Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks

Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang
Computer Science Department
University of California at Los Angeles
{jkong,pzerfos,hluo,slu,lixia}@cs.ucla.edu

Abstract

Providing security support for mobile ad-hoc networks is challenging for several reasons: (a) wireless networks are susceptible to attacks ranging from passive eavesdropping to active interfering, occasional break-ins by adversaries may be inevitable in a large time window; (b) mobile users demand "anywhere, anytime" services; (c) a scalable solution is needed for a large-scale mobile network. In this paper, we describe a solution that supports ubiquitous security services for mobile hosts, scales to network size, and is robust against break-ins. In our design, we distribute the certification authority functions through a threshold secret sharing mechanism, in which each entity holds a secret share and multiple entities in a local neighborhood jointly provide complete services. We employ localized certification schemes to enable ubiquitous services. We also update the secret shares to further enhance robustness against break-ins. Both simulations and implementation confirm the effectiveness of our design.

1 Introduction

In recent years, network security has received critical attention from both academia and industry. As the data network becomes more pervasive and its scale becomes larger, network intrusion and attack have become severe threats to network users. This is especially true for the emerging wireless data networks. Compared to their wired counterpart, wireless networks are prone to security attacks ranging from passive eavesdropping to active interfering. As it is even more difficult to protect network entities against the intruders in wireless environment, occasional break-ins in a large-scale mobile network are nearly inevitable over a large time period.

While we may employ sophisticated security techniques into the system design to prevent intrusions, we expect complete intrusion-free systems to be costly and unrealistic, if not impossible at all. Therefore, in order to handle network intrusions, we expect a paradigm shift from completely preventing intrusions to tolerating intrusions to certain extent. We seek to operate in the presence of system intrusions so that the attacker's damage will be contained locally and the overall system security will not be compromised.

This work presents a scalable intrusion-tolerant security solution for infrastructureless wireless mobile networks, in which the network topology is dynamically changing due to user mobility and node failures. In our design, we use the idea of *threshold secret sharing* and *secret share updates* to enable intrusion tolerance. No single entity in the network knows or holds the complete system secret (e.g., a certification authority's signing key). Instead, each entity only holds a secret share of the certification authority's signing key. Multiple entities, say K, in a one-hop network locality jointly provide complete security services, as if they were provided by a single and omnipresent certification authority. The system security is not compromised as long as there are less than K collaborative intruders in each adversary group. To further resist intrusions over long term, we periodically (for example once every several hours) update the secret shares for all entities.

The concepts of threshold secret sharing [14, 2, 4, 12, 15] and secret share updates [7, 5] are not new, and have been studied in the cryptography context. However, these proposals assume limited number of secret share holders, and are not scalable to network size. They typically involve excessive communication overhead, and assume a richly-connected network topology. Besides the scaling issue, these solutions do not work well in the mobile networking environment. They cannot satisfy the following two requirements for mobile networking security: (1) Mobile users demand "anywhere, anytime" ubiquitous security services, since they may freely roam. As long as the network condition is better than a predefined lower bound, security services should be available all the time. (2) Compared to wired networks, wireless networks are constrained by their unique features. Security services must be provided despite wireless channel error, network partitioning, and entity joins/leaves. For the above reasons, these existing solutions are not applicable to mobile networks with dynamic membership.

Though we use the aforementioned cryptographic concepts in our work, our focus is to address network-oriented

security issues, in particular mobility, service ubiquity, network dynamics, and scalability. We employ several techniques to achieve this goal.

Our design employs a certificate-based approach based on the public key infrastructure (PKI), which has been the foundation of several recent network security protocols. Any two communicating entities may establish a temporary trust relationship via unforgeable, renewable and globally verifiable certificates carried by each of the entities. Security functions such as confidentiality, data integrity, authentication, and non-repudiation can be readily provided via valid certificates that are usually issued by a globally trusted certification server. However, in a large scale wireless mobile network, if we still rely on centralized certification servers to provide these security services such as certificate issuing, renewal and revocation, both the maintenance and performance of such servers are non-trivial issues. In this paper, we propose new schemes to realize the certificate-related security services to accommodate the unique characteristics of ad-hoc wireless networks.

We provide ubiquitous services for mobile entities by distributing the certification authority(CA)'s functionality to each local neighborhood. A coalition of K neighbors can serve as the CA and jointly provide certification services for a requesting mobile entity. The fully localized and everywhere available features of our design enable service ubiquity for mobile users.

A novel self-initialization protocol is proposed to handle dynamic node membership (i.e., joins and leaves) and secret share updates. Each node can be (re)initialized by K neighbors. Once initialized, a node is qualified to be a coalition member to serve its neighborhood.

Our overall design scales to large network size. Security services are effectively provided in the presence of mobility, wireless channel errors, network partitioning, and node failures. Our implementation and network simulations have confirmed the effectiveness of our proposal.

The focus of this paper is wireless ad-hoc networks that do not have any infrastructure support, thus making the problem more challenging. But our design is equally applicable in several other scenarios. It can be plugged as value-added security service into many networking systems, such as cluster-based middleware and storage area networks.

This paper is organized as follows. We begin in § 2 by describing the problem and showing why the conventional approaches fail to solve it. § 3 presents our security architecture and the design rationale. § 4 illustrates the protocol design. § 5 describes the evaluation and measurements on our implementation on Unix and *ns-2* simulator, and § 6 concludes this paper.

2 Background

We consider a dynamic wireless ad-hoc network with N networking hosts/entities. Each entity i has a globally unique nonzero ID v_i. There is no other constraint on selecting the ID as long as it can uniquely identify the corresponding entity. Entities communicate with one another via the bandwidth-constrained, error-prone, and *insecure* wireless channel. They may freely roam in the network. The number of network entities N may change over time because mobile hosts may join, leave, or crash. Besides, N is not limited. There may be a large number of communicating entities.

2.1 Design challenges

Security design in such infrastructureless wireless mobile networks is challenging for several reasons:

- Security breach: Wireless transmissions are prone to security attacks, and it is very likely that adversaries will eventually break into a limited number of entities over a large time window.
- Mobility and service ubiquity: Mobile users incur dynamic topological changes. A mobile user may be able to perform effective and timely communication with its local neighbors but *not* with remote entities. For example, routing protocols may fail to establish robust communication over multi-hop paths, as it is the case with DSR, which is limited to 10-hop scenarios [8].
- Network dynamics: Channel errors, and node failures all incur dynamics into the network. Besides, an entity may join and leave the network over time.
- Network scale: The number of networking devices can be large, thus a scalable solution is critical.

2.2 Intrusion model

At first we briefly discuss what kind of intrusions is allowed in this work. In the worst case, all information, whether public or private, is known to the intruder when a network entity is compromised. The intruder can forge, modify, and delete any information. The intruder can also do bookkeeping to facilitate future break-ins. However, we have to limit the power of an intruder to make the problem tractable. Giving infinite power to the intruder simply makes any security design meaningless. We consider a realistic intrusion model in the system.

Since authentication is the basic building block for all security services, we focus our discussion on this part. Fundamentally, we have to assume that each network entity has some information that is unknown to or unforgeable by the intruder. Otherwise, once an entity is broken, there is no way others can differentiate the intruder and the genuine entity. We consider two specific cases:

1. An entity's private key will not be exposed for a certain period of time. Thus an entity is able to maintain its security identity by periodically renewing its private key via certificate renewal services (§ 4).
2. An entity's ID v_i is not forgeable by the intruder, or the intruder can be detected by intrusion detection mechanisms when it pretends to be the broken entity. Intrusion detection within one hop is more practical and some schemes have been recently proposed [10].

In the first case, we allow the intruder to know, modify and forge information other than the private key. The broken entity can authenticate itself via its valid certificate and a common challenge-response scheme on its certified public key. The intruder cannot answer the challenge without the private key. In the second case, we allow all information, including its private key, to be exposed to the intruder. The intruder and the broken entity are differentiated by the unforgeable ID. For example, intrusion on the ID can be detected by tamper resistance schemes [6] or localized one-hop monitoring (e.g., perception-based monitoring).

2.3 Problems with conventional approaches

Then we show why two common approaches, namely the centralized and the hierarchical approach, do not work well in large mobile networks.

In the centralized approach, a single certification authority (CA) provides certification services for the entire network. In a large mobile network, the scalability problem with this approach is quite obvious. Besides, from the security aspect, the CA will be exposed to single point of failure due to system faults, compromises and denial-of-service attacks.

In a hierarchical approach, the entire network is logically partitioned into domains where local CAs are deployed. This is also the approach discussed in [20] where collaborative CAs are deployed as access points for security services. At a first glance, this scales to network size and fits well in a large wireless network. However, several characteristics of mobile networks make this approach ineffective: (1) High mobility causes frequent route changes, thus contacting the local CA in a timely fashion is non-trivial [8]. Besides, in ad-hoc networks the local CA may be multi-hops away and may also move. This not only causes complicated dynamic repartitioning of the network, but also stretches the problem of locating and tracking a local CA server. (2) Multi-hop communication over the error-prone wireless channel exposes the data transmission to high loss rate. This reduces the success ratio and increases the average service latency. (3) Every local CA is exposed to single point of compromises or DoS attacks. Threshold secret sharing among local CAs [20] solves the problem, but aggravates the previous two concerns.

We use certificate renewal service as an example to evaluate these approaches in the $ns2$ network simulator. We measured results with mobility speed set at the moderate value of $5m/sec$ and various network sizes. In Figure 1, we observe that the success ratio, defined as the percentage of successful certificate renewals over all requests during the simulation time, is low for centralized approaches (around 70% – 90%), while our scheme is close to 100%. In Figure 2, we observe that the average delay required by each node to contact the local CA is much larger compared with our fully localized approach.

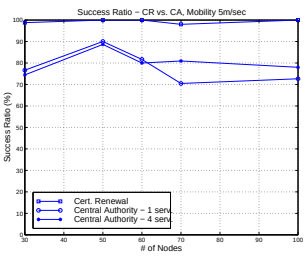

Figure 1. Success Ratio vs. node #

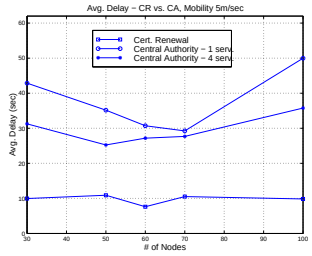

Figure 2. Avg. delay vs. node #

3 Architecture

In this section we describe our architecture for ubiquitous security services in wireless ad-hoc networks. From the cryptographic perspective, our design is based on the concepts of threshold secret sharing and secret share updates. From the system aspect, the architecture is fully distributed and localized.

3.1 Concepts

We adopt an RSA-based design, which is currently the most prevalent public key cryptosystem. The system CA's RSA key pair is denoted as $\{SK, PK\}$, where SK is the system secret/private key and PK is the system public key. SK is used to sign certificates for all entities in the network. A certificate signed by SK can be verified by the well-known system public key PK.

By threshold secret sharing, SK is shared among the network entities. Each entity v_i holds a *secret share* P_{v_i}, and *any* K of such secret share holders can collectively function as the role of CA. However, SK is not visible or known by any component of the network except at the system bootstrapping phase. We seek to preserve the secrecy of SK all the time after then.

Besides the system key pair, each entity v_i also maintains a personal RSA private and public key pair $\{\overline{sk_i}, \overline{pk_i}\}$. This pair of personal keys is used in end-to-end security to realize cipher key exchange, message privacy, message integrity, and non-repudiation.

To certify its personal keys, each entity v_i also holds a certificate $cert_i$ in the format of $\langle v_i, \overline{pk}_i, T_{sign}, T_{expire} \rangle$, which may read: "It is certified that the personal public key of entity v_i is $\overline{pk}_i$ from the signing time T_{sign} to the expiration time T_{expire}". A certificate is valid if it is signed by SK. To further control the validity of a certificate, we have employed two complementary methods.

Implicit certificate revocation We predefine a system parameter T_{renew} to bound the valid time of every certificate in the network. In any certificate $cert_i$, the condition $(T_{expire} \leq T_{sign} + T_{renew})$ must hold. In other words, the entity v_i has to renew its certificate within T_{renew}.

Explicit certificate revocation A certificate accused by the system certificate revocation list (CRL) is invalid no matter what its valid time says. Due to the implicit certificate revocation mechanism, only those certificates that haven't expired yet need to be stored in the CRL, thus saving its storage.

3.2 Basic operations

The basic operations in our architecture involve only local coalitions of secret share holders. K, the size of these coalitions, is an important system parameter that needs to be carefully tuned.

Secret share dealing An entity v_i obtains its secret share P_{v_i} during system bootstrapping phase or through our self-initialization service (§ 4.2). In the bootstrapping phase, network entities obtain their valid certificates and secret shares from a centralized management before joining (or forming) the ad-hoc network. To ease the job of secret share dealing, we have devised a self-initialization algorithm to securely deliver the secret share to a uninitialized entity by a local coalition of K secret share holders. As a result, after initializing K entities, the centralized dealer is not needed any more.

Certification services When an entity requests for certification service, a local coalition of K secret share holders is formed on the fly. Each secret share holder v_i provides to the requester a partial certificate that is signed by a value SK_i which is directly derived from the secret share P_{v_i}. Once the requester locally collects K such partial certificates, it combines them together and obtains its complete certificate that is signed by SK.

Secret share updates In our design, no adversary group having less than K collaborative adversaries can forge a valid certificate. Our system tolerates up to $K-1$ break-ins from each adversary group. In order to resist gradual break-ins over a long term period, each entity's secret share is updated periodically [7, 5]. As long as there are less than K entities broken between two consecutive secret share updates, the system signing key SK is protected against break-ins and can remain unchanged throughout.

3.3 Design rationale

Why a certificate-based approach? As it has been proposed in literatures [18], there are five basic aspects that characterize a secure communication: *Data integrity*, which ensures that only authorized parties are able to modify transmitted information. *Authentication*, through which the sender of a message can be correctly identified. *Message confidentiality*, which ensures that transmitted data are accessible only for reading by authorized entities. *Non-repudiation* requires that neither the sender, nor the receiver of a message be able to deny the transmission and *Service availability*, which demands that computer system assets be available to authorized entities when needed.

Certificate-based approaches readily provide workable solutions to the first four of the above mentioned functions. We seek to solve the problem of service availability through the design of the ubiquitous certification protocols, specifically for mobile ad-hoc networks.

Why threshold secret sharing? Threshold secret sharing exhibits several desirable properties that fit well in ad-hoc networks: (a) A key feature of ad-hoc networks is lack of centralized control. Distributing and sharing the control are consistent to the nature of ad-hoc networks. (b) Ubiquitous services are enabled once the secret is fully distributed to each locality. Besides, intrusion detection is more practical and efficient if localized. (c) The threshold K is the balance point between service availability and intrusion tolerance. An adversary group must destroy $(N-K+1)$ share holders to turn off certification services, whereas it must break in at least K share holders to steal the system secret SK. We next compare our choice of K-threshold with two extreme cases in the entire solution space:

- **"1 out of N" scheme** The centralized solution is actually a special case of the threshold secret sharing. The threshold $K = 1$ results in vulnerable security since SK is held by a single entity. The centralized server suffers from a single point of service denial and a single point of compromise. In the former case, no service can be provided to network entities. In the latter case, the system secret SK is revealed and the entire system is compromised.

- **"N out of N" scheme:** The threshold $K = N$ results in maximal security but minimal fault tolerance. Unless the adversary breaks into every entity in the system, it cannot expose SK. However, since all N entities are required in providing the service, the system secret is lost once system failure occurs at any single entity. Besides, this approach is not scalable in a large network.

Therefore, our K-threshold security design seeks to provide flexible tradeoffs among security, availability, intrusion tolerance, and fault tolerance. By choosing $1 < K < N$, we

avoid both the "single point of compromise" and the "single point of failure/DoS attack" problems. Though a certain number of entities may be intruded or faulty, the system secret SK is not exposed or lost.

Why secret share updates? In threshold secret sharing, adversaries need to compromise at least K entities to expose the system secret SK. However, they have the entire system lifetime to mount these attacks. Gradual break-ins into K entities over a long period of time may be possible and therefore long-lived secret sharing is not sufficient. A naive make-up is to periodically change SK itself. However, in a wireless ad-hoc network without centralized control, it is an open issue who has the authority to annul the current SK and enforce the change. This motivates our design choice to periodically update the shares instead of the SK itself.

4 Protocols

4.1 Localized certification service

As in PKI-based systems, our certification services include certificate issuing, renewal, and revocation. The renewal service also serves as an implicit revocation mechanism, as described in § 3.

Certificate issuing Technically this operation is same as certificate renewal. However, it raises more security concerns. Once an entity obtains its initial certificate, in our design it earns the trust of the entire network. A well-defined certificate issuing policy is needed to regulate certificate issuing. (a) At the network bootstrapping phase, we assume the entities can obtain their initial certificates from a trusted centralized management. (b) If an entity joins the network later on, or if it wishes to recover its certificate from a system crash, then a well-known certificate issuing policy must be pre-determined before the ad-hoc network is formed. Since our architecture delivers certification services within one-hop neighborhood, we suggest to use some reliable out-of-bound physical proofs, such as human perceptions and biometrics, to enforce certificate re-issuing policy.

Certificate renewal Once an initial certificate is issued to an entity, it must be renewed within T_{renew} time. The entity may also need to renew its certificate once it updates its personal key pair. To renew its certificate, a network entity must present its current valid certificate and a future expiration time $T <$ (current time$+T_{renew}$) for the new certificate. The local coalition uses the system PK and system CRL to verify the validity of the certificate, then accepts or denies the renewal request accordingly.

Certificate revocation Besides the implicit revocation scheme, the explicit revocation scheme is devised to revoke compromised certificates at real time. If v_x's certificate is considered compromised, an SK-signed counter-certificate $\langle \perp v_x, T^{\perp}_{sign} \rangle_{SK}$ is flooded over the network, where $\perp$ is a special tag denoting counter-certificates and $T^{\perp}_{sign}$ is the time when the counter-certification request is submitted. By the help from implicit revocation, each node only needs to maintain a subset of counter-certificates within the past T_{renew}. That is, given a counter-certificate $\langle \perp v_x, T^{\perp}_{sign} \rangle_{SK}$ and the current time NOW, a node needs to store the counter-certificate if $(T^{\perp}_{sign} + T_{renew} > NOW)$, or discard it otherwise.

Like a certificate, a counter-certificate is signed thus unforgeable. Besides the flooding mechanism, neighboring nodes can safely exchange their local CRL cache and obtain the maximal list. If a counter-certificate $\langle \perp v_x, T^{\perp}_{sign} \rangle_{SK}$ is listed in local CRL cache, any certificate of v_x signed before $T^{\perp}_{sign}$ is considered invalid. An out-of-bound certificate re-issuing policy may decide whether to re-issue v_x a new certificate signed after $T^{\perp}_{sign}$.

4.1.1 Protocol details

Our architecture is built upon Shamir's [14] threshold secret sharing. A secret can be shared by an arbitrary large community using a secret polynomial $f(x)$. If the degree of $f(x)$ is $K-1$, then any K members of the community can recover the secret via Lagrange interpolation, while any less than K members of the community reveals no information of the secret. This is normally denoted as K-threshold secret sharing.

At the system bootstrapping phase, the centralized secret share dealer obtains the RSA secret key $SK = \langle d, n \rangle$ and randomly selects a polynomial $f(x)$ of degree $K-1$, $f(x) = d + f_1 \cdot x + \cdots + f_{K-1} \cdot x^{K-1}$ such that the shared secret is $f(0) = d$. Each entity $v_i, (i = 1, 2, ..., N)$ holds a secret share $P_{v_i} = (f(v_i) \bmod n)$. For any coalition of K entities $\{v_1, v_2, \ldots, v_K\}$, Lagrange interpolation states that

$$d \equiv \sum_{j=1}^{K} (P_{v_j} \cdot l_{v_j}(0) \bmod n) \equiv \sum_{j=1}^{K} SK_j \pmod n \qquad (1)$$

where $l_{v_j}(0)$ are the Lagrange coefficients[1].

In a local neighborhood where there are K secret share holders, each share holder v_j can compute an SK_j from its secret share P_{v_j} by Lagrange interpolation, SK is recovered from the sum $d = (\sum_{j=1}^{K} SK_j \bmod n)$.

Instead of revealing the private exponent d to the coalition, a better security scheme is employed to accomplish certification services without constructing an explicit d. The corner stone of the *multi-signature* protocol [2, 4, 12, 15] is the following arithmetic formula:

$$X^{SK_1} \cdot X^{SK_2} \cdot \ldots \cdot X^{SK_K} = X^{SK_1 + SK_2 + \cdots + SK_K}.$$

In this scheme each member provides a partial certificate X^{SK_j} rather than revealing its private SK_j (Figure 3).

[1] Lagrange coefficient in the coalition is defined as $l_{v_j}(x) = \frac{(x-v_1)\cdots(x-v_{j-1})(x-v_{j+1})\cdots(x-v_K)}{(v_j-v_1)\cdots(v_j-v_{j-1})(v_j-v_{j+1})\cdots(v_j-v_K)}$.

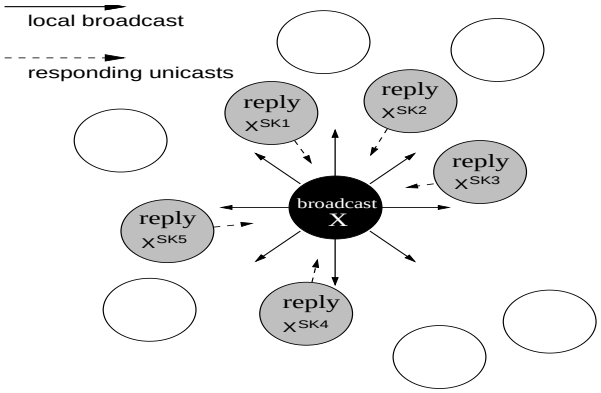

Figure 3. Localized Certification Service

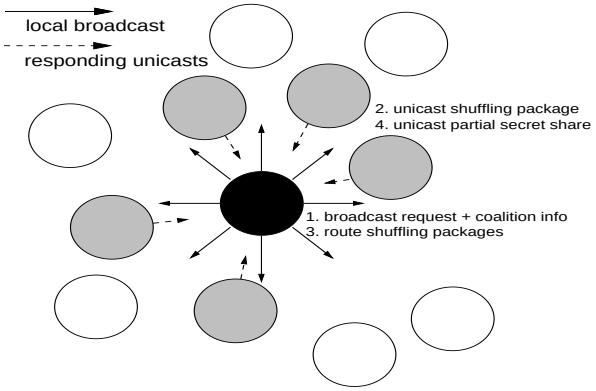

Figure 4. Localized Self-initialization Service

However, there is a technical obstruction, named as the "interpolation over $Z_{\phi(n)}$ problem" in [15], that needs to be solved to implement the multi-signature scheme. The solution proposed in [2, 4, 12] assumes a limited number of secret share holders as for fixed members of a social or legal group [1]. The total number of all share holders is considered constant, and the entire system needs to restart if this number changes. In other words, the existing secret shares and sometimes the current signing key need to be regenerated when a new member joins. Contributions from [5, 15] improve the robustness of scheme by various techniques like verifiable secret sharing [3, 17, 13] and proactive secret share update [7]. However, the application domain is still group-oriented multi-signature with fixed number of share holders. Their algorithms cannot be directly applied in large-scale networks with dynamic node membership.

We have devised a new algorithm to meet the demand of scalability and dynamic node membership [9]. In our architecture, membership changes do not affect existing secret shares or the current signing key.

Secret share dealing Unlike [2, 4, 12], we follow the simple procedures specified by Shamir [14]. Given an RSA signing key $SK = \langle d, n\rangle$, the shared secret is the private exponent $f(0) = d$, and the secret share for entity v_i is $P_{v_i} = (f(v_i) \bmod n)$.

Generating partial certificates using secret shares When a secret share is used in signing, it is treated as an exponent in RSA algorithm. Given a message M and a secret share P, the signed result is $(M^P \bmod n)$.

Combining partial certificates Having collected K partial certificates, the service requester can obtain the complete certificate by K-bounded coalition offsetting algorithm.

4.1.2 K-bounded coalition offsetting

In Equation 1, the sum of Lagrange interpolation $\sum_{i=1}^{K}(P_{v_i}{\cdot}l_{v_i}(0) \bmod n) = t{\cdot}n + d$ for certain t. However, no mathematical identity ensures that the result of the multiplicative multi-signature equals the SK-signature: $M^{t\cdot n+d} \equiv M^{t\cdot n}{\cdot}M^d \not\equiv 1{\cdot}M^d \equiv M^d \pmod n$.

Fortunately, each $(P_{v_i}{\cdot}l_{v_i}(0) \bmod n)$ is a value between 0 and $n-1$ due to modular arithmetic. Thus t satisfies the inequation $0 \le t \le K$. After Algorithm 1 we are able to recover M^d by the help from the original message M and the system public key $PK = \langle e, n\rangle$.

Algorithm 1 K-bounded Coalition Offsetting

Require: $Y_0 \equiv M^{\sum_{i=1}^{K}(P_{v_i}{\cdot}l_{v_i}(0) \bmod n)} \equiv M^{t\cdot n+d} \pmod n$ is the product of all partial certificates.

```
Z := M^{-n} mod n
j := 0, W := 1
while j ≤ K do
    Y := Y_0 · W mod n, then W := W · Z mod n
    if (M ≡ Y^e (mod n)) then
        Success, break the loop
    end if
    j := j + 1
end while
```

Ensure: $Y \equiv M^d \pmod n$

In an ad-hoc network, K is a small number corresponding to number of nodes in a neighborhood. Thus the loop in Algorithm 1 ends within reasonable rounds. Also it is well-known that PK-verification in RSA is an inexpensive operation [19]. The complexity of K-bounded coalition offsetting is the sum of $O(1)$ exponentiation, $O(k)$ modular multiplications, and $O(k)$ RSA PK-verifications.

4.2 Localized self-initialization

Each well-behaving, certificate-holding entity v_x can also obtain a secret share P_{v_x}, which is used to derive SK_x in providing certification services. We have devised a localized self-initialization algorithm to compute secret shares when the centralized dealer is absent.

Given an uninitialized entity v_x (i.e., v_x is not a secret share holder), a local coalition of K share holders,

$\{v_{(x,1)}, v_{(x,2)}, \cdots, v_{(x,K)}\}$, can compute its secret share P_{v_x} by Lagrange interpolation:

$$P_{v_x} \equiv f(v_x) \equiv \sum_{j=1}^{K} P_{v_{(x,j)}} \cdot l_{v_{(x,j)}}(v_x) \equiv \sum_{j=1}^{K} SS_{(x,j)} \pmod{n}.$$

Each coalition member $v_{(x,j)}$ computes a partial secret share $SS_{(x,j)}$ from its secret share $P_{v_{(x,j)}}$, then returns the partial secret share to the requester. The requester's secret share is the sum of the K partial secret shares.

As Lagrange coefficients are publicly known, v_x can derive $P_{v_{(x,j)}}$ by knowing $SS_{(x,j)}$ directly. To keep $P_{v_{(x,j)}}$ as a secret only to its owner $v_{(x,j)}$, we employ a complete *shuffling* scheme. In the complete shuffling scheme, a random nonce is exchanged between any two members in the coalition. The entity with larger ID treats the nonce as a positive number while the other side treats it as a negative number. Each member totally has $K-1$ such nonces. Before $v_{(x,j)}$ sends back the partial secret share, it sums the $K-1$ nonces and $SS_{(x,j)}$, then sends a shuffled partial secret share $SS'_{(x,j)} = SS_{(x,j)} + \sum(nonces)$ to v_x. It is easy to verify that v_x obtains the same value P_{v_x}.

As depicted in Figure 4, the protocol requires four steps of communication:

1. The uninitialized node v_x broadcasts the service request, along with local coalition information.
2. Each coalition member selects a random nonce for other members if its ID is lower than the other one. Each nonce is encrypted with the personal $\overline{pk}$ of the intended receiver. The requester acts as the router and accepts all shuffling packages from $K-1$ members (The member with the highest ID needs not select nonces).
3. The requester routes encrypted nonces to intended receivers.
4. Each member decrypts all nonces, computes a shuffled partial secret share, and then sends it back to v_x.

It can be shown that the self initialization protocol is still K-out-of-N secure if there are at least two uncompromised entities in the coalition. The cryptoanalysis details are available in [9].

4.3 Secret share update

To further enhance the robustness of our design, we periodically update all the secret shares to invalidate compromised secret shares. In the bootstrapping phase all secret shares are tagged with version 1 and ID 0. Each secret share update will increase the version by 1, and there is a predefined minimal lapse time between two consecutive updates. During the update transition period, secret share holders with the same version tag can collaboratively function as the CA.

We employ the proactive secret share update algorithms proposed in the literature [7] to create a community of K entities with new version of secret shares. Then the self-initialization protocol is employed to propagate the new version over the entire network. If there are multiple proactive updates concurrently going on, then we may have version conflicts. Our solution is to include the lowest ID of the original K updated entities into the version tag. The version tag with the lowest ID wins in the case of version conflicts.

4.4 Discussions

Verifiable Secret Sharing If there are compromised secret share holders in the network, values other than the secret shares may be used by the adversaries to sign and issue false partial-certificates. With verifiable secret sharing (VSS) [3, 17, 13] employed with the multi-signature algorithms, signing a message with a wrong secret share can be detected publicly or by the service requester. In the proactive update scheme proposed by [7], VSS is also employed to enforce proper secret share updates. In our architecture, the self-initialization algorithm can also be enhanced by VSS with witnesses associated with each partial secret share. The cryptographic details are available in [9].

Less than K one-hop neighbors So far we assume that the requesting entity has at least K one-hop neighbors. However, if the users constantly roam, this may not always hold. In our solution the requesting entity may broadcast the requests for a limited number of times (e.g., 2–3) over a time window, expecting new mobile entities to serve it. Similarly, the requesting node may also move to a new location, where it can find at least K share holders to serve it. Thus node mobility helps providing certification services (§ 5.2.1).

Storage requirements The system CRL is required to be locally stored at each node. From our implementation experience the size of a counter-certificate is normally in the range of 128 to 256 bytes (as for RSA signing key length 1024 to 2048 bit). When the size of an ad-hoc network $0 < N < 1000$ and the probability of compromise $0 \leq P \leq 1$, the storage required for system CRL is acceptable for most low-end devices. Besides, the implicit revocation mechanism helps to reduce the storage requirement significantly (§ 3).

5 Evaluation of implementation

We realize our design in both Unix platforms and a popular network simulator *NS-2* [11]. Our Unix implementation seeks to quantitatively characterize the computational cost of our solution, and the simulator helps to evaluate as-

key (bit)	RSA-PK (msec)	RSA-SK (sec)	PCC (sec)	Combine (sec)
512	0.093	0.0056	0.0466	0.0928
768	0.124	0.0173	0.1198	0.2416
1024	0.142	0.0386	0.2610	0.5280
1280	0.136	0.0669	0.4590	0.9742
1536	0.133	0.1089	0.7944	1.5598
2048	0.208	0.2462	1.7058	3.4410

Table 1. RSA and certification performance ($K = 5, SPEC = 20.5$)

key (bit)	RSA-PK (msec)	RSA-SK (sec)	PCC (sec)	Combine (sec)
512	0.884	0.0678	0.1835	0.1982
768	1.276	0.2165	0.5973	1.3430
1024	1.324	0.4672	1.1637	1.1978
1280	1.356	0.8734	2.2912	2.4109
1536	1.416	1.4863	3.5820	3.6952
2048	1.036	3.1883	7.7855	8.0324

Table 2. RSA and certification performance ($K = 5, SPEC = 12.1$)

key (bit)	RSA-PK (msec)	RSA-SK (sec)	PCC (sec)	Combine (sec)
512	2.782	0.2347	0.5499	0.6144
768	3.382	0.6403	1.4818	1.6478
1024	4.036	1.2953	3.1738	3.3283
1280	4.065	2.4607	5.5492	5.9019
1536	3.941	3.8543	10.1253	10.4301
2048	3.954	8.3826	20.6606	21.7095

Table 3. RSA and certification performance ($K = 5, SPEC = 1.37$)

K	$SPEC = 20.5$		$SPEC = 12.1$		$SPEC = 1.37$	
	PCC	Combine	PCC	Combine	PCC	Combine
2	0.260	0.526	1.293	1.334	2.991	3.304
3	0.261	0.528	1.149	1.171	2.998	3.293
5	0.261	0.528	1.164	1.198	3.174	3.328
7	0.263	0.531	1.140	1.207	3.163	3.530
10	0.262	0.537	1.309	1.410	3.099	3.394
20	0.261	0.532	1.308	1.464	3.078	3.458
30	0.261	0.537	1.160	1.510	3.082	3.410

Table 4. Certification performance in terms of system parameter K (RSA key: 1024bit, time unit: sec)

pects of mobility, ability to handle ubiquitous service, channel and node dynamics, in a large network setting.

5.1 Unix implementation and measurements

Our cryptographic implementation on UNIX is written in C and currently consists of about 8,000 lines of code. It implements certification, counter-certification, self-initialization, and proactive secret share update services, along with a number theory module and other supportive modules.

In Tables 1, 2 and 3 we measured[2] the performance of our certification service with the standard RSA operations on heterogeneous devices. The SPECint95 values [16] are 20.5, 12.1, and 1.37, respectively. Our measurements show that computation power is a critical factor that affects the efficiency of our RSA based scheme, and for typical scenarios the performance is acceptable. For example, a PentiumIII/500 laptop (SPECint95=20.5) performs well in all test cases, while a SPARCstation5/85 (SPECint95=1.37) requires more time (3–5 sec for certification service) when using typical values for key length (1024 or 1280 bits) and coalition size K ($\leq$10 in practice).

key (bit)	$SPEC = 20.5$		$SPEC = 12.1$		$SPEC = 1.37$	
	PSS	Sum	PSS	Sum	PSS	Sum
512	0.413	0.288	1.145	0.378	3.861	1.196
768	0.459	0.382	2.588	0.443	5.163	1.497
1024	0.490	0.319	3.321	0.781	7.024	1.847
1280	0.561	0.411	4.926	0.840	8.215	1.996
1536	0.798	0.460	3.480	0.630	10.251	2.006
2048	1.420	0.473	5.245	0.754	24.414	2.528

Table 5. Self initialization service ($K = 5$, time unit: msec)

In Table 4 we measured the performance of the certification service in terms of the coalition size K. We find that parameter K does not affect the system performance significantly because (i) partial certificates are computed in parallel by the coalition members; (ii) at the requester's side, all the operations in the partial certificate combination loop are moderate in terms of computation overhead (§ 4.1.2).

The operations used in self-initialization and proactive update, namely multiplicative inverse and Lagrange interpolation, are inexpensive to compute. We present Table 5 to show the effects. All measured computation overheads are at the scale of milliseconds.

5.2 *NS-2* simulation

We have also used the *ns-2* simulator to implement all the communication protocols described in § 4. We follow an application-layer approach and have developed a UDP-like transport agent that allows for delivery of actual application data units (ADUs) and one-hop broadcast.

In order to evaluate the communication efficiency of our protocol, we use the following metrics: *Success ratio* measures the ratio of the number of successful certification services over the number of attempts during the simulation time. *Average delay* measures the average latency for each node to perform a certification service, in the case of self initialization, the average time it needs to become a fully functional member of the network, from the moment it joins in. *Average number of failures* measures the number of times an entity fails on average, before successfully accomplishing its certification.

[2] In all the tables, key denotes RSA key length in bits, RSA-PK denotes standard RSA's PK-verification. RSA-SK denotes standard RSA's SK-signature. PCC denotes partial certificate computation which equals using secret share to sign a message. Combine denotes the delay caused by combining K partial certificates. PSS denotes partial secret share computation which equals a Lagrange interpolation operation. Sum denotes obtaining a secret share by summing together all partial secret shares.

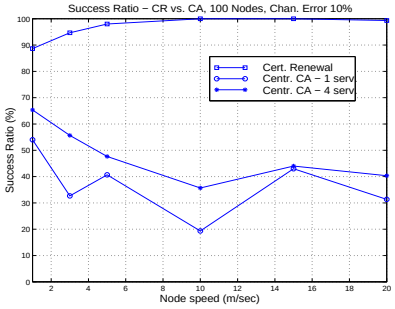

Figure 5. Certificate Renewal: Success Ratio vs. Node Speed, Error Rate 10%

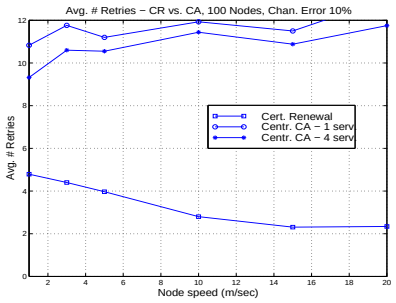

Figure 6. Certificate Renewal: Avg. # of Failures vs. Node Speed, Error Rate 10%

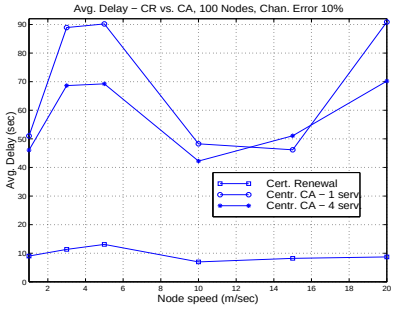

Figure 7. Certificate Renewal: Avg. Delay vs. Node Speed, Error Rate 10%

We study the performance of our protocols by running experiments in networks with sizes that range from 30 to 100 nodes. The node mobility varies from 1, 3, 5, 10, 15 and $20m/sec$. The random way-point model in *ns-2* is used to emulate mobility patterns. In the simulations, the expiration time of the certificate is selected as five minutes, and the coalition size $K = 5$, except for the topologies that consist of 30 nodes, where $K = 3$.

5.2.1 Certification services

We first examine the effectiveness of the certificate renewal service, as the node speed increases from $1m/sec$ to $20m/sec$ and the channel error rate becomes 10%. As it is shown in Figure 5, the success ratio of our approach is almost 100%, while the centralized and the hierarchical solutions fail. This confirms not only the effectiveness of our method in terms of mobility, but also service ubiquity, since during the simulation time, every node is required to renew its certificate multiple times, which means that the service should be available at any part of the network topology, at any time. The robustness of our distributed certification services is also demonstrated in Figure 6, from another perspective; the number of failures each node is experiencing on average, before successfully receiving its service. Our proposal requires significantly less effort in providing the service, compared to the centralized and the hierarchical cases. Moreover, we observe that mobility helps our protocol. As node speed increases, the average number of failures for each node not only remains unchanged in our approach, but also diminishes.

In Figure 7, we also present results for the average delay. From the figures, we observe that the average delay almost remains unchanged as mobility speed grows from $1m/sec$ to $20m/sec$. However, as it is evident from the same figures, both centralized and hierarchical solutions incur much higher delay, which also greatly fluctuates, thus making it hard to predict some useful information, such as the future expiration time in certificate renewal and consequently the frequency of renewal.

5.2.2 Self initialization & Proactive update

To evaluate the self initialization protocol of our certification services we focus on the time needed for the nodes that haven't been already initialized by the root-of-trust to become fully functional entities (by obtaining a secret share). In each experiment that we conducted, $2 * K$ nodes of our topology are assumed to have been initialized by an imaginary dealer, so that the remaining nodes be able to find a coalition of K neighbors, in order to perform self initialization. Figure 8 presents the average latency for each node to complete this phase. From the graph, we note that our algorithm scales well to the network size and node mobility; even for the largest topology of 100 nodes and node speed fixed at $20m/sec$, all nodes manage to self initialize in less that 500 seconds.

In the next set of figures 9 and 10, we present a more detailed analysis of the proactive update latency, for network topologies of 50 and 100 nodes respectively and for three node speeds (3, 10 and 15 m/sec, that correspond roughly to low, medium and high mobility). For the case of 50 nodes, we see that the first 20% of them needs almost 50 seconds to update. But as soon as a sufficient number of nodes manages to acquire their new secret shares, then the convergence of the algorithm is pretty fast, since those nodes may in turn help others to self initialize and so on; we reach 80% in another 50 seconds of simulation time. We also observe that the evolution of the algorithm is similar for all mobility speeds, which shows that our design is tolerant to mobility. The results for the scenario of 100 nodes are slightly different. As expected, the curves are shifted to the right, since the network is larger and consequently it takes up more time for all nodes to become part of it.

6 Conclusions

In this paper, we have described a solution to security support in wireless mobile networks. Our design has been motivated by three main factors: (a) We do not believe that any security system is completely unbreakable. Therefore,

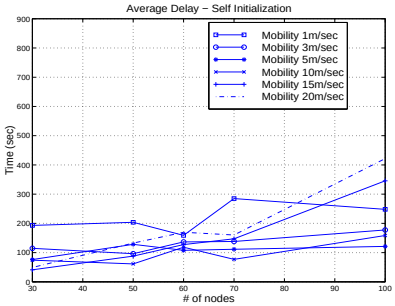

Figure 8. Self Initialization: Avg. Delay vs. Node Speed

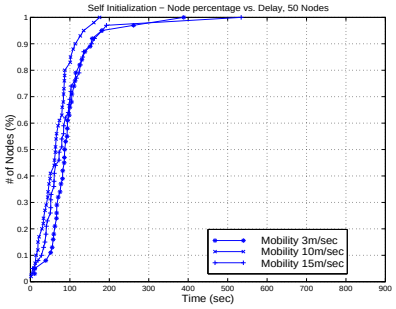

Figure 9. Proactive Update: Node percentage vs. Delay, 50 Nodes

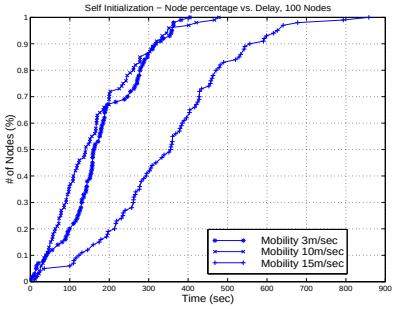

Figure 10. Proactive Update: Node percentage vs. Delay, 100 Nodes

our design has to work in the presence of such break-ins. (b) We seek to maximize the service availability in each network locality; this is crucial to supporting ubiquitous services for mobile users. (c) The solution has to be fully decentralized to operate in a large-scale network. To this end, we have addressed networking issues including mobility, scalability, service ubiquity, and network dynamics such as channel interference and node failures. Our experiences in implementation and simulations have shown positive results for our approach.

References

[1] Y. Desmedt. Society and Group Oriented Cryptography: A New Concept. In *CRYPTO*, pages 120–127, 1987.

[2] Y. Desmedt and Y. Frankel. Shared Generation of Authenticators and Signatures (Extended Abstract). In *CRYPTO*, pages 457–469, 1991.

[3] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *FOCS*, pages 427–437, 1987.

[4] Y. Frankel and Y. G. Desmedt. Parallel Reliable Threshold Multi-signature. Technical Report TR-92-04-02, Dept. of EECS, University of Wisconsin-Milwaukee, 1992.

[5] Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Optimal Resilience Proactive Public-Key Cryptosystems. In *FOCS*, pages 384–393, 1997.

[6] J. Hastad, J. Jonsson, A. Juels, and M. Yung. Funkspiel Schemes: an Alternative to Conventional Tamper Resistance. In *ACM CCS*, 2000.

[7] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Secret Sharing or: How to Cope with Perpetual Leakage. extended abstract, IBM T.J. Watson Research Center, November 1995.

[8] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[9] H. Luo and S. Lu. Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks. Technical Report TR-200030, Dept. of Computer Science, UCLA, 2000.

[10] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *MOBICOM*, 2001.

[11] NS-2 (The Network Simulator). `http://www.isi.edu/nsnam/ns/`.

[12] A. D. Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely (Extended Summary). In *STOC*, pages 522–533, 1994.

[13] B. Schoenmakers. A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting. In *CRYPTO*, pages 148–164, 1999.

[14] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[15] V. Shoup. Practical Threshold Signatures. In *EUROCRYPT*, pages 207–220, 2000.

[16] Standard Performance Evaluation Corporation. `http://www.specbench.org`.

[17] M. Stadler. Publicly Verifiable Secret Sharing. In *EUROCRYPT*, pages 190–199, 1996.

[18] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice-Hall, 2nd edition, 1999.

[19] M. J. Wiener. Performance Comparison of Public-Key Cryptosystems. *RSA CryptoBytes*, 4(1):1–5, 1998.

[20] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. *IEEE Networks*, 13(6):24–30, 1999.