SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks. 2010; **3**:261–277
Published online 31 July 2009 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/sec.130

Towards an efficient reputation based hybrid key management architecture for *ad hoc* networks

Lamia Benazzouz, Mohamed Elhoucine Elhdhili*,† and Farouk Kamoun
CRISTAL Laboratory, ENSI, Manouba 2010, Tunisia

Summary

Common key management schemes are hard to deploy in *ad hoc* networks because of their infrastructure-less nature and dynamic topology. In this paper, we propose a hybrid key management architecture that is adapted for *ad hoc* networks. The proposed architecture is based on a clustering algorithm called RECA that we have designed for security purposes in *ad hoc* networks. RECA allows an efficient division of the network into clusters managed by trustworthy clusterheads that are supervised by a reputation system. The hybrid key management architecture exploits this division for scalability and availability of security services. It considers each clusterhead as a certification authority (CA) for its members and implements a totally distributed CA over only clusterheads. Simulations results show that the proposed solution enhances security services availability in comparison with existing solutions. Copyright © 2009 John Wiley & Sons, Ltd.

KEY WORDS: reputation; RECA; key management; hybrid PKI; *ad hoc* networks

1. Introduction

Secure network communications is very important and authentication is one of the most eminent preconditions.

Deploying security mechanisms in *ad hoc* networks is difficult because they are peer to peer networks with a dynamic topology. Moreover, these networks are vulnerable to various kinds of attacks such as links eavesdropping, replaying, modification and man in the middle. In addition, new types of attacks can be performed in *ad hoc* networks such as routing protocols attacks since all nodes act as routers.

To encounter attacks, cryptography methods have been recognized as the most effective tools. These methods use symmetric or asymmetric algorithms. Symmetric algorithms are rapid but have key distribution problems. These problems are more emphasized in *ad hoc* networks. Asymmetric algorithms require a mapping between public keys and their owners. This is often realized by a public key infrastructure(PKI) which operates according to the client/server architecture. However, this architecture is not suitable for *ad hoc* networks because the availability of a server is not guaranteed. Thus, PKI tasks must be distributed on a set of nodes that collaborate to ensure the global functionality of a PKI. In the literature, different works have proposed distributed PKI architectures for *ad hoc* networks [1–17]. These works have advantages but present many limits especially regarding the selection of nodes over which we will distribute PKI functionalities.

*Correspondence to: Mohamed Elhoucine Elhdhili, CRISTAL Laboratory, ENSI, Manouba 2010, Tunisia.
†E-mail: med_elhdhili@yahoo.es

In this paper, we propose a hybrid PKI (centralized and distributed) for key management in *ad hoc* networks that is implemented on a network divided by a clustering algorithm called RECA (REputation based Clustering Algorithm). RECA divides the network into clusters managed by clusterheads that are elected based on their trust levels. Their behavior is monitored during the network lifetime, using a reputation system. This makes the network converge to a state exempt of untrustworthy clusterheads. The proposed solution considers each clusterhead as a centralized PKI for its members and implements a totally distributed PKI over clusterheads.

The paper is structured as follows: in Section 2, we present key management solutions proposed in the literature. Section 3 presents a brief description of RECA and describe the proposed public key architecture based on it. In Section 4, we compare performances of the proposed architecture with existing solutions. Finally, in Section 5, we conclude the paper and outline our immediate future works.

2. State of the Art

PKI solutions proposed for *ad hoc* networks are distributed ones and employ threshold cryptography to distribute the certification authority (CA) services over the network nodes. This scheme is based on Lagrange interpolation: if we consider a polynomial function of degree $K-1$, we can rebuild this function knowing just K of its coordinates. To share the certificate signing key S between N users, we consider a polynomial function $f(x)$ of degree $K-1 < N$. Then, we provide each user i of identity id_i with its partial key $f(\mathrm{id}_i)$. The polynomial function can be reconstructed by any K out of N collaborating users.

The secret S is equal to $f(0)$. Thus, a set of K collaborating users can rebuild it. In practice, S is never reconstructed: a requesting node obtains partial information from K collaborating nodes. It combines them to obtain a final information as if it was processed using S.

In what follows, we present distributed PKI solutions for *ad hoc* networks described in the literature. These solutions are classified according to their characteristics, being totally, partially, hierarchical or cluster based distributed PKIs.

2.1. Totally Distributed PKI

This type of solution was first presented in References [1,2]. It uses threshold secret sharing to distribute the services of the PKI to all nodes. It assumes the existence of an off-line (CA) with a public/private RSA key pair (P, S). This CA publishes P, certifies the network nodes and shares S among them according to Shamir's secret sharing [18] which is not verifiable. The sharing of S is enhanced in Reference [14] by introducing verifiability: a node can verify the validity of the partial key it receives from the CA. After this phase, the CA devotes itself to generate and diffuse certificates for new joining nodes in order to make each node store all network nodes certificates. For a node, the other CA services, as certificate renewal, will be ensured by any K collaborating neighbor nodes.

This solution allows peer to peer authentication. However, it presents many major drawbacks:

1. Some CA services such as certificate renewal or revocation are devoted to all nodes in the network without taking into account neither their trust levels nor their ability (in terms of resources) to offer these services.
2. A distributed certification service cannot be done unless the participation of K or more collaborating neighbor nodes. This can result in certification failures when a node has no K neighbors.
3. A node joining the network must obtain an initial certificate from the CA. So, when the CA leaves, no new node can join the network.
4. The solution might have scalability problems since any node have to store all other nodes certificates. Moreover, it generates a great overhead as any node gets services in a distributed manner.

To solve the last problems, Khalili and Deng *et al*. [3,4] combine identity based cryptography with threshold cryptography. They propose a totally distributed PKI that uses the identity of nodes rather than their public keys. This minimizes the memory devoted to store nodes certificates. However, authors assume that each node has a unique identifier associated with a private key. This key is either generated by a private key generator or in a distributed manner [19].

This solution [3,4] solves the storage problem but generates other difficult problems: how to ensure unique identifiers in *ad hoc* networks? Moreover, an identity and a private key do not represent a real key pair? Furthermore, the confidentiality of the private key is weak.

2.2. Partially Distributed PKI

Zhou and Hass [5] are the first to present a partially distributed PKI for *ad hoc* networks. Their solution

DOI: 10.1002/sec

is reconsidered in many other works [6–12]. The partially distributed PKI differs from the totally one by distributing the CA services only on a set of nodes called servers. Server nodes are selected by the off-line CA. However, in the works of Yi and Kravets, called MOCA [6–9], servers are selected by the network operator according to their physical security, power, and trust.

In comparison with the totally distributed PKI, this solution has minimized the number of certificates to be stored in server nodes. Moreover, clients do not store certificates. However, this solution has, in addition to the limits inherited from the totally distributed solution, the following drawbacks:

1. In comparison with the totally distributed PKI, distributed services generate much more overhead because servers can be at many hops from clients. This affects also services availability.
2. Server nodes are either selected by the off-line CA or by the network operator. However, due to dynamically changing topology, selected servers can be, in some cases, a separate group of the network which can affect the distributed PKI services availability.

2.3. Hierarchical Distributed PKI

This solution is presented in Reference [13]. It applies a hierarchical, verifiable, and distributed secret sharing [20] on a network seen as a logical tree where only leafs are real nodes. The root virtual node maintains the network private key S. This key is shared among its descendants. The shares are themselves shared among the nodes of the inferior level of the tree. This procedure is repeated until leaf nodes get their partial shares.

We note that the distributed secret sharing [20] permits to a group of nodes to share a secret among them where no one knows it. Thus no node knows the private key S and the key from which it has obtained a partial share.

This solution can be seen as a collection of totally distributed PKIs where each one concerns a subset of nodes. The only dependence between groups is that they maintain private keys generated from the same root key S either directly (descendants of the root node) or indirectly (other nodes). For certification services, a node can obtain, in a distributed manner, a one level certificate within its group or a multi-level certificate that can be used in many groups.

This solution do not need an off-line CA at all. Moreover it uses verifiable secret sharing which helps to verify and identify nodes that send fake messages during certification services. However, the distributed secret sharing, as described in Reference [20] assumes the existence of a secure link between each pair of nodes. This makes the method inadapt for *ad hoc* networks. In addition, authentication is impossible for nodes in different groups unless they have multi-level certificates which is not easy to implement. Finally, this solution uses complex algorithms for keys and certificates updates.

2.4. Cluster Based Distributed PKI

This type of solution [15–17] takes advantage of clustering to implement efficient distributed PKI. Clustering consists in dividing the network into clusters managed by clusterheads. It allows routing efficiency, enhanced network management, transmission management etc.

Authors of [15] suppose a network divided into cluster according to CGSR algorithm (Clusterhead Gateway Switch Routing) described in Reference [21]. One of the clusterheads is randomly selected to be the network CA. This CA publishes its public key P in the network and shares its private key S among *authenticated* clusterheads according to Shamir's secret sharing [18]. Then, every clusterhead gets a certificate signed by S in a distributed manner i.e., by requesting K other clusterheads that *authenticate* it. Finally, every ordinary node requests its clusterhead that *authenticates* it and distributively signs its certificate by collaborating with $K - 1$ other clusterheads.

The main concern about this solution is that key distribution assumes authenticated nodes. This authentication is based on 3-way handshake messages exchange which is not reliable. Another issue is that a malicious node can become the network CA because the CA is selected from a set of clusterheads that are not necessary trustworthy.

In Reference [16], the private key S is distributively generated by clusterheads using verifiable secret sharing. For authentication, nodes use certificates distributively signed by S. When a new node join the network, it is considered as a guest. It starts collecting warranty certificates signed by private keys of full member nodes i.e., nodes that have certificates signed by the network private key S. Then, it sends these warranties to K clusterheads that distributively sign a certificate to it.

This solution has three main drawbacks. First, it uses distributed verifiable secret sharing which necessitates a secure channel between each pair of clusterheads [20]. Second, it does not specify any clustering algorithm to elect clusterheads to which it

DOI: 10.1002/sec

assigns security services without knowing neither their trust nor their capacity to offer such services. Finally, it addresses only the certification of new nodes. What about nodes that are present at network bootstrapping?

In Reference [17], an off-line *CA* selects less mobile nodes as clusterheads according to a clustering algorithm and certifies them. Each clusterhead selects the more secure *K* nodes from its members as Private Key Generators *PKG* and forms with them a totally distributed PKI using identity based cryptography. PKG nodes generate private keys for cluster members distributively. Authentication between clusterheads is done like in PGP while authentication between ordinary nodes is done according to identity-based authentication as explained in Section 2.1.

Although this solution presents a clustering algorithm, it does not consider nodes trust levels for clusterheads election. This parameter is the main issue for securing the network. Moreover, this solution uses an off-line *CA* to which it assigns many functionalities as updating keys and registering new nodes. Thus, when the *CA* is damaged, many key management services fail. Finally, the use of identity based cryptography does not guarantee the confidentiality of nodes private keys since they are known by their clusterheads.

2.5. Conclusion

Existing distributed PKI solutions for *ad hoc* networks have advantages but still carry many limits. These solutions provide peer to peer authentication but most of them require an administrator to certify nodes joining the network for the first time. Furthermore, security tasks are distributed over nodes without taking into account neither their ability to offer these tasks (energy, mobility etc.) nor their trust level. Even in the MOCA solution [9], the selection of nodes is static while *ad hoc* networks are dynamic and vary in time. Cluster-based distributed PKI solutions solve this problem by considering clusterheads as distributed CA. However, clusterheads can be malicious as they are not elected based on their trust. Moreover, the private key *S* is either generated by a randomly selected clusterhead [15] which can be malicious or distributively generated by clusterheads [16] which necessitates secure channels between each pair of clusterheads. Finally, these solutions do not take advantage of clustering scalability because each node needs the collaboration of *K* other clusterheads to obtain a service rather than contacting only its clusterhead.

In our works, we opted for a hybrid key management solution that offers both centralized and distributed PKI services to ensure scalability and secure nodes collaboration. In fact, each clusterhead is a (CA) for nodes in its cluster and all clusterheads collaborate to offer distributed PKI services for each clusterhead.

Unlike existing solutions, only trustworthy nodes will be considered to offer security services. This makes the proposed public key architecture more secure. In addition, scalability is needed to palliate to distributed servers congestion while serving a great number of clients like in Reference [5]. Scalability limits also the quantity of information stored in a distributed node like in Reference [1] where each node stores all delivered certificates in the network. The proposed solution is scalable because a clusterhead serves a limited number of nodes and stores a limited number of certificates (its members certificates and clusterheads certificates). Furthermore, the proposed architecture aims to palliate the limits of existing PKI solutions in terms of availability of services and overhead generated. Availability is ensured by the clustering algorithm proposed to divide the network. In fact, this algorithm adapts itself to all network topology changes and clusterheads misbehavior by electing new trustworthy clusterheads and assigning nodes to new clusters.

In the next section, we will describe the proposed hybrid PKI based on the RECA clustering algorithm.

3. The RECA Hybrid PKI

The solution we propose presents two levels: an efficient clustering of the network using RECA and a hybrid PKI that takes advantage of that clustering. In this section, we will describe in a first step RECA. In a second step, we will present the hybrid PKI architecture and its functionalities.

3.1. Reputation Based Clustering Algorithm: RECA

RECA [22] is a reactive clustering algorithm that we have designed to provide a network division based essentially on nodes trust level. It allows an efficient and stable division of the network into clusters (groups) with trustworthy clusterheads. The behavior of the clusterheads is supervised according to a reputation system. This system allows to detect and eliminate malicious clusterheads. In what follows, we will present, in a first step, RECA algorithm. In a second step, we present its relevant performances.

DOI: 10.1002/sec

3.1.1. RECA algorithm

RECA algorithm is composed of an election protocol and an update policy. For clusterheads election, RECA uses a metric that is a linear function of the normalized node trust level T, relative mobility M, energy E, distance to neighbors D, and connectivity C. Only nodes with a trust level greater than a threshold can declare themselves as candidates for being clusterheads by broadcasting a message containing their metric components (T, M, E, C, and D). The node that has the minimum weight compared to its neighbors declares itself as a clusterhead. Then, the other nodes attach themselves to clusterheads. After the first division of the network, nodes will not broadcast their trust level T as this parameter will be computed by neighbors thanks to the reputation system. In fact, each node i will maintain a list of nodes with corresponding trust levels. These trust levels are initialized to the values broadcasted by nodes at network bootstrapping. Then, they will be updated according to nodes behavior: node i decreases j's trust level if it detects that:

- j did not forward its routing or data packets,
- j provides it with erroneous management information (after verification),
- j refuses to serve it.

Trust levels are safeguarded using a local authenticated method i.e., only the owner node can modify them after authentication.

Figure 1 shows an example of RECA network division. clusterheads are shown as squares and ordinary nodes are shown as circles. All shapes are dotted with the node identity and its trust level. Gray nodes are nodes that can declare themselves as candidate for being clusterheads because their trust levels are greater than the considered trust threshold $Trust\_min = 60$.

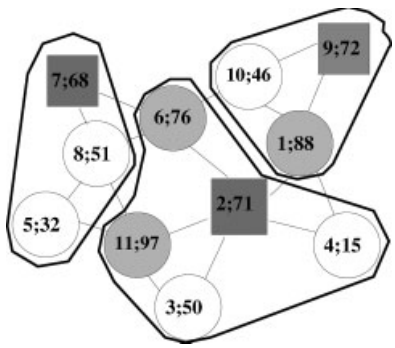

Fig. 1. Example of RECA network division.

Clusters are updated as a reaction to topology changes or malicious behavior of a clusterhead. Through time, the reputation system allows each node to update the trust levels of its neighbor nodes (ordinary nodes or clusterheads). Periodically, an ordinary node consults its neighborhood. If it finds that its clusterhead is unreachable or misbehaving, it attaches itself to a new trustworthy one or invokes the clustering algorithm locally. In the same way, if two clusterheads become neighbors, one of them (with the highest weight) will resign. Thus, keeping an adequate number of clusters in the network.

3.1.2. RECA evaluation

We conducted simulations to evaluate RECA performances in the presence of liars [22]. In this section, we will present relevant simulation results. For more details, readers are invited to view the full evaluation described in Reference [22]. We adopted the following assumptions:

- A1: The network contains a maximum of 50 per cent of liars which lie maximally by publishing the best metrics.
- A2: Nodes exchange data packets and monitor the behavior of their clusterheads by observing if they relay data packets and routing information or not.
- A2: Liars do not forward neither routing information nor data packets.

In what follows we report simulation results that deal with two main performance criteria:

- Number of trustworthy and untrustworthy clusterheads: this criterion will evaluate the convergence of RECA to a state exempt of untrustworthy clusterheads. This criterion is important since elected clusterheads will be assigned security management functionalities. Thus, it is highly recommended to consider only trustworthy clusterheads to offer such functionalities.
- Number of hops to clusterheads: it is better to have clusters where nodes are at one hop to their clusterheads. This will avoid using routing protocols and then possible attacks on them. In addition nodes will be sure that their clusterheads are trustworthy as being rated by themselves after direct monitoring.

Simulation parameters are listed in Table I. The weighting factors for computing weights are all equal to 0.2.

DOI: 10.1002/sec

Table I. Simulation parameters.

Parameter	Value
Simulation area	$1000 \times 1000\,m^2$
Number of nodes	$100 \rightarrow 300$
Transmission range	$50\,m \rightarrow 200\,m$
Mobility Model	random waypoint
Pause time	10 s
Nodes maximum speed	$0 \rightarrow 10\,m/s$

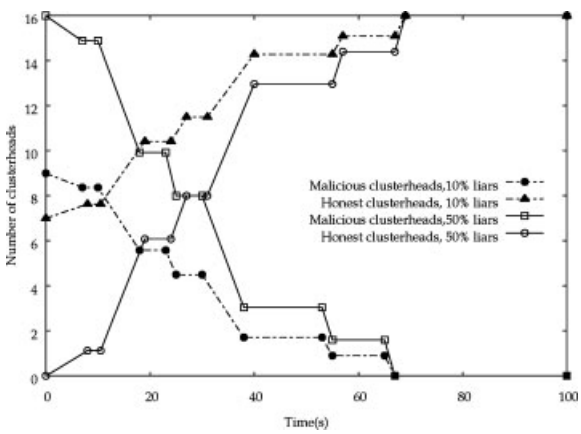

Fig. 2. RECA convergence evaluation, 100 nodes, $Tx = 150\,m$, $speed = 2\,m/s$, $Trust\_min = 50$.

Figure 2 shows the number of both malicious and honest clusterheads with respect to simulation time. We observe the following results:

1. At network bootstrapping, the number of malicious clusterheads increase with the increase of liars proportion until all network clusterheads are malicious(50 per cent of liars). This is coherent since liars always broadcast better metrics than those of honest nodes. Thus, if every honest node has at least one liar as a one hop neighbor, the first division of the network will result in only malicious clusterheads.
2. After a period of time that does not exceed 10 s, the number of malicious clusterheads begins to decrease because the reputation system begins to obtain information about nodes reputations. This allows ordinary nodes to leave the clusterheads that they judge malicious and attach themselves to trustworthy ones.
3. When the number of malicious clusterheads decrease, the number of honest ones increase. When all malicious clusterheads are eliminated, we observe that a stable and adequate number of clusterheads is kept in the network (about 16 clusters). This proves that, despite mobility, RECA gives a suitable division of the network in terms of clusters number.
4. After an average of 70 s only trustworthy clusterheads are considered in the network. This is deeply related to the assumptions $A2$ and $A3$ we have made above, which allow the reputation system to collect nodes behavior and detect untrustworthy clusterheads.

Figure 3 shows nodes proportions with respect to the number of hops to a clusterhead for varying liars (Figure 3(a)), transmission range (Figure 3(b)), mobility (Figure 3(c)) density (Figure 3(d)), and trust threshold $TRUST\_min$ (Figure 3(e)).

For all figures and for a trust threshold less than 50 per cent, we notice that the proportion of liars has practically no impact on the number of hops to clusterheads. This can be explained by the fact that nodes always find trustworthy clusterheads as neighbors or prefer to declare themselves clusterheads rather than attaching themselves to unknown distant clusterheads. We also notice that the majority of nodes (>70 per cent) are at one hop to their clusterheads. This enhances the availability of services offered by clusterheads to their members and will minimize the use of routing protocols when exchanging management data between a node and its clusterhead.

3.1.3. Conclusion

RECA allows an adequate division of the network into clusters. Simulation results have shown that it converges to a stable and convenient network division with no untrustworthy clusterheads and mainly one hop members. Consequently, the decomposition of the network supplied by RECA can be exploited to secure *ad hoc* networks applications or routing protocols.

In the next section, we will describe how we will exploit RECA to settle down a hybrid PKI for key management in *ad hoc* networks.

3.2. The Proposed Hybrid PKI Architecture

In this section, we will present, in a first step, the network and adversary models. In a second step, we will describe the proposed architecture. Finally, we will highlight the system bootstrapping and services.

3.2.1. Network model

This work considers an *ad hoc* network with bandwidth-constrained, error-prone, and insecure

DOI: 10.1002/sec

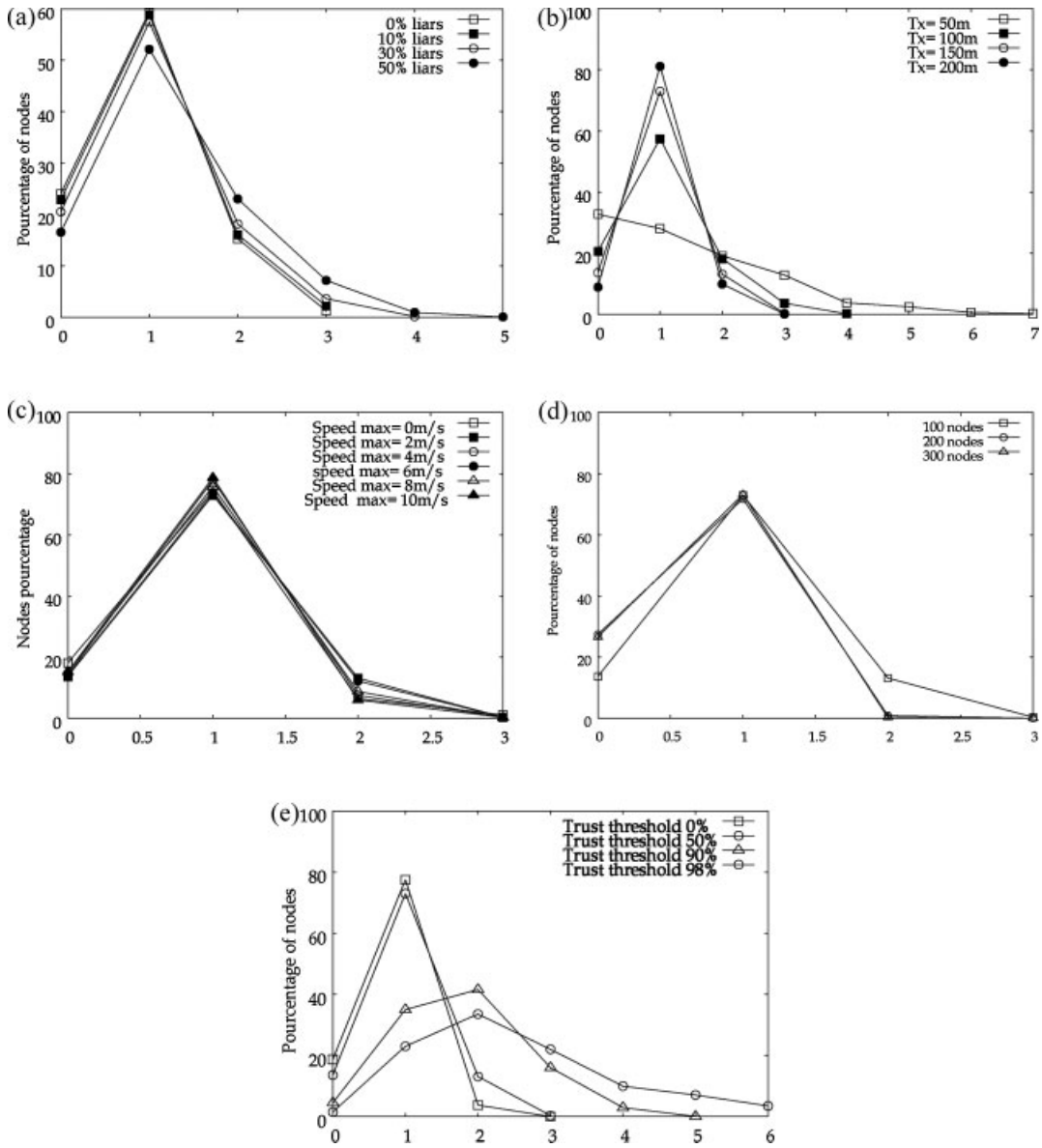

Fig. 3. Number of hops to clusterheads. (a) Impact of liars proportions on the number of hops to clusterheads, 100 nodes, speed = 2 m/s, Tx = 150 m, trust threshold = 50; (b) impact of nodes transmission range on the number of hops to clusterheads, 100 nodes, speed = 2 m/s, trust threshold = 50; (c) impact of mobility on the number of hops to clusterheads, 100 nodes, Tx = 150 m, trust threshold = 50; (d) impact of density on the number of hops to clusterheads, Tx = 150 m, speed = 2 m/s, trust threshold = 50; (e) impact of trust threshold on the number of hops to clusterheads, Tx = 150 m, speed = 2 m/s, trust threshold = 50.

wireless channels. Each node has a public/private key pair that it has computed by itself and a unique identifier. It also runs an instance of the clustering algorithm RECA and is equipped with a reputation system to identify misbehaving nodes among its one-hop neighborhood. Being mobile, nodes can join and leave the network at any time or roam from a cluster to another one.

3.2.2. Adversary model

We assume that an adversary can easily get a copy of all exchanged messages between the network nodes by listening the wireless channel. However, we assume that this adversary can corrupt up to $K - 1$ nodes before updating key shares. This will

DOI: 10.1002/sec

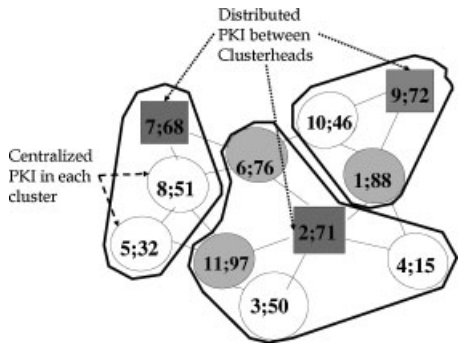

Fig. 4. RECA hybrid PKI.

prohibit an attacker from discovering the distributed PKI private key. We assume also that a malicious adversary may cause corrupted parties to divert from the specified protocol in any way, such as sending fake replies to distributed certification requests. A malicious adversary can itself send fake replies to nodes requesting security services. All these attacks will be avoided in our solution by using cryptography methods and Feldman verifiable secret sharing [23].

We also assume that a node can give erroneous metric components to be elected as a partial CA. This will be avoided by the reputation system proposed in RECA.

3.2.3. Network architecture

Unlike existing distributed PKIs [1–6,8–12,14–17], RECA public key architecture (see Figure 4) is not completely distributed. It applies RECA clustering algorithm to divide efficiently the network into clusters with trustworthy clusterheads. It considers each clusterhead as a centralized PKI for its members and implements a totally distributed PKI over clusterheads by applying Feldman verifiable and proactive secret sharing [23] described in the appendix. Thus, an ordinary node is served by its clusterhead while a clusterhead is served by a coalition of K other clusterheads, in a distributed manner.

3.3. Secured PKI Bootstrapping Protocol: SBP

PKI bootstrapping consists mainly in providing each node with a valid certificate and a partial share of the CA private key S. This can be done by a centralized SBP [24] i.e., by the CA, using Feldman's protocol described in the appendix or in a distributed manner by a totally distributed SBP [20] that is equivalent to N parallel and partially synchronous executions of the centralized SBP, where N is the number of network nodes.

We have opted for a centralized SBP for the following reasons:

1. The totally distributed SBP [20] needs secured and authenticated point to point channels and a dedicated broadcast channel which is so difficult to ensure in *ad hoc* networks where key material is not yet settled down.
2. Getting a certificate in a centralized SBP needs only one authentication by the CA which can be done using external means. After this phase, the CA can settle down secured and authenticated point to point channel with each node for the sharing of its private key.
3. The totally distributed SBP generates a large overhead and can be victim to numerous collisions since it needs N parallel executions of the centralized SBP.
4. It is easier to secure one rather than N executions of centralized SBP.

The system bootstrapping begins after the network division by the clustering algorithm RECA. In this phase, we assume that an existing trustworthy node acts as an initial CA. The existence of this node depends on applications. It can be the chairman in a conference or the leader in a military application. Its main role is to certify existing clusterheads, distribute its private key over them according to Feldman's verifiable secret sharing scheme and finally gives them its certificate according to the following details:

1 Each clusterhead C sends a signed initialization request INIT_{REQ} to the CA that contains its identity id_C, public key and other information that will be useful for the creation of its certificate.
2. When the CA receives a request from a clusterhead C, it verifies the signed INIT_{REQ}, validates the related information (validation is deeply related to application), then executes the following operations:
   - Computes a partial key $S_C = f(\text{id}_C) \bmod p$ for C.
   - Creates a certificate $cert_C$ for C.
   - Sends the set $(cert_C, enc(S_C), cert_{CA})$ to C where $enc(S_C)$ is S_C encrypted by C's public key. This will ensure the privacy of S_C and thus of S because discovering K partial shares reveals S using lagrange interpolation.
   - Sends the certificates of the already initialized clusterheads to C.

DOI: 10.1002/sec

- Saves $C's$ certificate and sends it to all already initialized clusterheads.

3. The clusterhead C receives $(cert_C, enc(S_C), cert_{CA})$ and the certificates of the others and saves them. It decrypts $enc(S_C)$ to get its partial share of the CA private key S. Once the CA serves existing clusterheads, it gives up its role as CA and can quit the network.

SBP Security discussion: In the proposed SBP, we avoid private CA key computation after discovering K partial keys given by the CA. In fact, in our solution, the CA encrypts partial keys with node public key before sending them.

The secrecy of S proved above can be broken if an adversary launches multiple bootstrapping requests with distinct identifiers and public keys. If the CA do not detect such an attack while validating requestors information, the adversary can obtain K partial shares and deduce the CA private key S. Thus, the robustness of SBP is deeply related to authenticating and tracing requestors using the reputation system.

3.4. Distributed Inter Clusters Services Against Active Attacks

In this section, we are going to describe certification services ensured by the distributed public key architecture that concerns only clusterheads. These services require the collaboration of at least K already initialized clusterheads.

3.4.1. Obtaining the other clusterheads certificates

A new clusterhead C_n obtains, first, the other clusterheads certificates. To do, it localizes one neighbor clusterhead by broadcasting a search request with (TTL $= 2, 3 \ldots$ max). The localized clusterhead sends back to C_n, the clusterheads certificates list that it maintains as well as the root CA certificate. If a requested clusterhead do not respond, C_n decreases the corresponding trust level and requests another clusterhead.

3.4.2. Certificate delivery and renewal

Due to RECA updates, new clusterheads can be elected and must obtain certificates signed by S. For this purpose, we use a distributed certificate delivery protocol inspired from Luo protocol [25] and that have the following properties:

1. A clusterhead needs only to send a certification request to clusterheads and get responses, from which, it chooses K valid ones and combines them to get its certificate. The requesting node neither requires to localize any group G of nodes nor to specify any node identity like in References [1,13].
2. The algorithm handles active attacks by finding out invalid received partial certificates as responses to its request. This permits to encounter active attacks and to trace clusterheads that send fake replies and thus supplies the reputation system with useful information.
3. The protocol can complete the certification service within one round if the requesting node finds out K valid partial certificates from the received partial certificates, for the first time. In fact, a clusterhead can ignore the node request or send a fake reply because it is malicious. In addition, clusterheads responses can be lost due to collisions.
4. Unlike the algorithms presented in References [1,13], valid partial certificates of distinct rounds can be combined.

In what follows, we give the algorithm details where C_n is a new clusterhead:

1. C_n sends a signed certificate signature request to clusterheads. It includes in this request its identity, public key, and a digest of its certificate to be signed. We will call this digest *cert*.
2. Every clusterhead C with partial key S_C which receives the request, verifies the signed request and C_n's information then:
   - Computes a partial signature $\mathrm{PS}_C = cert^{S_C} \bmod q$ where q is a parameter of the verifiable secret sharing [23].
   - Generates a random value $u \in Z_q$ and computes $A_1 = g^u$, $A_2 = cert^u$, $\alpha = \mathrm{hash}(g^{S_i}, PS_C, A_1, A_2)$ and $r = u - \alpha \times S_C$. Values A_1, A_2, and r will be used to verify partial signatures.
   - Sends (PS_C, A_1, A_2, r) to C_n.
3. C_n gets, for every requested clusterhead C, g^{S_C} which is publicly revealed by C [23], generates $C = \mathrm{hash}\,(g^{S_C}, PS_C, A_1, A_2)$ then validates the received partial signatures by verifying that $g^r(g^{S_C})^\alpha = A_1$ and that $cert^r(cert^{S_C})^\alpha = A_2$.
4. C_n combines K valid partial signatures to form a valid signature for its certificate.

After this step, C_n sends (unicast) its certificate to the network clusterheads. Thus, at any time, all initialized clusterheads have the same certificate list.

DOI: 10.1002/sec

For certificate renewal, a node executes the same operations as in obtaining a certificate for the first time. But, requested clusterheads have to check if the node certificate has been revoked or expired.

Discussion: The distributed certification protocol described above allows a node to verify the validity of each partial share before combining all shares. It permits also to trace nodes which have sent fake shares. Thus, the protocol is robust *via* verifiability and traceability. This can supply the reputation system with useful information concerning clusterheads trust levels.

All malicious behavior detected by a node, such as sending fake replies, will have an impact on the reputation system by decreasing the trust levels of misbehaving clusterheads. The node that detects a misbehavior informs its one hop neighbors.

3.4.3. Obtaining a partial key by a new clusterhead C_n

After obtaining a valid certificate, a new clusterhead C_n is incorporated into the distributed CA by being provided with its own share of the CA certificate signing key S. First C_n have to localize K clusterheads because each one of them must be aware of all collaborating clusterheads when it computes a partial key share S_{C,C_n} for C_n because this value depends on all collaborating clusterheads identities. For this purpose, we have defined the following steps:

1. C_n chooses K clusterheads (it has their certificates) and sends them a signed partial key request.
2. After authenticating C_n's request using the corresponding certificate, clusterheads which wish to participate in the incorporation process of C_n reply with a signed message containing their corresponding identities.
3. If C_n receives less than K responses, it requests additional clusterheads. Moreover, it updates its reputation system; it decreases the trust level of the clusterheads that have not answered and decreases the trust level of those that have answered.

After localizing K clusterheads, C_n obtains its own share of the *CA* certificate signing key S according to the following steps:

1. C_n sends to the K localized clusterheads a signed message that contains all their identities.
2. Each requested clusterhead C replies with a signed, encrypted, and *shuffled* partial share [1] $S_{C_n,C}$ such that $S_{C_n,C} = S_C L_{\mathrm{id}_C}(\mathrm{id}_{C_n})$
3. C_n validates each partial share by verifying if the equation $S_{C_n,C} = \prod_{t=0}^{K-1} (A_t)^{\mathrm{id}_C^t L_{\mathrm{id}_C}(\mathrm{id}_{C_n})} \pmod p$ holds. This equation permits also to trace which clusterheads have replied with fake partial shares. Their trust levels stored by C_n will be decreased.
4. If all partial shares are valid, C_n computes its partial key $S_{C_n} = \sum_{i=1}^{K+1} \overline{S_{C_n,i}} \bmod q$. Otherwise, C_n ignores all other partial shares because they are related to the identities of all clusterheads of the chosen group G. Consequently, C_n repeats all above steps and do not consider nodes that have sent him fake replies since the first steps.

Discussion: This protocol allows both verifiability and traceability. It permits to verify received partial keys one by one. This encounters active attacks against the considered service. It also permits to trace clusterheads that send fake partial shares and this updates the reputation system. Moreover, this protocol ensures the confidentiality of partial keys by encrypting them with the requestor public key and thus eliminates possible discovery attack of the CA private key S. This is possible because clusterheads obtain certificates prior to obtaining their partial keys.

3.4.4. Inter cluster authentication

Inter cluster authentication is done as in Figure 5. Suppose that a node N_1, member of a cluster CL_1 with clusterhead C_1, wants to validate a node N_2 member of a cluster CL_2 with clusterhead C_2. N_1 has not the public key of CL_2 in order to verify N_2's certificate. Thus, it sends a C_2's validation request to its clusterhead C_1 (including N_2's certificate). C_1 knows that N_2's certificate have been signed by C_2, it verifies N_2's certificate using C_2's public key (retrieved from C_2's certificate stored in the database maintained by C_1). Then, it responds to N_1.

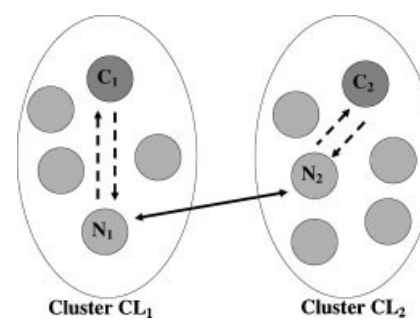

Fig. 5. Inter cluster authentication.

DOI: 10.1002/sec

We notice that inter cluster authentication minimizes the overhead; only two messages are exchanged between N_1 and its clusterhead C_1. Moreover, C_1 do not need to communicate with C_2 because it stores the corresponding certificate. Consequently authentication succeeds even if C_2 is unreachable.

3.4.5. Certificate revocation mechanism

Clusterheads certificate revocation mechanism is deeply related to the reputation system: a clusterhead revokes the certificate of another one if the corresponding trust level becomes inferior to a fixed threshold. A clusterhead builds up the trust level of peer ones based on information that it collects from:

1. *Interaction with other clusterheads:* This is done especially during the hybrid PKI functionalities (see Figure 6):
   (a) *Certificate delivery and renewal (type* 1*):* When a clusterhead detects that another one has sent him a fake partial signature (after verification), it decreases that clusterhead trust level.
   (b) *Obtaining a partial key:* When a clusterhead detects that another one has sent him a fake partial key (after verification), it decreases that clusterhead trust level.
   (c) *Obtaining the certificates of the other clusterheads:* When a clusterhead detects that another one has ignored its request for obtaining the list of clusterheads certificates, it decreases that clusterhead trust level.
   (d) *Inter cluster authentication*: When this service is executed, a clusterhead C_1 is requested by one of its members to verify a C_2 member node certificate. If it refuses to do that work, its trust level stored by the requestor will be decreased.
2. *One hop neighbor nodes (type* 2*):*A clusterhead takes decisions about other clusterheads behavior based on second hand informations (see Figure 6). These are informations sent by their one hop nodes that are at one hop to other clusterheads. We remind that, in the proposed hybrid PKI, a clusterhead cannot have one hand information about other clusterheads since RECA prohibits any two neighbor clusterheads.
3. *RECA level(type* 3*):* When an ordinary node detects that its clusterhead is behaving maliciously (not forwarding packets, not responding to attachment requests, deleting hello mesages etc.), it attaches itself to a new clusterhead and sends to it an accusation against its old clusterhead.
4. *Other clusterheads (type* 4*):* If a clusterhead realizes that the trust of a peer one has become inferior to a fixed threshold, it puts the corresponding certificate in its local *CRL* (Certificate Revocation List) and broadcasts a signed accusation against it. If another clusterhead receives such an accusation, it considers it if it is generated by a clusterhead whose certificate is not revoked and mark the accused node as suspect.

The update of the trust level of a peer clusterhead is related to the origin of information such that $d(type1) > d(type2) > d(type3) > d(type4)$ where d is a function that decreases the trust level i.e., the trust level of a peer clusterhead is decreases more according to information of type1, then type2 etc.

3.4.6. Intra cluster services

Intra cluster services are services offered by each clusterhead to its members. Knowing that each clusterhead is a (CA) for its members, these services will be the same as the ones offered by a centralized PKI:

1. Certificate creations.
2. Certificate publication.
3. CRL management.
4. Certificates storage.
5. Certificate renewal.

We remind that this is an original contribution in the proposed hybrid public key architecture. It has many advantages as it enhances availability and minimizes the overhead. In fact, in the hybrid PKI, an ordinary node obtains certification services by contacting only its clusterhead which can be one of its one hop

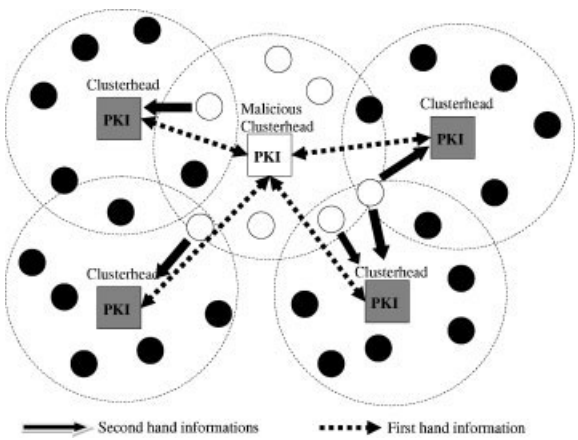

Fig. 6. Clusterheads trust level exchanges.

DOI: 10.1002/sec

neighbors with a high probability as proved in our previous works [22,26]. Consequently, the availability of security services will be roughly guaranteed for ordinary nodes, unlike solutions presented in the literature that make any node dependent on K other nodes to get a certification service.

Intra authentication is based on certificates. These certificates are signed by the same clusterhead. Thus, a node authenticates another one by verifying the corresponding certificate its clusterhead public key without the need of any other entity.

3.4.7. RECA key management updates

Here, we are going to describe how the network reacts to changes according to the proposed clustering algorithm RECA. We remind that, in the update phase of RECA, we supposed that clusterheads broadcast periodically hello messages which are used by any node to update its neighborhood list. We distinguish two cases: ordinary nodes and clusterheads.

For the first case, after computing its new neighborhood list, an ordinary node executes first the RECA update algorithm. If it detaches from its clusterhead C_{old} and becomes attached to another one C_{new}, it obtains a new certificate from C_{new}. If it becomes a clusterhead for the first time, it must obtain a valid certificate and a partial key by requesting a coalition of K other already initialized clusterheads.

For the second case, after executing the RECA update algorithm, if a clusterhead becomes an ordinary node attached to a clusterhead C, it obtains a new certificate from C. C must inform other existing clusterheads to update their certificate in its list.

4. Evaluation

In this section, we evaluate the performances of the proposed hybrid PKI denoted PKI-H. We also compare its performances to those of three solutions proposed in the literature that use similar certification protocols: the partially distributed PKI [5] denoted PKI-P, the totally distributed solution [1] denoted PKI-T, and the cluster based distributed solution [16] denoted PKI-C. We have developed a simulation prototype for each architecture using NS2 and we used RECA as a clustering algorithm for PKI-C as it does not specify any clustering algorithm. Simulations are conducted under a disturbed environment: bit errors, packet losses, and MAC collisions. We considered the two following performance criteria:

Table II. Simulation parameters.

Parameter	Value
Simulation area	$1000 \times 1000\,\mathrm{m}^2$
Number of nodes	$100 \rightarrow 300$
Transmission range	$50\,\mathrm{m} \rightarrow 200\,\mathrm{m}$
Mobility model	Random Waypoint
Pause time	10 s
Maximum speed	$0 \rightarrow 10\,\mathrm{m/s}$
K	$2 \rightarrow 20$
Liars proportion	$10\% \rightarrow 50\%$
Trust threshold	50
Simulation time	600 s
Certificate validity period	between 100 and 200 s
number of warranties for PKI-C	3

1. The certification success ratio which is computed as the number of certification replies divided by the number of certification requests. This criterion will inform us about the availability of certification services.
2. The certification delay which is the period of time between a certification request and the receipt of a valid response. This criterion will give us an idea about the response time of simulated architectures.

These two parameters are studied by varying nodes number, transmission range and speed as well as the parameter K. Simulation parameters are reported in Table II.

Using the considered simulation parameters, the hybrid PKI generates around 16 clusters. This is why we considered 16 server nodes for PKI-P.

4.1. Simulation Results for Varying Transmission Range

In this section, we study the impact of nodes transmission range on both certification success ratio and delay. We considered 100 nodes moving with a maximum speed of 2 m/s. We fixed K to 4. This value is chosen so that certification failures will not be caused by the unavailability of K clusterheads for the hybrid PKI (16 clusterheads) or K servers for the PKI-P (16 servers). However, it might be caused by the unavailability of K one hop neighbor servers in the case of PKI-T.

Figure 7(a)and (b) show a comparison of certification success ratios and delays, respectively of the four architectures while varying nodes transmission range. We notice the following:

1. Despite the existence of malicious nodes, The Hybrid PKI gives better success ratios and delays

DOI: 10.1002/sec

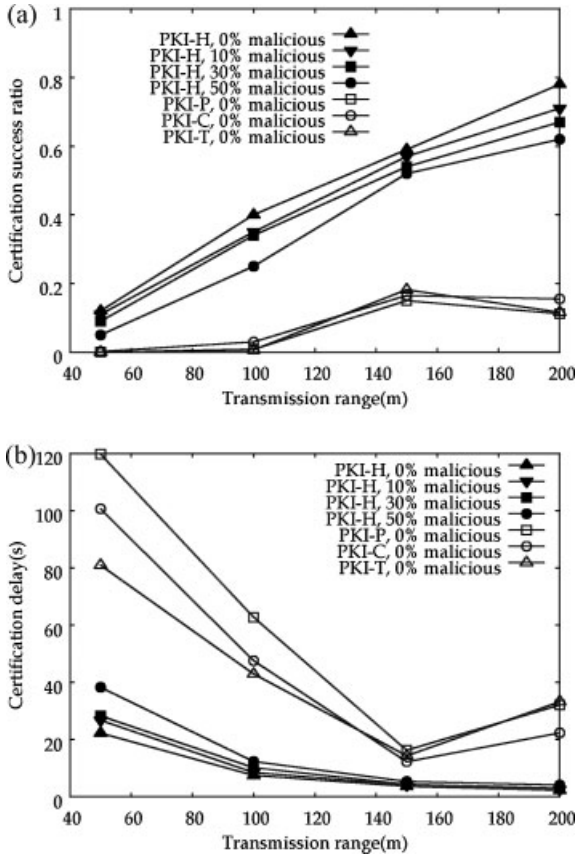

Fig. 7. Impact of transmission range on certification success ratio and delay, $K = 4$, $maxspeed = 2$ m/s. (a) Comparison of certification success ratios. (b) Comparison of certification delays.

than the other solutions. In fact, in the hybrid PKI, certification requests and corresponding responses are unicast while in the other architectures, requests are generally broadcasted, leading to more MAC collisions. In addition, the majority of these messages (ordinary nodes messages) are one hop messages since a node has a high probability to have a one hop clusterhead. This allows to minimize collisions, messages lost, and congestions. Moreover, in the hybrid PKI, ordinary nodes certification is centralized and the node has to contact only its clusterhead that RECA guarantee its presence. In the other solutions, even in the cluster based solution PKI-C, certification is distributed and a node must contact K other servers whose presence is not guaranteed.

2. In the hybrid PKI, when the number of malicious nodes increase, the success ratio decreases and delays increase. This is expected as a result since we have supposed that malicious nodes always send erroneous responses to certification requests. This causes certification failures and obliges nodes to launch other certification requests which increases the time needed to obtain a valid response.
3. In the hybrid PKI, success ratios increase with the increase of transmission range. In fact, the probability that a node leaves its clusterhead decreases with the increase of transmission range. Thus, it can receive a response to its certification request. This also explains that the delays decrease with the increase in transmission range.
4. In the PKI-C, PKI-P, and PKI-T, the success ratio increases with the increase of transmission range. It reaches a pick from which it begins to decrease. This can be explained by the following: for low transmission ranges, a node in the PKI-T can have less than K one hop servers and a node in either PKI-C or PKI-P find it difficult to join K nodes (unavailability of routes). When nodes transmission range increases, they have more chance to contact K servers. However, their messages are more exposed to collisions. That is why the success ratio decreases for high transmission ranges. We observe the reverse phenomenon for delays which can be explained by the same reasons we just have presented.
5. The PKI-C and PKI-P give almost the same results because they are partially distributed PKIs. However, PKI-C results are slightly better than those of PKI-P because PKI-C uses adequate and well placed servers (clusterheads supplied by RECA) as partial *CA* whereas PKI-P selects, randomly, a set of nodes as partial *CAs*. These nodes are not necessary the best ones (in terms of placement, robustness, and availability) to ensure *CA* services.

4.2. Simulation Results for Varying Mobility

In this section, we study the node speed impact both certification success ratio and delay. We considered 100 nodes with a transmission range of 150 m where the PKI-T, PKI-C, and PKI-P give their best results (see Figure 7). K was fixed to 4 for the same reasons given in the beginning of the previous section.

Figure 8 (a) and (b) show a comparison of certification success ratios and delays, respectively for the four architectures while varying nodes maximum speed. We notice that:

1. Despite mobility and time consuming RECA updates, the hybrid PKI has roughly better success ratios and delays than the three other solutions. In

DOI: 10.1002/sec

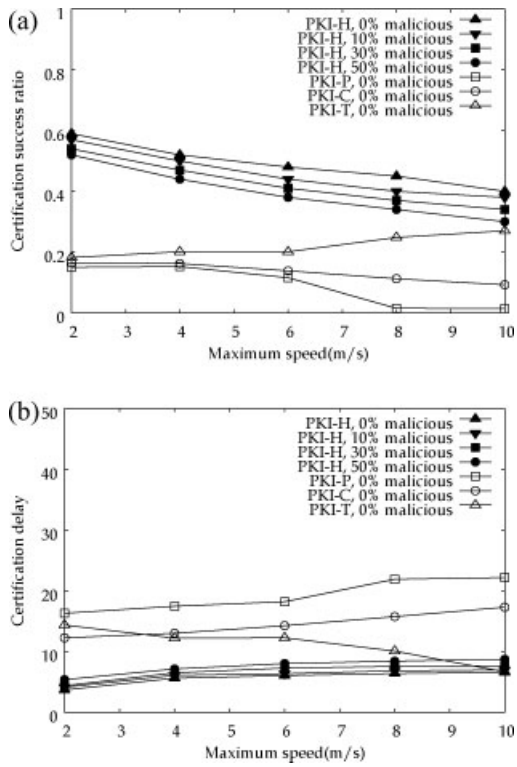

Fig. 8. Impact of mobility on certification success ratio and delay, $K = 4$, $Tx = 150$ m. (a) Comparison of certification success ratios. (b) Comparison of certification delays.

fact, the hybrid PKI minimizes messages used for certification services leading to a decrease in both collisions and delays.

2. In the hybrid PKI, nodes mobility will generate clusters updates that will affect delays and success ratios. This is why, as shown in Figure 8(a), success ratios decrease and delays increase, with the increase of mobility; an ordinary node can send a request to its clusterhead but do not receive a response because its clusterhead is no longer reachable or has changed its state. Moreover, when a node detaches from its clusterhead, it must await till it attaches itself to an existing clusterhead. Then it launches its certification request. This increases the certification delay
3. In the PKI-T, we observe that nodes have more chance to be served while increasing their speed. This is coherent because, with high mobility, nodes has more chance to reach K one hop neighbors.
4. Unlike PKI-T, routing has an important impact on both PKI-C and PKI-P performances. Thus, when mobility is high, responses can be lost due to the unavailability of routes. Mobility has then a negative effect on certification performances. However, we observe that PKI-C resists to increasing mobility better than PKI-P because it exploits adequate nodes (RECA clusterheads that are well placed in the network) as partial CAs.

4.3. Simulation Results for Varying Densities

In this section, we study the impact of density on both certification success ratio and delay. We considered a variable number of nodes with a transmission range of 150 m moving at maximum speed of 2 m/s. K was fixed to 4.

Figure 9 (a) and (b) show a comparison of certification success ratios and delays, respectively of the four architectures, with respect to density. We observe that:

1. The hybrid PKI has stable success ratios and delays unlike the other architectures because they are more exposed to collisions for high densities. This causes the loss of certification requests and responses.
2. The PKI-T has better certification delays than PKI-H for high density because a node has a high probability to find K neighbors. However, this results are

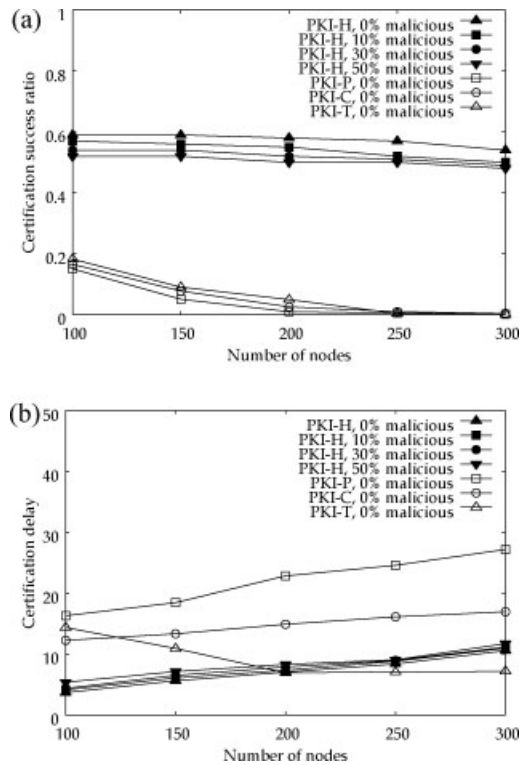

Fig. 9. Impact of density on certification success ratio and delay, $K = 4$, $Tx = 150$ m. (a) Comparison of certification success ratios. (b) Comparaison of certification delays.

DOI: 10.1002/sec

related to a very low certification success ratio as shown in Figure 9(a) because of collisions.

3. PKI-C and PKI-P performances are more affected with the increase of nodes mobility. However, PKI-C resists to increasing nodes mobility better than PKI-P. This is because PKI-C servers are well placed in the network.

4.4. Simulation Results for Varying *K*

In this section, we report simulation results while varying *K*. We considered 100 nodes with transmission range 150 m and moving with a maximum speed of 2 m/s. We remind that with these parameters, the hybrid PKI and PKI-C give around 16 clusters each, while the PKI-P is configured with 16 servers.

Figure 10 (a) and (b) show a comparison of certification success ratios and delays, respectively for the four architectures with respect to *K*. We notice that, despite the existence of malicious nodes, the hybrid PKI has better results than the other architectures because it ensures better certification services availability; a node contacts only its clusterhead to obtain a service and clusterheads are roughly at one hop to their members (see Figure 3). We also notice that for all architectures, the increase of *K* results in an increase of certification delays and a decrease in certification success ratios. However, the performances of the hybrid PKI are less sensitive to the increase of *K* than the three other architectures that stop working when *K* becomes greater than 8. In fact, in the hybrid PKI, only clusterheads are affected by the increase of *K*. Moreover, the overhead caused by certification services in PKI-T, PKI-C, and PKI-P increase rapidly with the increase of *K* since any node must be served in a distributed manner. Consequently, there are more collisions causing certification failures.

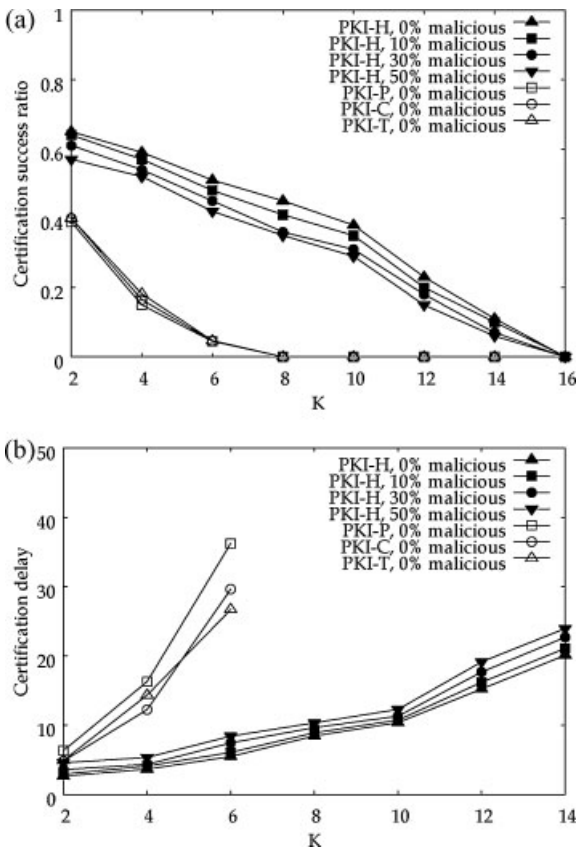

Fig. 10. Impact of *K* on certification success ratios and delays, *Maxspeed* = 2m/s, *Tx* = 150 m. (a) Comparison of certification success ratios. (b) Comparison of certification delays.

5. Conclusions and Future Works

In this paper, we have presented a hybrid public key architecture that is adapted for *ad hoc* networks. The proposed solution permits to distribute CA services only over trustworthy nodes, unlike solutions presented in the literature.

In our solution, we distribute CA services over trustworthy nodes that are elected by RECA. The proposed PKI-H considers each clusterhead as a CA for its members and implements a totally distributed architecture over clusterheads. Clusterheads behavior is supervised using a distributed reputation system. This system permits to detect and eliminate misbehaving clusterheads leading the system to a state where only trustworthy clusterheads manage the network security. Thus avoiding active attacks.

Implementing the hybrid PKI-H on RECA will enhance scalability and certification services availability such as certificate creation, revocation and renewal. Through the simulations done, certification success ratios and delays are enhanced in comparison with distributed PKI presented in the literature: unlike PKI-T, PKI-C, and PKI-P where each node cannot be served unless the participation of at least *K* nodes, in our solution only clusterheads are served, each, by *K* clusterheads while ordinary nodes are served, each, by its corresponding clusterhead. Thus, the overhead is minimized and collisions are reduced which enhances availability and scalability.

As future works, we plan to enhance the reputation system and test our architecture in a real environment.

DOI: 10.1002/sec

Appendix : Feldman Verifiable Secret Sharing

Felman (K, N) verifiable secret sharing [23] allows a node P_0, known as the dealer, to share a secret $S \in Z_q$ over N nodes $P_1, P_2, \ldots P_N$ so that an arbitrary coalition of K nodes among them can reconstruct the shared secret ($K < N$).

Feldman's protocol is verifiable: It allows each node P_i to verify the validity of the share S_i it receives from P_0. It allows also to detect and filter erroneous partial secrets while reconstructing $S = f(0)$.

Feldman verifiable secret sharing is executed according to the following steps:

1. P_0 selects two large prime numbers p and q so that q divides $p - 1$
2. It selects also a random polynomial function $f(x) = \sum_{t=0}^{K} a_t x^t$ over Z_q, of degree K such that $f(0) = S$.
3. It secretly sends to each node P_i of identity id_i its share $S_i = f(\mathrm{id}_i) \bmod q$. In this work S_i will be encrypted with the public key of P_i.
4. It broadcasts verification values $A_t = g^{a_t} \bmod p$, $t = 0, 1, 2 \ldots K$. This allows any node P_i of identity id_i to verify the correctness of the partial share S_i it receives by using the Equation (1) where g is a generator of Z_q.

$$g^{S_i} = \prod_{t=0}^{K} (A_t)^{\mathrm{id}_i^t} \bmod \mathrm{p} \qquad (1)$$

5. If a node P_i detects that its share S_i is invalid, it diffuses complaint against P_0
6. P_0 reveals the correct share S_i (that verifies Equation (1)) for each complaining node P_i.
7. If a revealed portion does not verify Equation (1), P_0 is disqualified.
8. The same Equation (1) is used to validate partial secrets while reconstructing S.

We notice that the network secret S cannot be computed knowing g and $A_0 = g^{a_0} = g^S$ which are publicly revealed (discrete log assumption [27]). An attacker that determines less than K partial secrets cannot deduce any supplementary information about S over what can be deduced from g^S [23].

References

1. Kong J, Zerfos P, Luo H, Lu S, Zhang L. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *Proceedings of the 9th International Conference on Network Protocols*, 2001; 251–260.
2. Luo H, Zerfos P, Kong J, Lu S, Zhang L. Self-securing ad hoc wireless networks. In *Proceedings of 7th International Symposium on Computers and Communications (ISCC'02)*, 2002.
3. Khalili A, Katz J, Arbaugh W. Toward secure key distribution in truly ad hoc networks. In *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, 2003, 342–346.
4. Deng H, Mukherjee A, Agrawal DP. Threshold and Identity-based Key Management and Authentication for Wireless Ad Hoc Networks, 2004; doi: ieeecomputersociety.org/10.1109/ITCC.2004.1286434
5. Zhou L, Haas ZJ. Securing ad hoc networks. *IEEE Network Magazine* 1999; **13**(6): 24–30.
6. Yi S, Kravets R. Practical PKI for Ad Hoc Wireless Networks, Department of Computer Science, University of Illinois, Technical Report UIUCDCS-R-2002-2273, UILU-ENG-2002-1717, 2001.
7. Yi S, Kravets R. Key Management for Heterogeneous Ad Hoc Wireless Networks, Department of Computer Science, University of Illinois, Technical Report UIUCDCS-R-2002-2290, UILUENG-2002-1734, 2002.
8. Yi S, Kravets R. Key management for heterogeneous ad hoc wireless networks. In *Proceedings of 10th IEEE International Conference on Network Protocols (ICNP'02)*, 2002; 12–15.
9. Yi S, Kravets R. MOCA: Mobile certificate authority for wireless ad hoc networks. In 2nd Annual PKI Research Workshop (PKI03), 2003.
10. Xu G, Iftode L. Locality driven key management architecture for mobile ad-hoc networks. In *Proceedings of First IEEE International Conference on Mobile and Sensor Networks (MASS'04)*, 2004; 24–27.
11. Wu B, Wu J, Fernandez EB, Magliveras S. Secure and efficient key management in mobile ad hoc networks. In *Proceedings of First International Workshop on Systems and Network Security (SNS2005) (in conjunction with IPDPS)*, 2005.
12. Wu B, Wu J, Fernandez EB, Ilyas M, Magliveras SB. Secure and efficient key management in mobile ad hoc networks. *Journal of Network and Computer Applications* 2005; **30**: 937–954.
13. Zhu B, Bao F, Deng RH, Kankanhalli MS, Wang G. Efficient and robust key management for large mobile ad hoc networks. *Computer Networks* 2005; **48**: 657–682.
14. Joshi D, Namuduri K, Pendse R. Secure, redundant, and fully distributed key management scheme for mobile ad hoc networks: an analysis. *EURASIP Journal on Wireless Communications and Networking* 2005; **5**(4): 579–589.
15. Lee HK, Choi J. Multistage Authentication Scheme for Mobile Ad-Hoc Network Using Clustering Mechanism, HPCC, LNCS 4208, 2006; 653–661.
16. Bechler M, Hof HJ, Kraft D, Pählke F, Wolf L. A cluster based security architecture for ad hoc networks, IEEE INFOCOM, 2004.
17. Fu Y, He J, Li G. A Composite Key Management Scheme for Mobile Ad Hoc Networks, OTM Workshops, LNCS 4277, 2006; 575–584.
18. Shamir A. How to share a secret. *Communications of the ACM* 1979; **22**(11): 612–613.
19. Li G, Han W. A New Scheme for Key Management in Ad Hoc Networks, ICN 2005, LNCS 3421, 2005; 242–249.
20. Pedersen T. A threshold cryptosystem without a trusted party. In *Proccedings of Eurocrypt'91, LNCS N547*, 1991; 522–526.
21. Chiang C, Wu H, Liu W, Gerla M. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings of IEEE Singapore International Conference on Networks*, 1997.
22. Elhdhili MH, Benazzouz L, Kamoun F. Reputation based clustering clgorithm for security management in ad hoc networks with

DOI: 10.1002/sec

liars, To apperar in proceedings of the 3rd International Conference on Risks and Security of Internet and Systems CRiSIS, 2008.
23. Feldman P. A practical scheme for non interactive verifiable secret sharing. In 28th Annual Symposium on Foundations of Computer Science, 1987; 427–438.
24. Pedersen T. Non interactive and information-theoretic secure verifiable secret sharing. In *Proccedings of Crypto'91, LNCS N576*, 1991; 192–140.
25. Haiyun L, Songwu L. Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks, *Rapport Technique*, 2000.
26. Elhdhili MH, Benazzouz L, Kamoun F. CASAN: clustering algorithm for security in ad hoc networks, 2008. doi: 10.1016/j.comcom.2008.04.001
27. Gennaro R, Jarecki S, Krawczyk H, Rabin T. Secure distributed key generation for discrete-log based cryptosystems, 2003.

DOI: 10.1002/sec