Lecture Notes in Computer Science (LNCS), Vol. 3093, Proc. 1st European PKI Workshop: Research and Applications, pp. 232-245, Samos Island, Greece, June 2004.

Distributed CA-based PKI for Mobile Ad hoc Networks using Elliptic Curve Cryptography

Charikleia Zouridaki[1], Brian L. Mark[1], Kris Gaj[1], Roshan K. Thomas[2]

[1] George Mason University, Electrical and Computer Engineering Dept., 4400 University Drive, Fairfax, VA, 22030, USA
{czourida, bmark, kgaj}@gmu.edu
[2] McAfee Research, Network Associates,Inc., 1145 Herndon Parkway, Suite 500, Herndon, VA, 20170, USA
RThomas@nai.com

Abstract. The implementation of a standard PKI in a mobile ad hoc network (MANET) is not practical for several reasons: (1) lack of a fixed infrastructure; (2) a centralized certification authority (CA) represents a single point of failure in the network; (3) the relative locations and logical assignments of nodes vary in time; (4) nodes often have limited transmission and computational power, storage, and battery life. We propose a practical distributed CA-based PKI scheme for MANETs based on Elliptic Curve Cryptography (ECC) that overcomes these challenges. In this scheme, a relatively small number of mobile CA servers provide distributed service for the mobile nodes. The key elements of our approach include the use of threshold cryptography, cluster-based key management with mobile CA servers, and ECC. We show that the proposed scheme is resistant to a wide range of security attacks and can scale easily to networks of large size.

Keywords: mobile ad hoc network, threshold cryptography, elliptic curve cryptography, cluster, scalability

1 Introduction

Providing security in MANETs is an inherently challenging problem due to the lack of a fixed infrastructure, the dynamically changing network topology, the limitations of the wireless channel, and the limited capabilities of the nodes. Since the nodes are mobile, they are particularly vulnerable to physical attacks from within and outside the network. The nodes are typically of small size and have limited computational, storage, and transmission power, as well as limited battery life. Such limitations place severe constraints on security architectures for MANETs. MANETs cannot always guarantee online access to a centralized CA due to the often intermittent and unreliable nature of the wireless channel. Thus, the use of a standard public key infrastructure (PKI) is generally infeasible in an ad hoc wireless environment.

The goal of securing ad hoc wireless networks has generated much interest in the research community in recent years. Password-based schemes for key establishment [1], [2], [3], [4], [5] avoid the need for a CA by carrying out authentication on the basis of a shared secret or password established prior to the deployment of the network. A security scheme similar to Pretty Good Privacy (PGP) has been proposed for MANETs [6], [7], whereby certificates are issued by users based on the establishment of chains of trust. This approach is well-suited to the ad hoc networking environment, but provides only probabilistic security guarantees and relies on transitive trust relationships, which may not be sufficient for some applications. Another approach to securing MANETs is based on a distributed certification authority (DCA) [8], [9] to avoid the problem of a single point of failure found in traditional PKI architectures. This approach provides deterministic security guarantees, but raises critical issues of scalability in practical MANETs. The concept of a distributed certification authority has also been applied to wired networks in a security architecture called COCA [10], which also provides fault tolerance. However, COCA cannot be directly applied to ad hoc networking environments where the behavior of the nodes is complicated by dynamic changes in wireless connectivity. In general, PKI architectures designed with wired networks in mind cannot be carried over straightforwardly to ad hoc networks [8].

We propose a comprehensive approach to providing a distributed CA-based PKI in MANETs, which potentially allows the network size to scale to hundreds or even thousands of nodes. The key elements of our approach are: (1) a scheme for dynamically partitioning the network into smaller clusters of nodes based on nodal mobility; (2) a distributed certification authority with multiple CA servers employing threshold-based cryptography with proactive share recovery, and replicated key repositories assigned to each cluster; (3) the use of elliptic curve cryptography (ECC). Distributing CA servers geographically over the coverage area of the MANET makes it more difficult for an adversary to compromise multiple CA servers simultaneously. Further, the use of threshold-based cryptography with proactive share recovery forces an adversary to simultaneously compromise more than half of the CA servers before the distributed CA itself is compromised. The distribution of the key management architecture over clusters reduces the storage requirements of the nodes and the CA servers, as well as the computational and signaling overhead. Finally, the use of elliptic curve cryptography dramatically reduces the computations involved in cryptographic operations, making the PKI-based scheme feasible even for nodes of modest computational power.

The main contributions of the paper are a practical architecture for implementing a distributed CA over a MANET via dynamic clustering and a detailed study of the computational gains achievable by using elliptic curve cryptography in the MANET setting. The proposed architecture provides a highly secure PKI with a flat trust management architecture and deterministic security guarantees. We discuss the advantages of the proposed architecture compared to existing schemes and how the architecture can resist a wide range of security attacks.

The remainder of the paper is organized as follows. Section 2 discusses the elements of the proposed distributed CA-based PKI architecture. Section 3 focuses on the operational aspects of the cluster-based key management protocols within the proposed PKI architecture. Section 4 provides a detailed analysis of the performance gains achieved by using ECC in the MANET setting. Finally, the paper is concluded in Section 5.

2 Distributed CA-based PKI Architecture

2.1 Overview

Figure 1 gives a three-tiered logical view of how the DCA architecture is organized. At the lowest tier individual nodes are organized into clusters using standard clustering schemes [11], [12], [13], [14]. The next tier consists of one or more certificate repositories in each cluster. The top tier consists of DCA servers. Although the servers and repositories are represented at higher levels from the other nodes, the proposed scheme is in fact a flat PKI trust hierarchy. We next discuss the details of how these logical tiers are organized.

The number of DCA servers should be a function of the network size and the degree of resilience required against attacks. We utilize a threshold based scheme to govern this. The number of servers is defined by n = 2k+1, where k is the maximum number of servers that can be compromised in a predefined period of time. Each server participates in issuing certificates and revocation certificates (counter-certificates) and in periodically signing the certificate revocation list (CRL) that contains the serial numbers of revoked certificates from the entire network. The servers are assumed to be physically more secure and computationally more powerful nodes. The initial distribution of the CA servers is described in section 3.1. If a server is compromised but undetected, it is because it functions properly; in this case no measures are necessary. Once a server is compromised and detected, it cannot perform service as part of the DCA, until its share is recovered or renewed. It does not influence or affect the functioning of the cluster or the system.

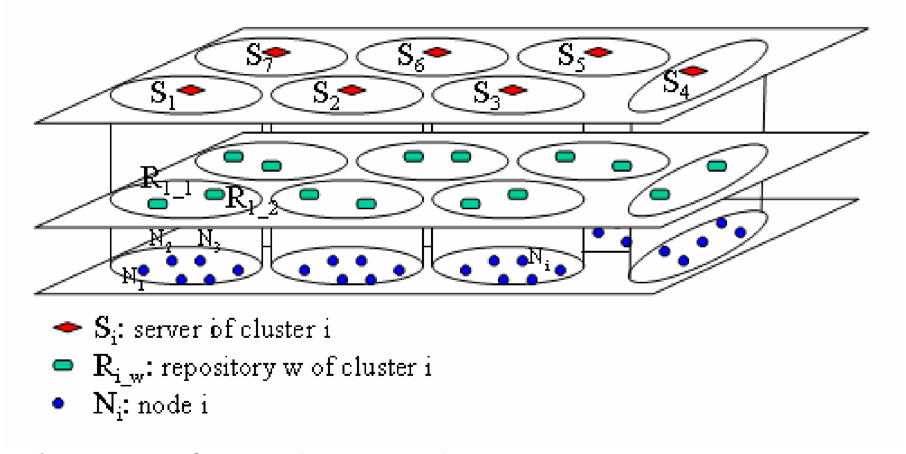

Fig. 1. DCA architecture (n=7, t – r = 2)

Table 1. Certificates available to current and future participants of the system

Available to:	Cluster Server	Other Server	Cluster Nodes	Non-Cluster Nodes	New Nodes
Certificate	X		Optional		Optional
Counter-Certificate	X	X	X	X	X
Most recent CRL	X	X	X	X	X

Within each cluster, a fixed number t of nodes are designated as repositories that store the certificates of the nodes within the cluster, the certificates of all servers, the counter-certificates of the network nodes, and the most recent version of the CRL. The repositories might also become compromised and thus become unavailable (note that in case of compromised repositories nothing that was not already public is revealed). However, up to r repositories may be compromised within a cluster before a new node within a cluster is elected to serve as a repository. This way, there will always be a minimum of t - r active repositories in each cluster.

The use of clustering has two advantages. First, it reduces the storage requirements of individual nodes as each node needs to store at most the certificates of the other nodes in the same cluster rather than the entire network. Second, clustering reduces the communication overhead and increases the efficiency of certificate management as certificates are always available to each node at a local repository within a small number of hops.

A key feature is that the CRL maintained in the repositories is timestamped, signed by the DCA, and updated every day. The corruption of the repositories is acceptable, since (1) corrupted certificates or counter-certificates will be detected via signature verification, (2) corruption that involves deletion of certificates and/or counter-certificates, is also detected, as the most recent CRL can be checked. If the information provided is up-to-date, it is considered correct. If it is not, another repository is accessed. Hence, the existence of t - r active repositories in each cluster ensures that the cluster's operation is never interrupted.

2.2 Threat Model and Resistance to Attacks

Our focus is on the compromise of DCA servers and certificate repositories and the effects they will have on the integrity and availability of certificate management services. We consider the following attack model to characterize the resiliency of our schemes. We denote an attack type by a triple (c, s, r), where c is the number of clusters involved, and s and r represent the number of DCA servers and repository copies, respectively, that can be compromised by an attack.

The first attack we consider is the well-known theft of the secret key of the CA. This represents a single point of failure in a CA infrastructure and allows an attacker to forge certificates. Our distributed implementation of the CA uses a threshold cryptography scheme to protect the secret key of the CA by distributing the secret (private key) among a number of servers (shares) and thus avoids the single point of failure vulnerability. This scheme can thus be used to deal with attacks of the form (n, k, r) where k is the maximum number of servers compromised out of a total number of n servers with $n = 2k+1$, and there is at most one DCA server per cluster. It can also deal with attacks of the form (p, k, r) where we have a total of n servers with $n = 2k+1$ and $p < n$. In this case, some clusters may have more than one server.

We next consider attacks that reduce the availability of certificate repositories. Our cluster-based distributed CA scheme can handle attacks ranging from (1, 0, r) to (1, s, r) by maintaining at least t replicas (where $t > r$) of the certificate repository within each cluster. The number $b = t - r$, is tunable so as to always guarantee b copies of the repository. In the case of (1, s, r), all s servers in a cluster are compromised and unavailable but as long as one replica of the repository is functioning we can provide all the necessary certificate services including the addition of a new member and the distribution of its certificate.

Another source of vulnerability in a CA system has to do with the authentication and validation of requests to issue certificates. In our scheme, this function is performed by a Registration Authority (RA). A common vulnerability is that the RA can be fooled into believing B when B impersonates A. We assume that the RA (1) does not belong to the MANET but is part of a wired network; (2) can communicate with the servers of the DCA securely; (3) does not know the private key of the DCA. The RA will verify the credentials of a node and if satisfied, contact at least k+1 servers and request issuance of a certificate. This model reflects the practical procedures and work scenarios present in many environments that use wireless networking. For example, before troops go out to the battlefield with their wireless devices, they will be required to report to and register at the RA. Upon successful verification of credentials, a wireless device with a private key and an associated certificate with the public key may be issued to a soldier to enable further communications in the battlefield.

Passive attacks such as eavesdropping of wireless communications are not a concern for us as any information gathered from such activities is public knowledge. Finally, the integrity of certificates through active tampering can be verified through the use of digital signatures.

2.3 Elliptic Curve-based Distributed Certification Authority

We propose a distributed certification authority based on threshold cryptography and proactive secret sharing [15], [16]. The traditional public key cryptosystem employed in [15], [16] is impractical for MANETs, as it imposes high computational and communication overhead. Therefore, we propose the use of ECC [17] to reduce this overhead for the mobile devices. The DCA consists of $n = 2k+1$ servers that will share a secret value x, through a $(k+1, n)$ threshold scheme [18]. Any $(k + 1)$ servers are required to combine their shares in order to sign a message, while the adversary who wants to learn or destroy its secret signature key has to compromise more than k servers during a single time period. A broadcast communication channel is assumed. The goal is to prevent the adversary from learning or destroying the secret x.

Time Periods and Update Phases. We assume that time is divided into time periods, during which the servers can perform the group signature operation. Each period is determined by a common global clock and its duration is specified as required (five days, two weeks, etc.). There are short update phases for the re-randomization of the original key. Previous shares become useless and should be erased, since combined information about two periods of the system could enable the adversary to break the system.

Each update phase consists of: 1) private key renewal, which renews the means of encryption and authentication among the servers; 2) lost share detection and recovery, which checks the current shares of the servers and reconstructs the corrupted shares, if any; and 3) share renewal, which re-randomizes the secret key x. The resolving accusation protocol [15] is implemented when two servers' claims are contradictory at specific phases of the protocols. This protocol might be completed in 3 or 4 steps, depending on the state of the transactions.

Cryptographic Tools. We assume that the signature scheme used by the DCA is Elliptic Curve El Gamal signatures, which is discussed in more detail in section 4. We assume that Feldman's verifiable secret sharing (VSS) scheme [19] is used for the sharing of the CA's key among the n participating servers. Each server has two pairs of keys, one for encryption and one for signature that will provide both private and authenticated communication. Table 2 shows the keys and parameters of the CA that are known to different elements of the system.

3 Operational Aspects of the Proposed Architecture

The notation used in this section is shown in Table 3.

3.1 Network Initialization

The proactive secret sharing system that we adopt for our DCA assumes an initialization phase, during which (1) the secret key of the CA is generated and shared between the servers, (2) each server generates its pair of authentication and encryption keys and publishes their public values to the group of servers, (3) an initial set of nodes is deployed. For simplicity we shall assume that the nodes form clusters, with one CA server per cluster. After the initialization phase, nodes may migrate between clusters and new nodes may join or leave the network.

3.2 Activating a Node

In the proposed scheme when a new N node wants to join the mobile network, it must carry out the following steps.

Table 2. Parameters of the DCA available to current and new participants of the system

Available to:	Server i	All Servers	Current Nodes	New Nodes
System parameters	X	X	X	X
Public key of DCA Y	X	X	X	X
Partial verification keys	X	X		
Share x_i of the secret x	X			
Server own signature key	X			
Server signature verification key	X	X	X	X
Server encryption key	X	X	X	X
Server decryption key	X			

A1. Node N obtains certificate c_N

A1.1. N contacts RA

A1.2. RA verifies credentials of N and securely contacts k+1 CA servers

A1.3. (k+1) CA servers issue c_N for N and send it to RA

A1.4. RA gives c_N and c_{DCA} to N

A2. Node N joins cluster i

A2.1. N sends c_N to R_{i_w}

A2.2. N requests C_i , CC, CRL from R_{i_w}

A2.3. R_{i_w} broadcasts c_N

A2.4. R_i store c_N

A2.5. j_i optionally store c_N

In step A1, the node that wants to enter the network contacts a fixed Registration Authority. The RA will verify the credentials and if satisfied, contact at least k+1 servers and request issuance of a certificate. Then, k+1 servers will perform the distributed signature operation protocol to issue a certificate for the new member. The issued certificate will be sent to the RA, the new member will obtain it and join the network. The new member can join any cluster of the network since its certificate's signature can be verified by any server or node (the public key of the DCA is stored by all network participants).

Table 3. Notation used in this section

DCA	set of CA servers	S_i	set of servers of K_i
K	number of clusters	C_i	set of certificates of nodes in K_i
R	number of repositories	R_i	set of repositories in K_i
RA	Registration Authority	c_{i_w}	certificate of j_{i_w}
CC	set of counter-certificates	cc_{i_w}	counter-certificate of j_{i_w}
c_{DCA}	certificate of DCA	S_{i_w}	server w of K_i
J	number of nodes	R_{i_w}	repository w in K_i
j_i	number of nodes in K_i	j_{i_w}	node w in K_i
K_i	cluster i	N	a new node

The introduction of a newly certified node in a cluster is described in step A2. The node has to contact one of the repositories of the cluster it wishes to join in order to publicize its certificate, which will be broadcast to the current nodes and repositories of the cluster. The node can obtain the certificate of any node in the cluster by contacting a local repository. Usually the certificates that are most likely to be used should be cached. The local storage of certificates makes communication within a cluster efficient, even when encryption is needed. Moreover, the repository will provide the new node with the most recent version of the certificate revocation list (CRL) containing serial numbers of revoked certificates from the entire network signed by the DCA and the counter-certificates issued since the last update of the CRL.

3.3 Deactivating a Node

The certificate revocation process takes place as described by the following protocol.

```
D1. if m nodes of $K_i$ want $cc_{i\_w}$ to be issued
    D1.1.  m nodes of $K_i$ send signed accusations about $j_{i\_w}$ to at least k+1 CA servers
    D1.2.  k+1 CA servers issue $cc_{i\_w}$ and add the serial number of $cc_{i\_w}$ to the CRL
    D1.3.  $cc_{i\_w}$ is broadcast to the network
    D1.4.  $S_{i\_w}$, $R_{i\_w}$, $j_{i\_w}$ store $cc_{i\_w}$
    D1.5.  CRL is renewed and timestamped by the DCA periodically
    D1.6.  $S_{i\_w}$, $R_{i\_w}$, $j_{i\_w}$ store CRL
D2. if node $j_{i\_w}$ wants to request revocation of its own certificate
    D2.1.  $j_{i\_w}$ sends a signed request for the issuing of $cc_{i\_w}$ to at least k+1 CA servers
    D2.2.  follow steps D1.2 to D1.6
```

Revoking a certificate for a given node can be initiated either by m users belonging to the same cluster, where m can vary depending on the application of the network, or by a node that wants to revoke its own certificate. When the revocation process is initiated by m users requesting revocation of node's j_{i_w} certificate, their request needs to be sent to at least k+1 servers. As a result, the DCA issues a counter-certificate and adds a serial number of the revoked certificate to the global CRL. The revocation certificates are broadcast to all nodes of the network, immediately after being issued. The storage of the counter-certificates is obligatory. All the servers and repositories keep the CRL that contains serial numbers of revoked certificates from the entire network. CRLs are renewed by the DCA every day or more often if necessary and are broadcast to all nodes in the network.

The revocation process may also be initiated by a node that wishes to revoke its own certificate either because it wants to leave the mobile network, or because its private key has been compromised. In this case, the node sends a signed request to at least k+1 servers to enable the issuing of its revocation certificate.

3.4 Node Migrations across Clusters

A highly mobile node j_{i_w} might leave source cluster K_s and enter destination cluster K_d. The following protocol describes the protocol to manage smooth node migrations across clusters.

M1. Node j_{i_w} leaves source cluster K_s
- M1.1 if the mobility management protocol indicates that the node j_{i_w} permanently leaves the source cluster K_s, j_{i_w} deletes the certificates C_s of the nodes in K_s
- M1.2 else, go to step M2.

M2. Node j_{i_w} joins destination cluster K_d
- M2.1. j_{i_w} sends its certificate c_{i_w} to repository w of K_d (R_{d_w})
- M2.2. j_{i_w} requests the certificates C_d of the nodes in cluster K_d from R_{d_w}
- M2.3. R_{d_w} broadcasts c_{i_w}
- M2.4. the repositories of K_d (R_d) store c_{i_w}
- M2.5. the nodes of K_d (j_d) optionally store c_{i_w}

When node j_{i_w} leaves source cluster K_s and enters destination cluster K_d, it does not know the certificates of the nodes in cluster K_d. Therefore, it contacts any of the cluster K_d repositories (R_{d_w}) in order to obtain them. At the same time, node j_{i_w} is introduced to the cluster by sending its certificate to R_{d_w}. Besides that, node j_{i_w} can send its certificate to each node it corresponds with and the certificate can be authenticated using the public key of the DCA that each node in the network knows. The certificates of the nodes of the source cluster K_s that are stored in node j_{i_w} are deleted, unless the mobility management protocol predicts that the node is temporarily moved to a new cluster. In this case, the node can be programmed to delete the certificates of cluster K_s when it has moved to a third cluster K_h. Another option is to keep the stored certificates if enough storage space is available.

3.5 Intra-Cluster Communications

Communication inside a cluster is relatively fast, regardless of whether the communication is encrypted or authenticated. This is because each node caches (1) the most frequently used certificates of the nodes within the cluster, (2) the revoked certificates from the entire network and (3) the most recent version of the CRL. Consequently, the nodes infrequently request certificates or counter-certificates from the repositories, hence reducing the communication overhead. The cluster's CA server periodically informs the cluster about the new network counter-certificates when they are issued and the updated CRL. Repositories broadcast the certificates of new nodes.

3.6 Inter-Cluster Communications

The way inter-cluster communication takes place depends on whether it needs to be authenticated or encrypted. Since the public key of the DCA is known to all the participants of the system, the certificate of any node can be verified by any other node. Thus, the authentication has very low communication overhead. On the contrary, when an encrypted message needs to be sent, the sending node does not know the public key of the receiving node, because it has only cached certificates of the same cluster. Then, the required certificate has to be requested from one of the repositories of the cluster to which the receiving node belongs. The knowledge of the counter-certificates of the whole network and the most recent CRL is an advantage, since all the nodes are aware of all the revoked certificates and as a result a revoked certificate will never be requested. This reduces the communication overhead. The reply that is sent contains the requested certificate. However, it should be noted that node, whose certificate is requested, might reply to the request route and might send the certificate by itself.

3.7 Cluster splitting and merging

The network size and distribution of nodes may change dynamically, which affects the number of clusters as defined by the cluster management protocol [11], [12], [13], [14]. For the purposes of the proposed DCA-based PKI architecture, we shall assume the basic functions of a generic cluster management protocol, i.e., cluster splitting and merging operations.

If the network size increases and new clusters are formed, the number of network clusters may become larger than the number of DCA servers, since the number of DCA servers is fixed n = 2k+1. In this case, not all clusters will contain one CA server. Assume, for example, that cluster i splits into two clusters i_1 and i_2. The CA server of cluster i will become part of a cluster of higher density, for instance i_1. Hence, cluster i_2 will not contain a CA server, but its nodes will be mapped to the server of cluster i_1. The key issue is that the CA servers are distributed into the network. However, the cluster i_2 will need to elect t repositories. Any of the cluster nodes that have never been accused of misbehavior can serve as a repository.

If the network size decreases and some clusters are merged, the number of network clusters may become smaller than the number of DCA servers. In this case, a cluster that results from the merger of two clusters will contain the CA servers and repositories of the original clusters.

4 Performance Gains using ECC

Having presented the distributed CA-based PKI scheme, we now discuss the performance improvement of the scheme through the use Elliptic Curve cryptography. ECC is appropriate for mobile nodes with limited computational power because it requires smaller keys and involves operations on smaller integers than in standard systems.

4.1 Cryptographic Tools

A sufficiently large prime p, an Elliptic Curve E over GF(p) with a total number of points N and a generator of the group of points on the elliptic curve P are chosen. Let c be a certificate to be signed. The above settings are publicized. The private key can be any number x, where where $1 \leq x \leq$ #E (GF (p))-1. Here, #E (GF (p)) denotes the number of points on the curve. Its corresponding public key is a point Y = x*P.

The signature scheme used by the DCA is Elliptic Curve El Gamal signatures. For the elliptic curve-based system we choose to implement the Elliptic Curve Digital Signature Algorithm (ECDSA) signature scheme for signatures since it is a standard and the Elliptic Curve El Gamal encryption scheme for encryption. We have chosen a key size of 160 bits (i.e., p, q are 160 bits long for both the EC El Gamal and ECDSA), which is equivalent to a 1024 bit key in traditional public-key cryptosystems. A 160 bit key size provides a sufficient level of security for most applications, while placing a reasonable computational burden on the nodes of a MANET.

4.2 Computational and Time Requirements of the DCA

A detailed analysis of the computational and communication overhead for each protocol of the elliptic curve-based DCA scheme and the traditional public key system employed in [15], [16] is given in [20]. The overhead depends on the values of the variables k, β, m, where k = (n-1)/2 and n the number of servers, β the number of servers of faulty shares that are recovered (during the recovery of lost shares protocol) and m the number of servers that misbehaved during the execution of the share renewal protocol. The formulas derived in [20] show that the number of computations (of modular multiplications) to be performed and the traffic generated is essentially proportional to k^2.

To illustrate the practicality of the elliptic curve-based DCA scheme on a large network, we shall consider a network of 51 servers (k = 25). We choose β and m to be equal to 2. We shall assume that each cluster contains approximately 100 nodes. As a result, the network contains about 5100 nodes and 51 servers.

To evaluate the elliptic curve-based system, it is crucial to compare it with the original, traditional public key system. Let EC-DCA denote our elliptic curve-based DCA scheme and T-DCA denote the original, traditional public key system. For the EC-DCA system, the results were derived as the number of modular multiplications with a modulus of 160 bits. For T-DCA, the results were derived in terms of the number of modular multiplications with a modulus 1024 bits. However, the time required to perform a modular multiplication with modulus of the size of z bits in software is proportional to the z^2. Thus, modular multiplications with modulus of the size of 160 bits can be normalized to modular multiplications for a 1024 bit modulus, to simplify the comparison.

Table 4 presents the comparison of the computations required by each server for each protocol of the scheme when n = 51. Here, the unit of computation is one modular multiplication, where the modulus is determined by the key size. The various types of computation involved in the cryptosystem are presented in terms of the number of equivalent modular multiplications required. We note that each server does not perform the same number of computations. Since not all of the servers contribute to the distributed signature operation, not all the servers need to recover their shares. Also, some servers may not participate if they are accused of cheating. Thus, we shall consider the maximum number of computations that have to be performed per server. Even though not all the servers will need to perform the maximum number of computations, the time needed to finish running a given protocol is determined by the servers that perform the most computations.

The number of computations that each device has to perform can be translated into the time that the device needs to perform those computations, taking into account the computational power of the present mobile devices. At present, smart cards can perform up to 3000 modular multiplications per second with the size of the modulus being 1024 [21]. Based on that, we can calculate how much time a server needs to perform the computations needed for each protocol. Table 4 presents a comparison of the maximum number of computations per server and the time required for the following protocols: (1) Distributed Signature Operation, (2) Private key renewal, (3) Lost share detection, (4) Recovery of lost shares, (5) Share renewal, (6) Resolving accusations - in 3 steps -, (7) Resolving accusations - in 4 steps -.

4.3 Communication and Storage Requirements

The communication overhead imposed by the proposed scheme is shown in Table 5. As the majority of involve messages that are broadcast, the number of messages sent differs from the number of messages received and thus processed within the DCA. The transmission requirements are manageable for 51 servers. The values of the variables involved are discussed in Section 4.2.

The key space is partitioned with the use of clusters and the use of small key sizes, since the system is elliptic curve-based. We calculate the key storage requirements for a network of 51 servers (k=25), 51 clusters and approximately 5100 nodes (about j=100 nodes per cluster). In particular, we compute the key space required for the storage of certificates, counter-certificates, CRL and DCA parameters per server, repository and node. Nodes choose to store as many certificates of the same cluster's nodes as needed, whereas the repositories have to store the certificates of all nodes

Table 4. Comparison of the *max* computations per server and the time required for each protocol (n = 51)

Max computations (modular multiplications) *per* server							
Protocol	1	2	3	4	5	6	7
EC-DCA scheme	13,314	8,574	17,578	4,694	16,915	164	2,221
T-DCA scheme	114,146	36,240	72,480	88,563	349,777	3,072	39,961
Time (in seconds)							
Protocol	1	2	3	4	5	6	7
EC-DCA scheme	4.44	2.86	5.86	1.56	5.63	0.05	0.74
T-DCA scheme	38.04	12.08	24.52	29.52	116.59	1.02	13.32

Table 5. Traffic generated within the DCA for each protocol (n = 51)

Protocol	Messages sent	Messages received
Distributed Signature	78	3901
Private key renewal	51	2550
Lost share detection	102	5100
Recovery of lost shares	147	4900
Share renewal	102	5100
Resolving accusation - 3 steps -	2	100
Resolving accusation - 4 steps -	2	100

in the cluster. Therefore, the key space required per node varies. In case all 100 certificates of the cluster are stored, we can calculate the maximum key space required per node, which equals the key space required per repository. We also assume that no more than 20 counter-certificates are issued per cluster; thus 51*20 in all clusters. The size of the CRL, under the assumption that one serial number is 32 bits, is 4.08KB. It is computed that the maximum key space required for the storage of certificates, counter-certificates, CRL and DCA parameters per server, repository and node are 29.66KB, 28.6KB, and 28.6KB, respectively. It is apparent that the key space required is very small. This is one of the major advantages of our system.

5 Conclusions

The proposed distributed CA-based PKI architecture addresses several key challenges in securing MANETs: (1) The physical vulnerability of the nodes in a hostile environment is addressed by employing the distribution of the CA's functionality across multiple nodes and using threshold cryptography with proactive recovery; (2) the insecurity of the wireless links is dealt with the use of keys so that the information exchanged is authenticated and encrypted; (3) the storage constraints are addressed with the use of ECC (the key size is reduced) and the use of clusters (the number of keys stored is reduced); (4) the energy constraints are addressed with the use of an ECC-based cryptosystem and clustering to reduce communication overhead. Finally, the use of clustering allows the proposed PKI scheme to scale to large networks. The proposed architecture could be implemented using current smartcard technology [21].

Acknowledgment

This work was supported in part by the U.S. National Science Foundation under Grant CCR-0209049.

References

1. N. Asokan, P. Ginzboorg, "Key Agreement in Ad-hoc Networks", Northsec 1999, Sweden.
2. S. M. Bellovin, M. Merrit, "Encrypted Key Exchange: Password-based protocols secure against dictionary attacks". In Proceedings of the IEEE Symposium on Research in Security and Privacy, 1992.
3. S. Lucks, "Open Key Exchange: How to defeat dictionary attacks without encrypting public Keys". In Security Protocol Workshop '97, Ecole Normale Suprieure, Paris, 1992.
4. D. P. Jablon, "Extended password key exchange protocols immune to dictionary attack". In Proceedings of the WETICE '97 Workshop on Enterprise Security, Cambridge, MA, USA, 1998.
5. T. Wu, "The secure remote password protocol". In Symposium on Network and Distributed Systems Security (NDSS '98), pages 97-111, San Diego, California, 1998. Internet Society.
6. J. P. Hubaux, L. Buttyan, S. Capkun. "The quest for security in mobile ad hoc networks". In Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2001.
7. S. Capkun, L. Buttyan, J.-P. Hubaux. "Self-Organized Public-Key Management for Mobile Ad Hoc Networks". Technical Report EPFL/IC/200234, Swiss Federal Institute of Technology, Lausanne, June 2002.
8. L. Zhou, Z. J. Haas, "Securing Ad Hoc Networks", IEEE Network Magazine, November 1999.
9. S. Yi, R. Kravets, "Key Management for Heterogeneous Ad Hoc Wireless Networks". Technical Report UIUCDCS-R-2002-2290/UILU-ENG-2002-1734, University of Illinois at Urbana-Champaign, July 2002.
10. L. Zhou, F. Schneider, R. van Renesse, "COCA: A Secure Distributed On-line Certification Authority". Technical Report, Cornell University, 2000. Revised, 2002.
11. P. Basu, N. Khan, T.D. Little, "A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks", In Proceedings of Distributed Computing Systems Workshop, 2001.
12. S. Banerjee, S. Khuller, "A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks" In Proc. of IEEE INFOCOM, pages 1028-1037, 2001.
13. C.R. Lin, M. Gerla, "Adaptive Clustering for Mobile Wireless Networks", IEEE Journal of Selected Areas in Communications, vol. 15, no. 7, pages 1265-1275, 1997.
14. P. Krishna, N. Vaidya, M. Chatterjee, D. Pradhan "A cluster-based approach for routing in dynamic networks", Proc. of ACM SIGCOMM Computer Communication, pages 49-65, April 1997.
15. A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, "Proactive Secret Sharing or: How to Cope with Perpetual Leakage". In D. Coppersmith, editor, Proc. of CRYPTO'95, volume 963 of LNCS, pages 339-352, 1995.
16. S. Jarecki , "Proactive Secret Sharing and Public Key Cryptosystems", Master's Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, September 1995.
17. D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag, New York, 2004.
18. Y. Desmedt, Y. Frankel, "Threshold cryptosystems. Advances in Cryptology"- CRYPTO, LNCS 435:307-315, 1990.
19. P. Feldman, "A practical scheme for non-interactive verifiable secret sharing". In Proc. of IEEE Fund. Of Comp., Sci., pages 427-437, 1987.
20. C. Zouridaki, "Evaluation of the Proactive Public Key and Signature System and a new implementation based on Elliptic Curves", M.S. Thesis, Dept. of ECE, George Mason University, 2002.
21. W. Rankl and W. Effing, Smart Card Handbook, 2nd edition, John Wiley & Sons, Ltd., 2000.