Self-securing Ad Hoc Wireless Networks

Haiyun Luo, Petros Zerfos, Jiejun Kong, Songwu Lu, Lixia Zhang
UCLA Computer Science Department
Los Angeles, CA 90095
{*hluo,pzerfos,jkong,slu,lixia*}*@cs.ucla.edu*

Abstract

Mobile ad hoc networking offers convenient infrastructureless communication over the shared wireless channel. However, the nature of ad hoc networks makes them vulnerable to security attacks. Examples of such attacks include passive eavesdropping over the wireless channel, denial of service attacks by malicious nodes and attacks from compromised *nodes or* stolen *devices. Unlike their wired counterpart, infrastructureless ad hoc networks do not have a clear line of defense, and every node must be prepared for encounters with an adversary. Therefore, a centralized or hierarchical network security solution does not work well.*

This work provides scalable, distributed authentication services in ad hoc networks. Our design takes a self-securing approach, in which multiple nodes (say, k) collaboratively provide authentication services for other nodes in the network. We first formalize a localized trust model that lays the foundation for the design. We further propose refined localized certification services based on our previous work, and develop a new scalable share update to resist more powerful adversaries. Finally, we evaluate the solution through simulation and implementation.

1. Introduction

Mobile ad hoc networking offers convenient infrastructureless communications over the shared wireless channel. A group of networking devices communicate among one another using wireless radios and operate by following a peer-to-peer network model. The nature of such a network makes them vulnerable to security attacks. Examples of attacks include passive eavesdropping over the wireless channel, denial of service attacks by malicious nodes and attacks from *compromised* entities or *stolen* devices. Unlike wired networks where an adversary must gain physical access to the wired link or sneak through security holes at firewalls and routers, wireless attacks may come from anywhere along all directions. The infrastructureless ad hoc network will not have a clear line of defense, and every node must be prepared for encounters with an adversary. Therefore, a centralized or hierarchical network security solution [1, 2] does not work well in mobile, ad hoc networks.

This work provides scalable, distributed authentication services in ad hoc networks. Two nodes[1] authenticate each other via signed, unforgeable certificates issued by a "virtual" trusted certification authority. Compared with common network authentication solutions [1, 2] that rely on physically present, trusted third-party (certification authority) server(s), our design takes a *self-securing* approach, in which multiple nodes (say, k) collaboratively serve as a certification authority server. Therefore, the authority and functionality of the authentication server are distributed to each node's locality. Any local k nodes are trusted as a whole and collaboratively provide authentication services.

Some nice features of our design are as follows. The system does not expose to any single point of compromise, single point of denial of service attack, or single point of failure. Authentication can be performed in every network neighborhood; this feature is important to authenticate roaming users in a mobile ad hoc network. Furthermore, our solution scales to large network size, and is robust against wireless channel errors.

This paper follows the design guidelines of [6] and makes several new contributions. We first formalize a localized trust model that lays the foundation for the design in Section 3, and then expand the adversary model that the system should handle. We further propose refined localized certification services (Section 5), and develop a new scalable solution of share updates to resist more powerful adversaries (Section 6). The new solution is evaluated through simulation and implementation (Section 7). We comment on several important design issues in Section 8 and conclude this paper in Section 9.

[1] For simplicity, we only handle node authentication in this work. The same design applies equally well in user authentication.

2. Related Works

Popular network authentication architectures include Kerberos [5] and the X.509 standard [1]. Two entities authenticate each other via a globally trusted certificate authority (CA). While this model gains great popularity in wired networks, it does not work well in large ad hoc wireless environments for several reasons: (a) Ad hoc networks provide no infrastructure support. The cost of maintaining such centralized servers may be prohibitively high. (b) Each of the CA servers is exposed to a single point of compromises and failures. (c) Multihop communications over the error-prone wireless channel expose data transmissions to high loss rate and large latency. Frequent route changes induced by mobility also make locating and contacting CA servers in a timely fashion non-trivial. Variations of the above model, such as hierarchical CAs and CA delegations [2] can ameliorate, but cannot addresses issues such as service availability and robustness [6].

PGP [3, 7] follows a "web-of-trust" authentication model. However, this approach does not scale beyond a relatively small community of trusted individuals. It would be difficult for each node to maintain a long list of trusted friends. Moreover, the members of a network may not even reach consensus on who is trusted and who is not, since independent "communities of trust webs" [3] may be formed as a by-product of this approach.

Security function sharing has been an active research area in the literature [8, 9, 10, 11, 12], where threshold secret sharing [13] serves as a basic primitive. Resilience against compromised nodes is enhanced by distributing the functionality of the centralized CA server among a *fixed* group of servers. Proactive secret sharing [14] can further improve robustness via periodic updates. However, the focus of these proposals is to maximize the security of the shared secret. They typically assume a small group of a few servers with rich network connectivity. Our scheme is inspired by these proposals, but extends the idea further to minimize the effort and complexity for mobile nodes to locate and contact the service providers in a dynamic multihop wireless network. There is no differentiation between servers and clients in our system: certification services are distributed into *every node* and a threshold number of nodes can collaboratively act as a server to provide certification services for other nodes.

There are several recent works on security in wireless networks. [15] proposes a Kerberos-based authentication scheme for mobile users in wireless cellular networks. [16] directly applies the threshold secret sharing and proactive secret share update techniques in a fixed group of "special nodes". Instead of following the conventional client-server model, we take a peer-to-peer approach to maximize service availability and facilitate localized communication.

3. System & Adversary Models

This work considers an ad hoc wireless network, where mobile nodes communicate with one another via the bandwidth-constrained, error-prone, and insecure wireless channel. We assume n mobile nodes, and n may be dynamically changing as mobile nodes join, leave, or fail over time. Besides, n is not constrained since there may be a large device population. The network provides neither physical nor logical infrastructure support, and the reliability of multihop packet forwarding based on underlying transport layer and ad hoc routing is not assured. We also make the following assumptions. (1) Each node has a unique nonzero ID and a mechanism to discover its one-hop neighbors. (2) Communication between one-hop neighboring node is more reliable compared with multihop communication over the error-prone wireless channel. (3) Each node has at least k one-hop legitimate neighboring nodes[2]. (4) Mobility is characterized by a maximum node moving speed S_{max}. (5) Each node is equipped with some local detection mechanism to identify misbehaving nodes among its one-hop neighborhood, e.g., those proposed in [4, 18]. This assumption is based on the observation that although intrusion detection in ad hoc networks is generally more difficult than in wired networks [4], detecting misbehaviors among one-hop neighbors is easier and practical due to the broadcast nature of the wireless transmission [18].

Our design handles two kinds of attacks: the DoS attacks and node break-ins. Adversaries may issue DoS attacks from various layers of the network stack ranging from network layer *Smurf* and *Teardrop*, transport layer *TCP flooding* and *SYN flooding*, and various attacks in application layer. For adversaries that seek to compromise networking nodes, we assume that the underlying cryptographic primitives such as RSA are computationally secure. However, we do allow occasional break-ins through factors such as insecure OS, software bugs and backdoors etc. Several adversaries may conspire to form a group. For ease of presentation, we denote such an adversary group by a single adversary. We characterize the adversaries in the following two models as proposed in [14]:

- Model I: During the entire lifetime of the network, the adversary cannot break into or control k or more nodes.
- Model II: Consider time being divided into intervals of length T. During any time interval T, the adversary cannot break into or control k or more nodes.

Although at any time constant it cannot break into or control k or more nodes, the adversary of model II can choose

[2] If a node could not find k neighbors, it may wait for new nodes coming in or roam to a new location for more neighbors.

its victims at each time interval. As time goes on each node in the network can be broken during some time interval. [6] handles the adversary of model I. In this paper, we extend with scalable parallel share update techniques to handle model II adversaries (Section 6).

4. The Architecture

This section presents our overall architecture. We first formalize a localized trust model, and then briefly introduce the certification design based on the *de facto* standard RSA.

4.1. Localized Trust Model

A well-defined trust model is fundamental in authentication protocols. In the dominant trusted third party (TTP) trust model [2], an entity is trusted only if it is verified by a central authority. While implementations of the TTP model possess efficiency and manageability properties in centralized systems, they suffer from scalability and robustness problems. In PGP's "web-of-trust" model [7], each entity manages its own trust based on direct recommendation. [17] seeks to further quantify the notions of trust and recommendation. To address the unique networking issues in in an ad hoc wireless network, we provide a localized trust model as follows.

In our localized trust model, an entity is trusted if any k trusted entities claim so within a certain time period T_{cert}. These k entities are typically among the entity's one-hop neighbors. Once a node is trusted by its local community, it is globally accepted as a trusted node. Otherwise, a locally distrusted entity is regarded as untrustworthy in the entire network. k and T_{cert} are two important parameters with T_{cert} characterizing the time-varying feature of a trust relationship [17].

Two options for setting k are as follows. The first is to set k as a globally fixed parameter that is honored by each entity in the system. In this case, k acts as a system-wide trust threshold. The second option is to set k as a location-dependent variable. For instance, k may be the majority of each node's neighboring nodes. This second option provides more flexibility to work in concert with diverse local network topology. However, there is no clear system-wide trust criterion. Due to lack of effective mechanisms to authoritatively determine a node's neighborhood in a mobile environment, the adversaries may take the advantage of this feature. In our design, we choose the first option with a network-wide fixed k that is tuned according to the network density and system robustness requirements. If a node could not find k neighbors in certain location, it may roam to meet more modes or wait for new nodes to move in. We will show how mobility helps a node to "accumulate" enough number of nodes in Section 5 and 7.2.1.

Trust management and maintenance are distributed in both space (k) and time (T_{cert}) domains in our localized trust model. This property is particularly appropriate for a large dynamic ad hoc wireless network, where centralized trust management would be difficult or expensive. Besides, a node indeed cares most the trustworthiness of its immediate neighbors in practice. This is because a node will communicate with the rest of the world via its one-hop neighbors.

4.2. Primitives

In an RSA-based design, the system CA's RSA key pair is denoted as $\{SK, PK\}$, where SK is the system private/secret key and PK is the system public key. SK is used to sign certificates for all nodes in the network. A certificate signed by SK can be verified by the well-known public key PK. By threshold secret sharing, SK is shared among network nodes. Each node 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, known or recoverable by any network node. We seek to preserve the secrecy of SK all the time.

Besides the system key pair, each node v_i also holds a personal RSA key pair $\{\bar{sk}_i, \bar{pk}_i\}$. To certify its personal keys, each node v_i holds the certificate $cert_i$ in the format of $< v_i, \bar{pk}_i, T >$, which reads as: "It is certified that the personal public key of v_i is $\bar{pk}_i$ during the time interval $[t, t+T]$. A certificate is valid only if it is signed by system secret key SK.

Our design makes extensive use of the polynomial secret sharing due to Shamir [13]. A secret, specifically the certificate-signing key SK, is shared among all n nodes in the network according to a random polynomial of order $k-1$. A coalition of k nodes with k polynomial shares can potentially recover SK by Lagrange interpolation, while no coalition up to $k-1$ nodes yields any information about SK. To further defend the polynomial secret sharing against the model II adversaries, Herzberg et al. proposed periodical secret share updates with different polynomials [14]. We extend this technique with scalable algorithms to further improve the the robustness of our system against adversaries of model II.

4.3. Overview

In our architecture, each node carries a certificate signed with SK. PK is assumed to be well-known for certificate verification. Nodes without valid certificates are denied from access to any network resources such as routing and packet forwarding. When a mobile node moves to a new location, it exchanges certificates with its new neighbors. Authenticated neighboring nodes help each other forward and route packets. They also monitor each other to detect possible break-ins. Specific monitoring mechanisms are left to each individual node's choice.

Certificates are stamped with expiration time. Nodes have to be issued a new certificate upon the expiration of its old certificate. In the centralized authentication architecture, nodes have to contact a CA server for this service. In our architecture, we distribute the certificate-signing key SK into *each node* of the network. Node v_i requests new certificate from any coalition of k nodes, typically among its one-hop neighbors. Upon the receipt of v_i's certification request, a node checks its records. If its record shows v_i as a well-behaving legitimate node, it returns a "partial" certificate by applying its share of SK. Otherwise the request is dropped. By collecting k partial certificates, v_i combines them together to generate the full new certificate as if it were from a CA server. A misbehaving or broken node that is detected by its neighbors will be unable to renew its certificate. It will be cut off from the network at the expiration of its current certificate. With the distributed CRL mechanisms proposed in this paper, a still-valid certificate can be revoked.

A valid certificate in our system represents the trust from a coalition of k nodes. Nodes with valid certificates are globally trusted. Each node contributes to the overall trust management and maintenance by monitoring and certifying its neighboring nodes. By this means, we realize the localized trust model as proposed in Section 4.1. By distributing certification services into each node's one-hop locality, we realize ubiquitous service availability for mobile nodes and robustness against DoS attacks.

5. Localized Certification Services

In this section, we present our localized certification services that include certificate issuing/renewal with dynamic coalescing, certificate revocation and distributed CRLs. The work in [6] proposed a k-bounded coalition offsetting technique to enable scalable distributed certificate generation. Node v_i firstly locates a coalition $\mathcal{B}$ of k neighbors $\{v_1, \cdots, v_k\}$ and broadcasts certification requests to them. A node $v_j \in \mathcal{B}$ checks its monitoring data on v_i to decide if certification service is granted. Upon receiving k partial certificates from coalition $\mathcal{B}$, node v_i multiplies them together to recover its full certificate.

There are two drawbacks in the above approach. Firstly, if any node in coalition $\mathcal{B}$ fails to respond due to node failures or moving out of range, all the other partial certificates become useless. The computation of other nodes are all wasted and v_i has to restart the whole process from the very beginning. We present an optimization called *dynamic coalescing* to solve the problem in Section 5.1. Our optimization is based on the observation that the coalition can be formulated dynamically from any k responding nodes, instead of being specified by v_i a priori.

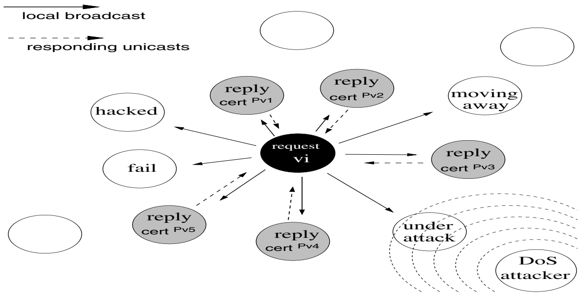

Figure 1. Dynamic Coalescing

The second drawback is that when node v_j receives a certification request from v_i, its records may not provide enough information on v_i. It may be because the interaction between v_i and v_j does not last long enough. Moreover, v_i may not exist in v_j's records at all if they just met. v_j has two options in this scenario. One is to serve v_i's request, since no bad records are located. The risk is that a roaming adversary who cannot get a new certificate from his previous location may take the advantage. The other option is to drop the request, since no records can demonstrate v_i well-behaving. But a legitimate mobile node may not be able to get a new certificate. [6] took the second policy to handle roaming adversaries. The cost is the limitation on node mobility: a mobile node has to prepare a certificate with enough long validity time before he moves. In this paper we take the first approach to enable free node mobility. Roaming adversaries are handled with the distributed certificate revocation mechanisms presented in Section 5.2.

5.1. Certificate Issuing/Renewal with Dynamic Coalescing

Each node v in the network holds a polynomial share P_v of the certificate signing exponent SK according to a random polynomial $f(x)$ s.t. $P_v = f(v)$. $f(x) = SK + \sum_{j=1}^{k-1} f_j x^j$, where $f_1, \cdots, f_{k-1}$ are uniformly distributed over a finite filed. Node v_i broadcasts its request for a new certificate, without specifying the coalition. A neighboring node v_j that receives the request and decides to serve the request will return a partial certificate $CERT_{v_j}$ by directly applying its polynomial shares on the certificate statement $cert$ as $CERT_{v_j} = (cert)^{P_{v_j}} \bmod N$. Upon receiving at least k such partial certificates, node v_i picks k to form the coalition $\mathcal{B}$. Without loss of generality, suppose v_i chooses $\{CERT_{v_1}, \cdots, CERT_{v_k}\}$. Node v_i then converts each of them according to the IDs of these k responding nodes:

$$CERT'_{v_j} = \left(CERT_{v_j}\right)^{l_{v_j}(0)} \quad \bmod N$$

where $l_{v_j}(0) = \prod_{r=1,r\neq j}^{k} \frac{v_r}{v_r - v_j} \bmod N$. v_i then multiplies them together to generate the candidate certificate as $CERT' = \prod_{r=1}^{k} CERT'_{v_j} \bmod N$. Finally node v_i employs the k-bounded coalition offsetting algorithm [6] to recover its new certificate $CERT$ from $CERT'$.

The computation complexity for each serving nodes is $O(1)$. For the requesting node the computation complexity is still $O(k)$, but the actual load is doubled compared with [6]. Because it consists of a single round of localized communication: one broadcast request and k unicast responses (Figure 1), our protocol has minimum requirements on the reliability of the underlying wireless channel. As long as k neighbors respond, other neighbors are free to move or fail; additional responses may be discarded.

5.2. Certificate Revocation and CRL

The records that v_j maintains consist of two parts: its direct monitoring data on neighboring nodes, and a certificate revocation list (CRL). Each entry of the CRL is composed of a node ID and a list of the node's accusers. If a node's accuser list contains less than k legitimate accusers, the node is marked as "suspect". Otherwise, the node is determined by v_j to be misbehaving or broken and marked as "convicted". We choose the threshold that convicts a node as k to ensure a legitimate node not be convicted by malicious accusations from an adversary.

In two scenarios a node is marked "convicted". When by direct monitoring v_j determines one of its neighboring nodes to be misbehaving or broken, v_j puts the node into its CRL and directly marks the node "convicted". In this scenario v_j also floods a signed accusation against the node. The range of the flooding is studied below. The other scenario happens when v_j receives an accusation against some node. It firstly checks if the accuser is a convicted node in its CRL. If it is, the accusation is concluded to be malicious and dropped. If not, v_j updates its CRL entry of the accused node by adding the accuser into the node's accuser list. The accused node will be marked "convicted" if the number of accusers reaches k. When a node is convicted, v_j delete the node from all accuser lists. A convicted node will be marked "suspect" if its number of accusers drops below k.

The range of the accusation propagation is an important design parameter. A large range causes excessive communication overhead, while a small range may not be enough to cover a roaming adversary. The practical scheme for controlled flooding is by setting the TTL (time-to-live) field in the IP headers of the accusation packets. One way to set TTL is based on the certificate validity period T_{cert}, the one-hop wireless transmission distance D, and our assumption on maximum node moving speed S_{max}. In a uniformly distributed network, to ensure a misbehaving node or a compromised node cannot escape the area of accusation before the expiration of its current certificate, the TTL should be set as $TTL \geq \lceil \frac{T_{cert} \cdot 2S_{max}}{D} \rceil$. If TTL is set to m, the nodes whose accusations reach v_j must be at most m hops away. Therefore v_j's CRL contains nodes at most $m+1$ hops away. To further decrease the CRL complexity, T_{cert} after an entry's last update, v_j can remove it from its CRL. The reason is that after T_{cert} a convicted node should have its certificate expired, and thus be cut off from the network. v_j holds each CRL entry for T_{cert} so that it will not serve a convicted node that carries still-valid certificate.

In our design, CRL is constrained in both space domain and time domain. It is built and maintained *on-demand*, and stored *locally*. These properties comply with the overall scalability and robustness of our architecture, and the ad hoc nature of the network.

6. Parallel Share Updates

Our localized certification so far is robust against the adversaries of model I as defined in Section 3. To handle stronger model II adversaries, Herzberg et al. proposed periodical updates of each node's secret share [14]. However, the proposed protocol in the proactive secret sharing context requires each node collect inputs from all other nodes to finalize its update. It is therefore not applicable in our scenario for several reasons: (a) The solution is not scalable. In an ad hoc wireless network, a node cannot afford to maintain updated global knowledge of the dynamic network membership; (b) The communication overhead is too high for wireless channel; (c) These proposals require a global broadcast channel, which does not exist in typical ad hoc wireless networks.

In this section, we propose two approaches to achieve scalable and efficient share update in ad hoc wireless networks. The first approach is a simple sequential process based on the self-initialization as presented in [6]. Firstly a coalition of k nodes update their shares by applying the existing protocols as proposed in [14]. The self-initialization protocols then follow to update the shares of the rest of the network. The second approach features parallel share updates over the network for fast convergence. We present the parallel share update mechanism as follows.

Similar to [14], we divide time into periods. Each time period is composed of a share update phase and an operational phase. At the beginning of the share update phases, a chosen coalition of k nodes in the system collaboratively generate a random share update polynomial $f_u(x) = f_{u,1}x + \cdots + f_{u,k-1}x^{k-1}$ where $f_u(0) = 0$. f_u is then encrypted by PK for privacy against adversaries. The coalition then collaboratively apply their polynomial shares of SK to sign the encrypted f_u. This signature prevents an adversary of model II from emulating a coalition of k nodes to fake share updates. The encrypted polynomial, together with its signature, is then propagated in the network by flooding. Once a node receives the encrypted update polynomial, it requests share-update service from k neighboring nodes to evaluate its $P_{u,v_i} = f_u(v_i)$. Note that as long as they apply the same version of shares, these k nodes can serve the request without having their share updated. The process is composed of the following three steps.

Collaborative generation of the update polynomial f_u. At the beginning of each update phase, each node initiates updates with probability $1/\hat{n}$,where $\hat{n}$ is an estimate on the total number of networking nodes. This ensures that statistically there is only one node to initiate the update process. Once a node v_i decides to initiate the update, it locates a coalition of k neighbors and collaboratively generate the encrypted update polynomial $(f_u)_{PK}$ and a signature.

Robust propagation of the update polynomial. Node v_i floods the encrypted update polynomial $(f_u)_{PK}$ with the signature in the network. We take the advantage of the robustness of the flooding protocol in connectivity-redundant ad hoc networks, to ensure that each node receives the update polynomial at least once.

Distributed evaluation of share update P_{u,v_i}. Since the propagated f_u is encrypted by the system PK, each node v_i solicits its k neighbors to collaboratively evaluate $P_{u,v_i} = f_u(v_i)$ for it. Similar to the localized certification services, each of these k neighboring nodes returns a partial share update. Node v_i adds these k partial updates together to recover P_{u,v_i}. Node v_i then updates its share and erases its old share at the end of the update phase.

Our design is (k,n)-secure [19] in the sense that given up to $k-1$ shares of SK and a history of polynomial many partial results, an adversary learns no more about SK than without these information. Due to lack of space, we leave the detailed algorithms, communication protocols and proof to the technical report [20].

7. Performance Evaluation

We implemented our design in both network simulator *ns-2* and UNIX platform. The Unix implementation is used to evaluate the computational cost, and the network simulator helps us to evaluate the communication aspects of our protocols, such as scalability, service availability, mobility and channel error in ad hoc wireless networks.

7.1. Evaluation on Computation Aspects

We test our system on machines with three different SPECint_rate95 values: 10, 107 and 202 [23], which represent the processing power of a Pentium 75, PentiumII 300, and PentiumIII 500 respectively. Our measurements show that computation power is a critical factor of the performance. For example, a mobile laptop computer with a PentiumIII 500Mhz CPU performs well in all test cases, while a low-end computer with a Pentium 75 CPU requires about 5 sec for certification service with key length 1024 bits and coalition size $k \leq 10$. The parallel share update needs more time to complete: almost 80 sec for the low-end device. However, since in practice share updates are not very frequent (once several hours/days), we can tolerate that amount of delay.

From figure 2[3], we observe that the standard RSA signing is almost 2.5 times faster than the partial certificate computation (PCC). This is because standard RSA signing uses a major optimization technique [22], which is not applicable without the knowledge of the two prime factors of the RSA modulo. From the same figure, we also observe that a short personal key (e.g. 768 bits for low-end devices) can reduce the processing delay to 1.3 sec. Figure 3 shows how the RSA key length affects the processing delay of the parallel share update. If we choose the key length to be 1024 bits, a Pocket PC needs 24 sec to finish a parallel share update.

Figures 4 and 5 show how k affects processing delay. In figure 4 we can see that k has a small impact on the processing delay of certification services. This is because the multiplication of k partial certificates is an inexpensive operation even when k is relatively large. However, k greatly affects the computation of share updates. From figure 5, we observe that the processing delay grows linearly with k. This is because each node in the k coalition has to process $k-1$ PK-encrypted polynomial coefficients.

7.2. Evaluation on Communication Aspects

We use four metrics to evaluate the performance of our communication protocols. *Success Ratio* measures the ratio of the number of successful certification services over the number of requests. *Average Delay* measures the average latency for a node to request a certification service. *Overhead* measures the communication overhead in bytes. *Convergence time* measures the time required to complete a share update. We run simulation in networks with 30 to 100 nodes. Node moving speed varies from 1 m/sec to 20 m/sec. We apply the random way-point model to emulate node mobility patterns. The certificate expiration is set to 5 time units. Parameter k is set to 5 by default, except for the topologies that consist of 30 nodes where k is set to 3.

7.2.1 Certification services

Figure 6 shows the success ratio of the certification services as the network size grows from 30 to 100 nodes. From the

[3] In the performance diagrams, *Speed* indicates SPECint_rate95 hardware value. *RSA* stands for a standard RSA SK-signing operation. *PCC* denotes the partial certificate computation. *Combine* stands for multiplicative combination plus k-bounded coalition offsetting. *PSSU* denotes a single node's processing delay for share update.

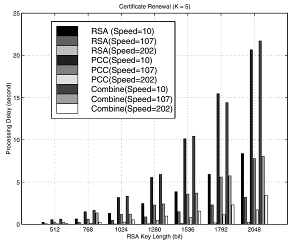

Figure 2. Cert. Service: RSA Key Length

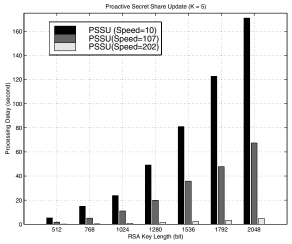

Figure 3. Share Update: RSA Key Length

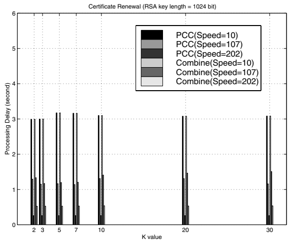

Figure 4. Cert. Service: k

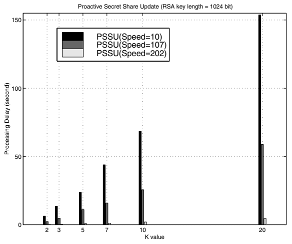

Figure 5. Share Update: k

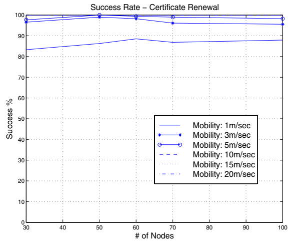

Figure 6. Cert. Service: Success Ratio vs. Node #

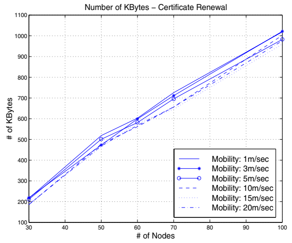

Figure 7. Cert. Service: Bytes vs. Node #

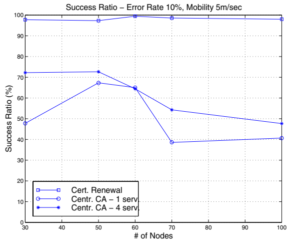

Figure 8. Success Ratio vs. Node #, Error Rate 10 %

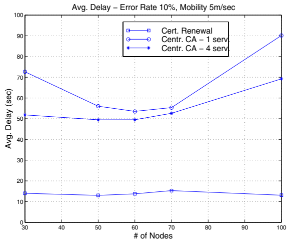

Figure 9. Avg. Delay vs. Node #, Error Rate 10 %

figure, we observe that the success ratio is always over 95% and in most cases 100%. As the node speed increases from 1 m/sec to 20 m/sec, we observe that the success ratio almost remains unchanged. For the case of 1 m/sec, the success ratio is around 95%. The main reason is that under low node density and low mobility a node may have difficulty in locating k neighboring nodes. Figure 7 shows that the total communication overhead grows linearly as the number of nodes increases, i.e., the per node overhead remains fixed. Moreover, the communication overhead is almost independent of the node moving speed.

The impact of wireless channel errors on our protocols is shown in figures 8 and 9. Node mobility is set to 5 m/sec and the channel error is set to 10%. From these figures, we see that our localized design is robust against the channel error. However, the performance of both the single CA and 4-CA centralized approaches degrades significantly.

7.2.2 Parallel share update

Although the secret share update protocol may not be executed as often as that of certificate renewal, its ability to scale to the network size and resilience against the node mobility are critical to ensure the secret share consistency. Figure 10 shows the average delay experienced by each node while updating its secret share. We observe that the delay remains almost constant as the number of nodes in the network increases, and mobility has little impact on it even at the high speed of 15m/sec. Figure 11 shows the time required for the share update process to converge. From the figure we can see that the convergence time is almost independent of the network size and node moving speed as expected. These results demonstrate the scalability to the network size and the resilience against node mobility of our design.

Figures 12-13 expose more details of the share update process. In these figures the y-axis represents the percentage of nodes that manage to update their shares within the time on the x-axis. They show the cumulative distribution of the update process. We use the network of 60 and 100 nodes, and the node moving speed from 3 to 15 m/sec. From these figures we can see that as the size of the network grows, the impact of mobility decreases. This is a great improvement compared with the solution that we proposed in our previous work [6], where the convergence time increases linearly as the network scale grows.

8. Discussions

We now come back to discuss several important issues.

Obtaining initial certificates Any new node needs an initial certificate before it can join the network. Moreover, an admitted node has to bear a valid certificate when it requests its certificate to be renewed. Our localized certification never *creates* or issues a brand-new certificate. This policy is to prevent malicious node to have multiple certificates based by forged or stolen IDs.

How to issue initial certificates poses the root of trust problem. A node may be issued an initial certificate by an offline authority through external means (e.g., in-person ID). Alternatively, we may use any coalition of k networking nodes to issue an initial certificate via collaborative ad-

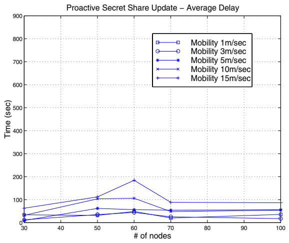

Figure 10. Parallel Share Upd.: Avg. Delay vs. Node #

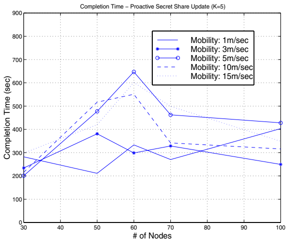

Figure 11. Parallel Share Upd.: Convergence Time vs. Node #

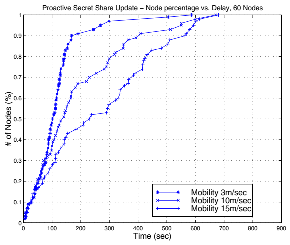

Figure 12. Parallel Share Upd.: Node percentage vs. Delay, 60 Nodes

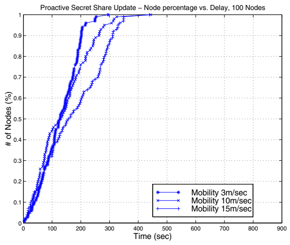

Figure 13. Parallel Share Upd.: Node percentage vs. Delay, 100 Nodes

mission control for this new node. The admission control policy has to be consistent with the robustness of the overall trust model, system model and the adversary models.

Bootstrapping of the first k nodes To initialize the very first k nodes, we assume an offline authority who knows the full certificate signing key SK and the associated polynomial $f(x)$ of degree $k-1$. Note that this is the standard assumption of related works on secret sharing [14, 9, 21].

Parameter k Revisited Our design so far assumes each node to have at least k legitimate neighbors. This assumption is critical for certification services to be robust against the adversaries defined in Section 3. The parameter k also determines the availability of our services. In our current design, these three factors are coupled and represented by a single parameter k. This coupling effect reduces the flexibility of our system. How to decouple these three aspects poses new challenges for future research.

Intrusion Detection in Ad Hoc Networks As presented in our system model (Section 3), we assume each node is equipped with some local detection mechanism to identify misbehaving nodes among its one-hop neighborhood. While we admit that this might seem a strong assumption, some initial progress has been reported recently [18]. We believe that as time goes, better *local* intrusion detection mechanism will be available to serve our purpose.

9. Conclusions

This paper describes a self-securing approach to node authentication in mobile, ad hoc wireless networks. To this end, we propose a localized trust model, together with its realization to address networking issues of node mobility, network dynamics and wireless channel errors. A scalable share update scheme is developed, and several optimization techniques that greatly enhance the efficiency and robustness of our algorithms and protocols are proposed. Through localized design, we ensure the scalability of our architecture to facilitate practical deployment in a potentially large-scale network with dynamic node membership.

References

[1] A. Aresenault and S. Turner, “Internet X.509 public key infrastructure,” *draft-ietf-pkix-roadmap-06.txt*, 2000

[2] R. Perlman, “An overview of PKI trust models”, *IEEE Network*, p. 38-43, vol.13, (no.6) Nov.-Dec. 1999

[3] A. Abdul-Rahman, “The PGP trust model,” *EDI-Forum: the Journal of Electronic Commerce*, Apr. 1997

[4] Y. Zhang and W. Lee, “Intrusion detection in wireless ad hoc networks,” *ACM MOBICOM*, 2000

[5] J. Kohl and B. Neuman, “The Kerberos network authentication service (version 5),” *RFC-1510*

[6] J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang, “Providing robust and ubiquitous security support for MANET,” *IEEE ICNP* 2001, 2001.

[7] S. Garfinkel, “PGP: Pretty Good Privacy,” *O'Reilly & Associates Inc.*, USA, 1995

[8] L. Gong, “Increasing availability and security of an authentication service,” *IEEE Journal on Selected Areas in Communications*, Vol.11, No.5, Jun. 1993

[9] Y. Frankel, P. Gemmell, P. Mackenzie and M. Yung, “Proactive RSA,” *CRYPTO*, 1997

[10] Y. Frankel, P. Gemmel, P. MacKenzie, M. Yung, “Optimal-resilience proactive public-key cryptosystems,” *FOCS '97*, 1997.

[11] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, “Robust and efficient sharing of RSA functions,” *Journal of Cryptology*, 1996

[12] Y. Frankel and Y. G. Desmedt. “Parallel reliable threshold multisignature,” *Technical Report TR-92-04-02*, Dept. of EECS, University of Wisconsin-Milwaukee, 1992

[13] A. Shamir, “How to share a secret,” *Communications of ACM*, 1979

[14] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, “Proactive secret sharing,” *Extended abstract*, 1995

[15] A. Fox and S. Gribble, “Security on the move: Indirect authentication using Kerberos,” *ACM MOBICOM*, 1996

[16] L. Zhou and Z. J. Haas. “Securing ad hoc networks,” *IEEE Networks*, 13(6):24–30, 1999

[17] A. Abdul-Rahman and S. Hailes, “A distributed trust model”, *ACM New Security Paradigms Workshop*, 1997

[18] S. Marti, T. Giuli, K. Lai and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks,” *ACM MOBICOM*, 2000

[19] A. D. Santis, Y. Desmedt, Y. Frankel and M. Yung, “How to share a function securely,” *STOC* 1994

[20] H. Luo and S. Lu, “Ubiquitous and robust authentication services for ad hoc wireless networks,” *UCLA Computer Science Technical Report 200030*, Oct. 2000

[21] R. Canetti, S. Halevi, and A. Herzberg, “Maintaining authenticated communication in the presence of break-ins,” *Journal of Cryptology*, 13(1):61-105, 2000

[22] A. Menezes, P. Oorschot and S. Vanstone, “Handbook of applied cryptography,” CRC Press, 1996

[23] Standard Performance Evaluation Corporation, *http://www.specbench.org*

[24] A. K. Lenstra and E. R. Verheul. “Selecting cryptographic key sizes,” *Public Key Cryptography*, pp. 446–465, 2000