T_{header} = (\mathrm{MAC}_{hdr}/C) + (\mathrm{PHY}_{hdr}/C_{control}) \quad (4)$$

$$T_{ACK} = l_{ACK}/C_{control} \quad (5)$$

$$T_{RTS} = l_{RTS}/C_{control} \quad (6)$$

$$T_{CTS} = l_{CTS}/C_{control} \quad (7)$$

Also, the number of messages received by one entity increases with the relation ratio value (Fig. 16) since each entity must communicate its decisions to all its neighbors. Thus, a large number of neighbors involves to send/receive more messages.

Figs. 15 and 16 are very similar, although the number of received messages increases faster than the number of rounds, since several entities can choose a superior CA at the same time.

9. Conclusions

Dynamism of mobile ad-hoc networks implies changing trust relationships among their nodes. Although peer-to-peer PKIs are quite dynamic and certification paths can be built although part of the infrastructure is temporarily unreachable, the discovery of certification paths is not an easy task since there can be multiple paths between two entities and all the options do not lead to the target entity. This is not the case of hierarchical PKIs, where there is only one path between two entities.

In this paper, we describe a protocol that establishes a virtual hierarchy in a peer-to-peer PKI, based on the trustworthiness of the participant entities. The level of trustworthiness of each entity is determined in accordance with two parameters: the number of issued certificates (OUT_i) and the number of received certificates (IN_i).

An advantage of our protocol is that it does not establish new trust relationships among the entities but it takes the existing relationships to establish the hierarchy. Thus, it is not necessary to issue new certificates or adjust the trust points.

In addition, PROSEARCH is adaptable to entities with limited processing and storage capacities, since hierarchy is established considering a maximum certification path length (L_{MAX}).

As Section 4 points out, thanks to unidirectional trust relationships of the hierarchy, the verifier can discover easier and more rapidly the paths than in a peer-to-peer model.

Section 8 shows that our protocol can be carried out in a short time (approximately 135 ms in the worst case) what is a token of its efficiency.

The outcomes of our simulation show that PROSEARCH is appropriate to be used in networks with a relation ratio from 0.3 to 0.7. In these cases, it is possible to find one root CA and the probability that an entity does not find a path to another is less than 30%. When the relation ratio is less than 0.3, there are several root CAs and the probability of failure is larger than 60%. On the other hand, when the relation ratio is greater than 0.7, it is easy to find a path to the other entities using the traditional methods.

The fact that our protocol not always finds a single root CA does not involve that there is not a path among the roots. In those cases, we advise to use alternative methods to find the shortest path among the resulting root CAs.

Finally, it is worthy to remark that the hierarchy found by the protocol is not always the best solution, in the sense that the minimum path length is not ever guaranteed since some existing trust relations in the P2P PKI might be lost. However, in our opinion this is not an important drawback since simulation results show that in most cases an acceptable hierarchy is found, especially considering that the simplicity of the protocol makes it easy-to-implement even for constrained devices.

Acknowledgements

This work has been partially supported by European Union FP6 Project UBISEC (IST-2002-506926) and the Spanish Research Council under the projects SECONNET (TSI2005-07293-C02-01) and ARPA (TIC2003-08184-C02-02).

References

[1] ITU-T, Recommendation X.509: Information Processing Systems – Open Systems Interconnection – The Directory: Authentication Framework (Technical Corrigendum), International Telecommunication Union, 2000.
[2] A. Shamir, How to share a secret, Communications of the ACM 22 (1979) 612–613.
[3] S. Yi, R. Kravets, MOCA: Mobile certificate authority for wireless ad-hoc networks, Proceedings of 2nd Annual PKI Research Workshop (PKI03), 2003.
[4] C. Budakoglu, T.A. Gulliver, Hierarchical key management for mobile ad-hoc networks, Proceedings of 2004 IEEE 60th Vehicular Technology Conference (VTC2004-Fall) vol. 4, 2004, pp. 2735–2738.
[5] Y. Dong, H.W. Go, A.F. Sui, V.O.K. Li, L.C.K. Hui, S.M. Yiu, Providing distributed certificate authority service in mobile ad hoc networks, Proceedings of First International Conference on Security and Privacy for Emerging Areas in Communications Networks 2005 (SecureComm 2005) 2005, pp. 149–156.
[6] J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang, Providing robust and ubiquitous security support for mobile ad-hoc networks, Proceedings of Ninth International Conference in Network Protocols (ICNP'01) 2001, 251–260.
[7] D. Dhillon, T.S. Randhawa, M. Wang, L. Lamont, Implementing a fully distributed certificate authority in an OLSR MANET, Proceedings of 2004 IEEE Wireless Communications and Networking Conference (WCNC) vol. 2, 2004, pp. 682–688.
[8] P.R. Zimmermann, The Official PGP User's Guide, MIT Press, Cambridge, MA, 1995.
[9] J.-P. Hubaux, L. Buttyan, S. Capkun, The quest for security in mobile ad hoc networks, Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001), 2001.
[10] R. Housley, W. Polk, W. Ford, D. Solo, RFC3280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2002.

[11] Y. Elley, A. Anderson, S. Hanna, S. Mullan, R. Perlman, S. Proctor, Building certification paths: Forward vs. reverse, Proceedings of Network and Distributed System Security Symposium (NDSS 2001), 2001.
[12] C. Adams, S. Lloyd, Understanding PKI: Concepts, Standards, and Deployment Considerations, Addison-Wesley, Reading, MA, 2003.
[13] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, RFC2560 – X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, 1999.
[14] W.T. Polk, N.E. Hastings, Bridge Certification Authorities: Connecting B2B Public Key Infrastructures, NIST, 2000.
[15] R. Perlman, An overview of PKI trust models, IEEE Network 13 (1999) 38–43.
[16] J. Hernandez-Serrano, J. Pegueroles, M. Soriano, GKM over large MANET, Proceedings of IEEE International Workshop on Self Assembling Wireless Networks (SAWN2005), 2005, pp. 484–490.
[17] P. Chatzimisios, A.C. Boucouvalas, V. Vitsas, Optimisation of RTS/CTS handshake in IEEE 802.11 wireless LANs for maximum performance, Proceedings of IEEE Global Telecommunications Conference Workshops, 2004 (GlobeCom Workshops 2004), 2004, pp. 270–275.
[18] G. Anastasi, M. Conti, E. Gregori, Chapter 3: IEEE 802.11 AD HOC networks: Protocols, performance, and open issues, in: S. Basagni, M. Conti, S. Giordano, I. Stojmenovic (Eds.), Mobile Ad Hoc Networking, Wiley-Interscience, New York, 2004, p. 94.

Cristina Satizábal received her degree in electronic and telecommunications engineering from Cauca University (Colombia) in 2000. Since August 2001, she belongs to the department of engineering and architecture of Pamplona University (Colombia) and currently she is carrying out a Ph.D. in Telematics engineering at the Technical University of Catalonia (Spain). Her research interest includes Public Key Infrastructure (PKI), Privilege Management Infrastructure (PMI) and Intrusion Detection Systems (IDS).

Juan Hernández-Serrano was born in Salamanca (Spain) in 1979. He is a Ph.D. student at the Technical University of Catalonia (UPC) in the Department of Telematics Engineering, Barcelona, Spain. He received his Degree in Electrical Engineering emphasis in Telecommunication from UPC in 2002. In the same year he joined the Information Security Group-ISG (http://isg.upc.es) within the Telematics Services Research Group-SERTEL (http://sertel.upc.es) at the Department of Telematics Engineering of the UPC (http://www. entel.upc.es).

His research interests include multicast security, optimal resources allocation on broadband networks and security services for multimedia transmissions. Nowadays he is working on his Ph.D. Thesis about Key Management for Large Dynamic groups in Multicast Ad-hoc Systems.

Jordi Forné was born in Barcelona in 1967. He received the M.S. degree in Telecommunications Engineering from the Universitat Politècnica de Catalunya (UPC) in 1992 and the Ph.D. degree in 1997. In 1991, he joined the Cryptography and Network Security Group at the Department of Applied Mathematics and Telematics. Currently, he is with the Information Security Workgroup within the Telematics Services Research Group at the Department of Telematics Engineering of the UPC. His research interests include network security, electronic commerce and PKI. Currently he works as an associate professor at the Telecommunications Engineering School in Barcelona.

Josep Pegueroles was born in Tortosa (Spain) in 1974. He received the M.S. degree in Telecommunications Engineering in 1999, and the Ph.D. degree in 2003, both from the Polytechnic University of Catalonia (UPC). In 1999 he joined the Information Security Workgroup-ISG (http://isg.upc.es) within the Telematics Services Research Group-SERTEL (http://sertel.upc.es) at the Department of Telematics Engineering-ENTEL (http://entel.upc.es) of the UPC (http://www.upc.es). Currently he works as assistant professor at the Telecommunications Engineering School in Barcelona-ETSETB (http://www.etsetb.upc.es). His research interests include security for multimedia networked services and secure group communications.