Securing Ad Hoc Networks[*]

Lidong Zhou
Department of Computer Science

Zygmunt J. Haas
School of Electrical Engineering

Cornell University
Ithaca, NY 14853

Abstract

Ad hoc networks are a new wireless networking paradigm for mobile hosts. Unlike traditional mobile wireless networks, ad hoc networks do not rely on any fixed infrastructure. Instead, hosts rely on each other to keep the network connected. The military tactical and other security-sensitive operations are still the main applications of ad hoc networks, although there is a trend to adopt ad hoc networks for commercial uses due to their unique properties. One main challenge in design of these networks is their vulnerability to security attacks. In this paper, we study the threats an ad hoc network faces and the security goals to be achieved. We identify the new challenges and opportunities posed by this new networking environment and explore new approaches to secure its communication. In particular, we take advantage of the inherent redundancy in ad hoc networks — multiple routes between nodes — to defend routing against denial of service attacks. We also use replication and new cryptographic schemes, such as threshold cryptography, to build a highly secure and highly available key management service, which forms the core of our security framework.

1 Introduction

Ad hoc networks are a new paradigm of wireless communication for mobile hosts (which we call *nodes*). In an ad hoc network, there is no fixed infrastructure such as base stations or mobile switching centers. Mobile nodes that are within each other's radio range communicate directly via wireless links, while those that are far apart rely on other nodes to relay messages as routers. Node mobility in an ad hoc network causes frequent changes of the network topology. Figure 1 shows such an example: initially, nodes A and D have a direct link between them. When D moves out of A's radio range, the link is broken. However, the network is still connected, because A can reach D through C, E, and F.

Military tactical operations are still the main application of ad hoc networks today. For example, military units (e.g., soldiers, tanks, or planes), equipped with wireless communication devices, could form an ad hoc network when they roam in a battlefield. Ad hoc networks can also be used for emergency, law enforcement, and rescue missions. Since an ad hoc network can be deployed rapidly with relatively low cost, it becomes an attractive option for commercial uses such as sensor networks or virtual classrooms.

1.1 Security goals

Security is an important issue for ad hoc networks, especially for those security-sensitive applications. To secure an ad hoc network, we consider the following attributes: *availability*, *confidentiality*, *integrity*, *authentication*, and *non-repudiation*.

[*] Supported in part by ARPA/RADC grant F30602-96-1-0317, AFOSR grant F49620-94-1-0198, Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-99-1-0533, and National Science Foundation Grants 9703470, ANI-9805094, and NCR-9704404. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of these organizations or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotation thereon.

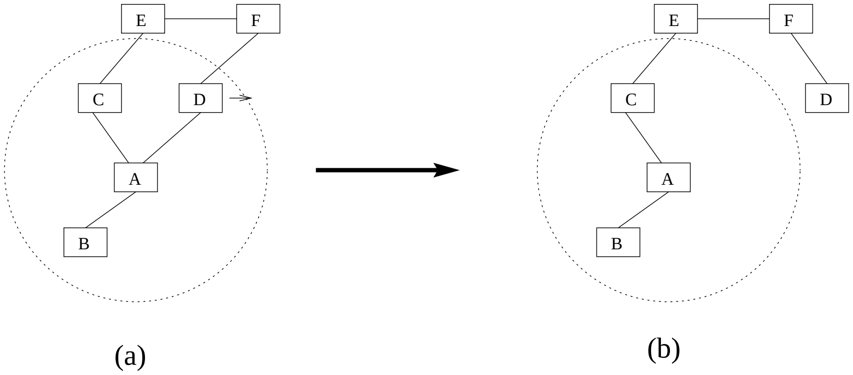

Figure 1: Topology change in ad hoc networks: nodes A, B, C, D, E, and F constitute an ad hoc network. The circle represents the radio range of node A. The network initially has the topology in (a). When node D moves out of the radio range of A, the network topology changes to the one in (b).

Availability ensures the survivability of network services despite denial of service attacks. A denial of service attack could be launched at any layer of an ad hoc network. On the physical and media access control layers, an adversary could employ jamming to interfere with communication on physical channels. On the network layer, an adversary could disrupt the routing protocol and disconnect the network. On the higher layers, an adversary could bring down high-level services. One such target is the key management service, an essential service for any security framework.

Confidentiality ensures that certain information is never disclosed to unauthorized entities. Network transmission of sensitive information, such as strategic or tactical military information, requires confidentiality. Leakage of such information to enemies could have devastating consequences. Routing information must also remain confidential in certain cases, because the information might be valuable for enemies to identify and to locate their targets in a battlefield.

Integrity guarantees that a message being transferred is never corrupted. A message could be corrupted because of benign failures, such as radio propagation impairment, or because of malicious attacks on the network.

Authentication enables a node to ensure the identity of the peer node it is communicating with. Without authentication, an adversary could masquerade a node, thus gaining unauthorized access to resource and sensitive information and interfering with the operation of other nodes.

Finally, *non-repudiation* ensures that the origin of a message cannot deny having sent the message. Non-repudiation is useful for detection and isolation of compromised nodes. When a node A receives an erroneous message from a node B, non-repudiation allows A to accuse B using this message and to convince other nodes that B is compromised.

There are other security goals (e.g., authorization) that are of concern to certain applications, but we will not pursue these issues in this paper.

1.2 Challenges

The salient features of ad hoc networks pose both challenges and opportunities in achieving these security goals.

First, use of wireless links renders an ad hoc network susceptible to link attacks ranging from passive eavesdropping to active impersonation, message replay, and message distortion. Eavesdropping might give an adversary access to secret information, violating confidentiality. Active attacks might allow the adversary to delete messages, to inject erroneous messages, to modify messages, and to impersonate a node, thus violating availability, integrity, authentication, and non-repudiation.

Secondly, nodes, roaming in a hostile environment (e.g., a battlefield) with relatively poor physical protection, have non-negligible probability of being compromised. Therefore, we should not only consider malicious attacks from outside a network, but also take into account the attacks launched from within the network by compromised nodes. Therefore, to achieve high survivability, ad hoc networks should have a distributed architecture with no central entities. Introducing any central entity into our security solution could lead to significant vulnerability; that is, if this centralized entity is compromised, then the entire network is subverted.

Thirdly, an ad hoc network is dynamic because of frequent changes in both its topology and its membership (i.e., nodes frequently join and leave the network). Trust relationship among nodes also changes, for example, when certain nodes are detected as being compromised. Unlike other wireless mobile networks, such as mobile IP [21, 48, 34], nodes in an ad hoc network may dynamically become affiliated with administrative domains. Any security solution with a static configuration would not suffice. It is desirable for our security mechanisms to adapt on-the-fly to these changes.

Finally, an ad hoc network may consist of hundreds or even thousands of nodes. Security mechanisms should be scalable to handle such a large network.

1.3 Scope and roadmap

Traditional security mechanisms, such as authentication protocols, digital signature, and encryption, still play important roles in achieving confidentiality, integrity, authentication, and non-repudiation of communication in ad hoc networks. However, these mechanisms are not sufficient by themselves.

We further rely on the following two principles. First, we take advantage of *redundancies* in the network topology (i.e., multiple routes between nodes) to achieve availability. The second principle is *distribution of trust*. Although no single node is trustworthy in an ad hoc network because of low physical security and availability, we can distribute trust to an aggregation of nodes. Assuming that any $t+1$ nodes will unlikely be all compromised, consensus of at least $t+1$ nodes is trustworthy.

In this paper, we will not address denial of service attacks towards the physical and data link layers. Certain physical layer countermeasures such as spread spectrum have been extensively studied (e.g., [44, 6, 42, 17, 37]). However, we do focus on how to defend against denial of service attacks towards routing protocols in Section 2.

All key-based cryptographic schemes (e.g., digital signature) demand a key management service, which is responsible for keeping track of bindings between keys and nodes and for assisting the establishment of mutual trust and secure communication between nodes. We will focus our discussion in Section 3 on how to establish such a key management service that is appropriate for ad hoc networks. We present related work in Section 4 and conclude in Section 5.

2 Secure Routing

To achieve availability, routing protocols should be robust against both dynamically changing topology and malicious attacks. Routing protocols [30, 25, 43, 32, 49, 16, 23, 35] proposed for ad hoc networks cope well with the dynamically changing topology. However, none of them, to our knowledge, have accommodated mechanisms to defend against malicious attacks. Routing protocols for ad hoc networks are still under active research. There is no single standard routing protocol. Therefore, we aim to capture the common security threats and to provide guidelines to secure routing protocols.

In most routing protocols, routers exchange information on the topology of the network in order to establish routes between nodes. Such information could become a target for malicious adversaries who intend to bring the network down.

There are two sources of threats to routing protocols. The first comes from external attackers. By injecting erroneous routing information, replaying old routing information, or distorting routing information, an attacker could successfully partition a network or introduce excessive traffic load into the network by causing retransmission and inefficient routing.

The second and also the more severe kind of threats comes from compromised nodes, which might advertise incorrect routing information to other nodes. Detection of such incorrect information is difficult: merely requiring routing information to be signed by each node would not work, because compromised nodes are able to generate valid signatures using their private keys.

To defend against the first kind of threats, nodes can protect routing information in the same way they protect data traffic, i.e., through the use of cryptographic schemes such as digital signature. However, this defense is ineffective against attacks from compromised servers. Worse yet, as we have argued, we cannot neglect the possibility of nodes being compromised in an ad hoc network. Detection of compromised nodes through routing information is also difficult in an ad hoc network because of its dynamically changing topology: when a piece of routing information is found invalid, the information could be generated by a compromised node, or, it could have become invalid as a result of topology changes. It is difficult to distinguish between the two cases.

On the other hand, we can exploit certain properties of ad hoc networks to achieve secure routing. Note that routing protocols for ad hoc networks must handle outdated routing information to accommodate the dynamically changing topology. False routing information generated by compromised nodes could, to some extent, be considered outdated information. As long as there are sufficiently many correct nodes, the routing protocol should be able to find routes that go around these compromised nodes. Such capability of the routing protocols usually relies on the inherent redundancies — multiple, possibly disjoint, routes between nodes — in ad hoc networks. If routing protocols can discover multiple routes (e.g., protocols in *ZRP* [16], *DSR* [25], *TORA* [32], and *AODV* [35] all can achieve this), nodes can switch to an alternative route when the primary route appears to have failed.

Diversity coding [1] takes advantage of multiple paths in an efficient way without message retransmission. The basic idea is to transmit redundant information through additional routes for error detection and correction. For example, if there are n disjoint routes between two nodes, then we can use $n - r$ channels to transmit data and use the other r channels to transmit redundant information. Even if certain routes are compromised, the receiver may still be able to validate messages and to recover messages from errors using the redundant information from the additional r channels.

3 Key Management Service

We employ cryptographic schemes, such as digital signatures, to protect both routing information and data traffic. Use of such schemes usually requires a key management service.

We adopt a public key infrastructure because of its superiority in distributing keys and in achieving integrity and non-repudiation. Efficient secret key schemes are used to secure further communication after nodes authenticate each other and establish a shared secret session key.

In a public key infrastructure, each node has a public/private key pair. Public keys can be distributed to other nodes, while private keys should be kept confidential to individual nodes. There is a trusted entity called *Certification Authority* (*CA*) [11, 47, 26] for key management. The *CA* has a public/private key pair, with its public key known to every node, and signs certificates binding public keys to nodes.

The trusted *CA* has to stay on-line to reflect the current bindings, because the bindings could change over time: a public key should be revoked if the owner node is no longer trusted or is out of the network; a node may refresh its key pair periodically to reduce the chance of a successful brute-force attack on its private key.

It is problematic to establish a key management service using a single *CA* in ad hoc networks. The *CA*, responsible for the security of the entire network, is a vulnerable point of the network: if the *CA* is unavailable, nodes cannot get the current public keys of other nodes or to establish secure communication with others. If the *CA* is compromised and leaks its private key to an adversary, the adversary can then sign any erroneous certificate using this private key to impersonate any node or to revoke any certificate.

A standard approach to improve availability of a service is replication. But a naive replication of the *CA* makes the service more vulnerable: compromise of any single replica, which possesses the service private key, could lead to collapse of the entire system. To solve this problem, we distribute the trust to a set of nodes by letting these nodes share the key management responsibility.

3.1 System model

Our key management service is applicable to an asynchronous ad hoc network; that is, a network with no bound on message-delivery and message-processing times. We also assume that the underlying network layer

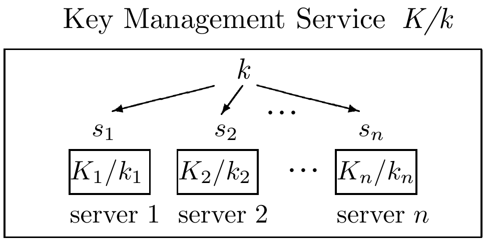

Figure 2: The configuration of a key management service: the key management service consists of n servers. The service, as a whole, has a public/private key pair K/k. The public key K is known to all nodes in the network, whereas the private key k is divided into n shares $s_1, s_2, \ldots, s_n$, one share for each server. Each server i also has a public/private key pair K_i/k_i and knows the public keys of all nodes.

provides reliable links[i]. The service, as a whole, has a public/private key pair. All nodes in the system know the public key of the service and trust any certificates signed using the corresponding private key. Nodes, as clients, can submit *query* requests to get other clients' public keys or submit *update* requests to change their own public keys.

Internally, our key management service, with an $(n, t+1)$ configuration ($n \geq 3t+1$), consists of n special nodes, which we call *servers*, present within an ad hoc network. Each server also has its own key pair and stores the public keys of all the nodes in the network. In particular, each server knows the public keys of other servers. Thus, servers can establish secure links among them. We assume that the adversary can compromise up to t servers in any period of time with a certain duration[ii].

If a server is compromised, then the adversary has access to all the secret information stored on the server. A compromised server might be unavailable or exhibit Byzantine behavior (i.e., it can deviate arbitrarily from its protocols). We also assume that the adversary lacks the computational power to break the cryptographic schemes we employ.

The service is correct if the following two conditions hold:

- (**Robustness**) The service is always able to process *query* and *update* requests from clients. Every *query* always returns the last updated public key associated with the requested client, assuming no concurrent *updates* on this entry.

- (**Confidentiality**) The private key of the service is never disclosed to an adversary. Thus, an adversary is never able to issue certificates, signed by the service private key, for erroneous bindings.

3.2 Threshold cryptography

Distribution of trust in our key management service is accomplished using *threshold cryptography* [4, 3]. An $(n, t+1)$ threshold cryptography scheme allows n parties to share the ability to perform a cryptographic operation (e.g., creating a digital signature), so that any $t+1$ parties can perform this operation jointly, whereas it is infeasible for at most t parties to do so, even by collusion.

In our case, the n servers of the key management service share the ability to sign certificates. For the service to tolerate t compromised servers, we employ an $(n, t+1)$ threshold cryptography scheme and divide the private key k of the service into n shares $(s_1, s_2, \ldots, s_n)$, assigning one share to each server. We call $(s_1, s_2, \ldots, s_n)$ an $(n, t+1)$ *sharing* of k. Figure 2 illustrates how the service is configured.

For the service to sign a certificate, each server generates a partial signature for the certificate using its private key share and submits the partial signature to a combiner.[iii] With $t+1$ correct partial signatures,

[i] Our key management service actually works under a much weaker link assumption, which is more appropriate for ad hoc networks. We leave such details to a separate paper currently in preparation.

[ii] The duration depends on how often and how fast share refreshing (Section 3.3) is done.

[iii] Any server can be a combiner. No extra information about k is disclosed to a combiner. To make sure that a compromised combiner cannot prevent a signature from being computed, we can use $t+1$ servers as combiners to ensure that at least one combiner is correct and is able to compute the signature.

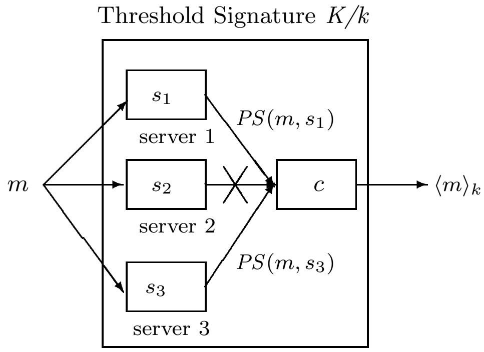

Figure 3: Threshold signature: given a service consisting of 3 servers. Let K/k be the public/private key pair of the service. Using a $(3, 2)$ threshold cryptography scheme, each server i gets a share s_i of the private key k. For a message m, server i can generate a partial signature $PS(m, s_i)$ using its share s_i. Correct servers 1 and 3 both generate partial signatures and forward the signatures to a combiner c. Even though server 2 fails to submit a partial signature, c is able to generate the signature $\langle m \rangle_k$ of m signed by service private key k.

the combiner is able to compute the signature for the certificate. However, compromised servers (there are at most t of them) cannot generate correctly signed certificates by themselves, because they can generate at most t partial signatures. Figure 3 shows how servers generate a signature using a $(3, 2)$ threshold signature scheme.

When applying threshold cryptography, we must defend against compromised servers. For example, a compromised server could generate an incorrect partial signature. Use of this partial signature would yield an invalid signature. Fortunately, a combiner can verify the validity of a computed signature using the service public key. In case verification fails, the combiner tries another set of $t + 1$ partial signatures. This process continues until the combiner constructs the correct signature from $t + 1$ correct partial signatures. More efficient robust combining schemes are proposed [13, 12]. These schemes exploit the inherent redundancies in the partial signatures (note that any $t+1$ correct partial signatures contain all the information of the final signature) and use error correction codes to mask incorrect partial signatures. In [13], a robust threshold *DSS* (Digital Signature Standard) scheme is proposed. The process of computing a signature from partial signatures is essentially an interpolation. The authors uses the Berlekamp and Welch decoder, so that the interpolation still yields a correct signature despite a small portion (fewer than one fourth) of partial signatures being missing or incorrect.

3.3 Proactive security and adaptability

Besides threshold signature, our key management service also employs *share refreshing* to tolerate *mobile adversaries*[iv] and to adapt its configuration to changes in the network.

Mobile adversaries are first proposed by Ostrovsky and Yung [31] to characterize adversaries that temporarily compromise a server and then move on to the next victim (e.g., in form of viruses injected into a network). Under this adversary model, an adversary might be able to compromise all the servers over a long period of time. Even if the compromised servers are detected and excluded from the service, the adversary could still gather more than t shares of the private key from compromised servers over time. This would allow the adversary to generate any valid certificates signed by the private key.

Proactive schemes [24, 20, 19, 10, 9] are proposed as a countermeasure to mobile adversaries. A proactive threshold cryptography scheme uses share refreshing, which enables servers to compute new shares from old ones in collaboration without disclosing the service private key to any server. The new shares constitute a new $(n, t + 1)$ sharing of the service private key. After refreshing, servers remove the old shares and use the new ones to generate partial signatures. Because the new shares are independent of the old ones, the

[iv] Note that the term *mobile* here is different from that of *mobile* networks.

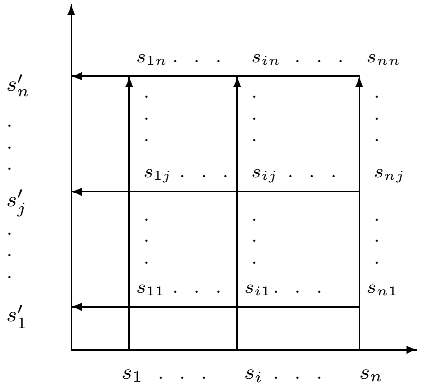
s_{1n} . . . s_{in} . . . s_{nn}
s'_n
s_{1j} . . . s_{ij} . . . s_{nj}
s'_j
s_{11} . . . s_{i1} . . . s_{n1}
s'_1
s_1 . . . s_i . . . s_n

Figure 4: Share refreshing: given an $(n, t+1)$ sharing $(s_1, \ldots, s_l)$ of a private key k, with share s_i assigned to server i. To generate a new $(n, t+1)$ sharing $(s'_1, \ldots, s'_l)$ of k, each server i generates subshares $s_{i1}, s_{i2}, \ldots, s_{in}$, which constitute the ith column in the figure. Each subshare s_{ij} is then sent securely to server j. When server j gets all the subshares $s_{1j}, s_{2j}, \ldots, s_{nj}$, which constitute the jth row, it can generate its new share s'_j from these subshares and its old share s_j.

adversary cannot combine old shares with new shares to recover the private key of the service. Thus, the adversary is challenged to compromise $t+1$ servers between periodic refreshing.

Share refreshing relies on the following homomorphic property. If $(s^1_1, s^1_2, \ldots, s^1_n)$ is an $(n, t+1)$ sharing of k_1 and $(s^2_1, s^2_2, \ldots, s^2_n)$ is an $(n, t+1)$ sharing of k_2, then $(s^1_1 + s^2_1, s^1_2 + s^2_2, \ldots, s^1_n + s^2_n)$[v] is an $(n, t+1)$ sharing of $k_1 + k_2$. If k_2 is 0, then we get a new $(n, t+1)$ sharing of k_1.

Given n servers. Let $(s_1, s_2, \ldots, s_n)$ be an $(n, t+1)$ sharing of the private key k of the service, with server i having s_i. Assuming all servers are correct, share refreshing proceeds as follows: first, each server randomly generates $(s_{i1}, s_{i2}, \ldots, s_{in})$, an $(n, t+1)$ sharing of 0. We call these newly generated s_{ij}'s *subshares*. Then, every subshare s_{ij} is distributed to server j through a secure link. When server j gets the subshares $s_{1j}, s_{2j}, \ldots, s_{nj}$, it can compute a new share from these subshares and its old share ($s'_j = s_j + \Sigma^n_{i=1} s_{ij}$). Figure 4 illustrates a share refreshing process.

Share refreshing must tolerate missing subshares and erroneous subshares from compromised servers. A compromised server may not send any subshares. However, as long as correct servers agree on the set of subshares to use, they can generate new shares using only subshares generated from $t+1$ servers. For servers to detect incorrect subshares, we use verifiable secret sharing schemes, for example, those in [7, 33]. A verifiable secret sharing scheme generates extra public information for each (sub)share using a one-way function. The public information can testify the correctness of the corresponding (sub)shares without disclosing the (sub)shares.

A variation of share refreshing also allows the key management service to change its configuration from $(n, t+1)$ to $(n', t'+1)$. This way, the key management service can adapt itself, on the fly, to changes in the network: if a compromised server is detected, the service should exclude the compromised server and refresh the exposed share; if a server is no longer available or if a new server is added, the service should change its configuration accordingly. For example, a key management service may start with the $(7, 3)$ configuration. If, after some time, one server is detected to be compromised and another server is no longer available, then the service could change its setting to the $(5, 2)$ configuration. If two new servers are added later, the service could change its configuration back to $(7, 3)$ with the new set of servers.

This problem has been studied in [5]. The essence of the proposed solution is again share refreshing. The only difference is that now the original set of servers generate and distribute subshares based on the new configuration of the service: for a set of $t+1$ of the n old servers, each server i in this set computes an $(n', t'+1)$ sharing $(s_{i1}, s_{i2}, \ldots, s_{in'})$ of its share s_i and distribute subshare s_{ij} secretly to the jth server of the n' new servers. Each new server can then compute the new share from these subshares. These new

[v] Operator "+" here could be an addition operation on a finite field such as Z_p, where $(a+b)$ means $(a+b)$ mod p.

shares will constitute an $(n', t'+1)$ sharing of the same service private key.

Note that share refreshing does not change the service key pair. Nodes in the network can still use the same service public key to verify signed certificates. This property makes share refreshing transparent to all nodes, hence scalable.

3.4 Asynchrony

Existing threshold cryptography and proactive threshold cryptography schemes assume a synchronous system (i.e., there is a bound on message-delivery and message-processing times). This assumption is not necessarily valid in an ad hoc network, considering the low reliability of wireless links and poor connectivity among nodes. In fact, any synchrony assumption is a vulnerability in the system: the adversary can launch denial of service attacks to slow down a node or to disconnect a node for a long enough period of time to invalidate the synchrony assumption. Consequently, protocols based on the synchrony assumption are inadequate.

To reduce such vulnerability, our key management service works in an asynchronous setting. Designing such protocols is hard; some problems may even be impossible to solve [8]. The main difficulty lies in the fact that, in an asynchronous system, we cannot distinguish a compromised server from a correct but slow one.

One basic idea underlying our design is the notion of weak consistency: we do not require that the correct servers be consistent after each operation; instead, we require *enough* correct servers to be up-to-date. For example, in share refreshing, without any synchrony assumption, a server is no longer able to distribute the subshares to all correct servers using a reliable broadcast channel. However, we only require subshares to be distributed to a quorum of servers. This suffices, as long as correct servers in such a quorum can jointly provide or compute all the subshares that are distributed. This way, correct servers not having certain subshare(s) could recover its subshare(s) from other correct servers.

Another important mechanism is the use of multiple signatures for correct servers to detect and to reject erroneous messages sent by compromised servers. That is, we require that certain messages be accompanied with enough signatures from servers. If a message contains digital signatures from a certain number (say, $t+1$) of servers testifying its validity, at least one correct server must have provided one signature, thus establishing the validity of the message.

We have implemented a prototype of such a key management service. The preliminary results have shown its feasibility. Due to the length restriction of this paper, we are unable to provide a detailed description of this service. Full papers describing the key management service and its underlying proactive secret sharing protocol in asynchronous system are in preparation.

4 Related Work

4.1 Secure routing

Secure routing in networks such as the Internet has been extensively studied [36, 27, 30, 45, 46, 18]. Many proposed approaches are also applicable to secure routing in ad hoc networks. To deal with external attacks, standard schemes such as digital signatures to protect information authenticity and integrity have been considered. For example, Sirios and Kent [45] propose the use of a keyed one-way hash function with windowed sequence number for data integrity in point-to-point communication and the use of digital signatures to protect messages sent to multiple destinations.

Perlman [36] studies how to protect routing information from compromised routers in the context of Byzantine robustness. The study analyzes the theoretical feasibility of maintaining network connectivity under such assumptions. Kumar [27] recognizes the problem of compromised routers as a hard problem, but provides no solution. Other works [30, 45, 46] give only partial solutions. The basic idea underlying these solutions is to detect inconsistency using redundant information and to isolate compromised routers. For example, in [46], where methods to secure distance-vector routing protocols are proposed, extra information of a predecessor in a path to a destination is added into each entry in the routing table. Using this piece of information, a path-traversal technique (by following the predecessor link) can be used to verify the correctness of a path. Such mechanisms usually come with a high cost and are avoided (e.g., in [30]) because routers on networks such as the Internet are usually well protected and rarely compromised.

4.2 Replicated secure services

The concept of distributing trust to a group of servers is investigated by Reiter [39]. This is the foundation of the *Rampart* toolkit [38]. Reiter and others [40] have successfully used the toolkit in building a replicated key management service Ω, which also employs threshold cryptography. One drawback of *Rampart* is that it may remove correct but slow servers from the group. Such removal renders the system at least temporarily more vulnerable. Membership changes are also expensive. For these reasons, *Rampart* is more suitable for tightly coupled networks than for ad hoc networks.

Gong [14] applies trust distribution to *Key Distribution Center (KDC)*, the central entity responsible for key management in a secret key infrastructure. In his solution, a group of servers jointly act as a *KDC* with each server sharing a unique secret key with each client.

In [29], Malkhi and Reiter present *Phalanx*, a data repository service that tolerate Byzantine failures in an asynchronous system. The essence of *Phalanx* is a Byzantine quorum system [28]. In a Byzantine quorum system, servers are grouped into quorums satisfying a certain intersection property. The service supports *read* and *write* operations and guarantees that a *read* operation always returns the result of the last completed *write* operation. Instead of requiring each correct server to perform each operation, the service performs each operation on only a quorum of servers. However, this weak consistency among the servers suffices to achieve the guarantee of the service because of the intersection property of Byzantine quorum systems.

In [2], Castro and Liskov extend the replicated state-machine approach [41] to achieve Byzantine fault tolerance. They use a three-phase protocol to mask away disruptive behavior of compromised servers. A small portion of servers may be left behind, but can recover by communicating with other servers.

None of the systems provide mechanisms to defeat mobile adversaries and to achieve scalable adaptability. The latter two solutions do not consider how a secret (a private key) is shared among the replicas. However, they are useful in building highly secure services in ad hoc networks. For example, we could use Byzantine quorum systems to secure a location database [15] for an ad hoc network.

4.3 Security in ad hoc networks

In [22], an authentication architecture for mobile ad hoc networks is proposed. The proposed scheme details the formats of messages, together with protocols that achieve authentication. The architecture can accommodate different authentication schemes. Our key management service is a prerequisite for such a security architecture.

5 Conclusion

In this paper, we have analyzed the security threats an ad hoc network faces and presented the security objectives that need to be achieved. On one hand, the security-sensitive applications of ad hoc networks require high degree of security; on the other hand, ad hoc networks are inherently vulnerable to security attacks. Therefore, security mechanisms are indispensable for ad hoc networks. The idiosyncrasy of ad hoc networks poses both challenges and opportunities for these mechanisms.

This paper focuses on how to secure routing and how to establish a secure key management service in an ad hoc networking environment. These two issues are essential to achieving our security goals. Besides the standard security mechanisms, we take advantage of the redundancies in ad hoc network topology and use diversity coding on multiple routes to tolerate both benign and Byzantine failures. To build a highly available and highly secure key management service, we propose to use *threshold cryptography* to distribute trust among a set of servers. Furthermore, our key management service employs share refreshing to achieve proactive security and to adapt to changes in the network in a scalable way. Finally, by relaxing the consistency requirement on the servers, our service does not rely on synchrony assumptions. Such assumptions could lead to vulnerability. A prototype of the key management service has been implemented, which shows its feasibility.

The paper represents the first step of our research to analyze the security threats, to understand the security requirements for ad hoc networks, and to identify existing techniques, as well as to propose new mechanisms to secure ad hoc networks. More work needs to be done to deploy these security mechanisms in an ad hoc network and to investigate the impact of these security mechanisms on the network performance.

6 Acknowledgment

We would like to thank Robbert van Renesse, Fred B. Schneider, and Yaron Minsky for their invaluable contributions to this work. We are also grateful to Úlfar Erlingsson and the anonymous reviewers for their comments and suggestions that helped to improve the quality of the paper.

References

[1] E. Ayanoglu, C.-L. I, R. D. Gitlin, and J. E. Mazo. Diversity coding for transparent self-healing and fault-tolerant communication networks. *IEEE Transactions on Communications*, 41(11):1677–1686, November 1993.

[2] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *Proceedings of the 3rd Symposium on Operating System Design and Implementation*, New Orleans, February 1999.

[3] Y. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–457, July - August 1994.

[4] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *Advances in Cryptology — Crypto '89 (Lecture Notes in Computer Science 435)*, pages 307–315, Santa Barbara, California, U.S.A., August 1990. Springer-Verlag.

[5] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Technical Report ISSE_TR-97-01, George Mason University, July 1997.

[6] A. Ephremides, J. E. Wieselthier, and D. J. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, 75(1):56–73, January 1987.

[7] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science*, pages 427–437, 1987.

[8] M. J. Fisher, N. A. Lynch, and M. S. Peterson. Impossibility of distributed consensus with one faulty processor. *Journal of the ACM*, 32(2):374–382, April 1985.

[9] Y. Frankel, P. Gemmel, P. MacKenzie, and M. Yung. Optimal resilience proactive public-key cryptosystems. In *Proceedings of the 38th Symposium on Foundations of Computer Science*, 1997.

[10] Y. Frankel, P. Gemmell, P. MacKenzie, and M. Yung. Proactive RSA. In B. Kaliski, editor, *Advances in Cryptology — Crypto '97 (Lecture Notes in Computer Science 1294)*, pages 440–454, Santa Barbara, California, U.S.A., August 1997. Springer-Verlag.

[11] M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson. The digital distributed systems security architecture. In *Proceedings of the 12th National Computer Security Conference*, pages 305–319. NIST, 1989.

[12] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In N. Koblitz, editor, *Advances in Cryptology — Crypto '96 (Lecture Notes in Computer Science 1109)*, pages 157–172, Santa Barbara, California, U.S.A., August 1996. Springer-Verlag.

[13] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In U. Maurer, editor, *Advances in Cryptography — Proceedings of Eurocrypt '96 (Lecture Notes in Computer Science 1070)*, pages 354–371, Zaragoza, Spain, May 1996. Springer-Verlag.

[14] L. Gong. Increasing availability and security of an authentication service. *IEEE Journal on Selected Areas in Communications*, 11(5):657–662, June 1993.

[15] Z. J. Haas and B. Liang. Ad hoc mobility management using quorum systems. *IEEE/ACM Transactions on Networking*, 1999.

[16] Z. J. Haas and M. Perlman. The performance of query control schemes for zone routing protocol. In *SIGCOMM'98*, June 1998.

[17] A. A. Hassan, W. E. Stark, and J. E. Hershey. Frequency-hopped spread spectrum in the presence of a flower partial-band jammer. *Transactions on Communications*, 41(7):1125–1131, July 1993.

[18] R. Hauser, T. Przygienda, and G. Tsudik. Lowering security overhead in link state routing. *Computer Networks*, 31(8):885–894, April 1999.

[19] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public-key and signature schemes. In *Proceedings of the Fourth Annual Conference on Computer Communications Security*, pages 100–110. ACM, 1997.

[20] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In D. Coppersmith, editor, *Advances in Cryptology — Crypto '95 (Lecture Notes in Computer Science 963)*, pages 457–469, Santa Barbara, California, U.S.A., August 1995. Springer-Verlag.

[21] J. Ioannidis, D. Duchamp, and J. M. Gerald Q. IP based protocols for mobile internetworking. *ACM SIGCOMM Computer Communication Review (SIGCOMM'91)*, 21(4):235–245, September 1991.

[22] S. Jacobs and M. S. Corson. MANET authentication architecture. Internet Draft (draft-jacobs-imep-auth-arch-01.txt), February 1999.

[23] P. Jacquet, P. Muhlethaler, and A. Qayyum. Optimized link state routing protocol. IETF MANET, Internet Draft, November 1998.

[24] S. Jarecki. Proactive secret sharing and public key cryptosystems. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, September 1995.

[25] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad-hoc wireless networks. *Mobile Computing*, 1996.

[26] C. Kaufman. DASS: Distributed authentication security service. Request for Comments: 1507, September 1993.

[27] B. Kumar. Integration of security in network routing protocols. *SIGSAC Reviews*, 11(2):18–25, 1993.

[28] D. Malkhi and M. Reiter. Byzantine quorum systems. *The Journal of Distributed Computing*, 11(4), 1998.

[29] D. Malkhi and M. Reiter. Secure and scalable replication in Phalanx. In *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems*, pages 51–58, October 1998.

[30] S. Murphy and J. J. Garcia-Luna-Aceves. An efficient routing algorithm for mobile wireless networks. *MONET*, 1(2):183–197, October 1996.

[31] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proceedings of the 10th ACM Symposium on Principles of Distributed Computing*, pages 51–59, 1991.

[32] V. D. Park and M. S. Corson. A highly adaptable distributed routing algorithm for mobile wireless networks. In *IEEE INFOCOMM'97*, Kobe, Japan, 1997.

[33] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology — Crypto'91 (Lecture Notes in Computer Science 576)*, pages 129–140, Santa Barbara, California, U.S.A., August 1992. Springer-Verlag.

[34] C. E. Perkins. IP mobility support. Request for Comments: 2002, October 1996.

[35] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *IEEE WMCSA'99*, New Orleans, LA, February 1999.

[36] R. Perlman. *Network Layer Protocols with Byzantine Robustness.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1988.

[37] M. B. Pursley and H. B. Russel. Routing in frequency-hop packet radio networks with partial-band jamming. *IEEE Transactions on Communications*, 41(7):1117–1124, July 1993.

[38] M. K. Reiter. The Rampart toolkit for building high-integrity services. *Theory and Practice in Distributed Systems (Lecture Notes in Computer Science 938)*, pages 99–110, 1995.

[39] M. K. Reiter. Distributing trust with the Rampart toolkit. *Communications of the ACM*, 39(4):71–74, April 1996.

[40] M. K. Reiter, M. K. Franklin, J. B. Lacy, and R. N. Wright. The Ω key management service. *Journal of Computer Security*, 4(4):267–297, 1996.

[41] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, December 1990.

[42] N. Shacham and J. Westcott. Future directions in packet radio architecture and protocols. *Proceedings of the IEEE*, 75(1):83–99, January 1987.

[43] J. Sharony. A mobile radio network architecture with dynamically changing topology using virtual subnets. In *Proceedings of ICC/SUPERCOM'96*, pages 807–812, Dallas, TX, June 1996.

[44] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread Spectrum Communications*, volume I-II. Computer Science Press, Rockville, MD, 1985.

[45] K. E. Sirois and S. T. Kent. Securing the Nimrod routing architecture. In *Proceedings of Symposium on Network and Distributed System Security*, pages 74–84, Los Alamitos, CA, February 1997. The Internet Society, IEEE Computer Society Press.

[46] B. R. Smith, S. Murphy, and J. J. Garcia-Luna-aceves. Securing distance-vector routing protocols. In *Proceedings of Symposium on Network and Distributed System Security*, pages 85–92, Los Alamitos, CA, February 1997. The Internet Society, IEEE Computer Society Press.

[47] J. J. Tardo and K. Algappan. SPX: Global authentication using public key certificates. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 232–244, Oakland, California, May 1991. IEEE Computer Society.

[48] F. Teraoka, Y. Yokore, and M. Tokoro. A network architecture providing host migration transparency. *ACM SIGCOMM Computer Communication Review (SIGCOMM'91)*, 21(4):209–220, September 1991.

[49] C.-K. Toh. Associativity-based routing for ad hoc mobile networks. *Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems*, 4(2):103–139, March 1997.