SAND REPORT

SAND2003-4395
Unlimited Release
Printed December 2003

Enhancements for Distributed Certificate Authority Approaches for Mobile Wireless Ad Hoc Networks

William Erik Anderson, John T. Michalski and Brian P. Van Leeuwen

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/ordering.htm

SAND2003-4395
Unlimited Release
Printed December 2003

Enhancements for Distributed Certificate Authority Approaches for Mobile Wireless Ad Hoc Networks

William Erik Anderson
Cryptography and Information Systems Surety Department

John T. Michalski and Brian P. Van Leeuwen
Networked Systems Survivability & Assurance Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0785

Abstract

Mobile wireless ad hoc networks that are resistant to adversarial manipulation are necessary for distributed systems used in military and security applications. Critical to the successful operation of these networks, which operate in the presence of adversarial stressors, are robust and efficient information assurance methods. In this report we describe necessary enhancements for a distributed certificate authority (CA) used in secure wireless network architectures. Necessary cryptographic algorithms used in distributed CAs are described and implementation enhancements of these algorithms in mobile wireless ad hoc networks are developed. The enhancements support a network's ability to detect compromised nodes and facilitate distributed CA services. We provide insights to the impacts the enhancements will have on network performance with timing diagrams and preliminary network simulation studies.

This page intentionally left blank.

Contents

Figures

Tables

This page intentionally left blank.

1. Introduction

Many new military and homeland security systems, such as distributed combat systems, sensor systems, and emergency response systems, depend on timely situational awareness information. In many cases, situational awareness information is collected and distributed by a large number of nodes grouped into an ad hoc network. Since there typically is no fixed communication infrastructure in place to support the exchange of information in ad hoc networks, the nodes must depend on one another to create and maintain a wireless network that reconfigures as the network topology changes. Changes in the network topology are often attributed to node movement, dynamic node participation, and environmental factors that disrupt the wireless communication channel. To further complicate matters, the nodes that makeup ad hoc networks are typically mobile and battery operated and thus must minimize power consumption in order to increase operation life. Additionally, the dependence on a wireless communication channel requires that node transmissions be minimized to use the allocated spectrum efficiently. Therefore, robust network technologies are necessary to operate in this challenging, resource-constrained environment.

In applications where information has privacy, integrity, non-repudiation, authentication, and/or availability requirements, networks must employ information assurance (IA) features. Unlike a wired communication channel that has a physical medium that can be protected, wireless communications are transmissions over a free space medium that can be easily eavesdropped without detection. Networks comprised of mobile nodes may have dynamic node participation and may also have subgroups of nodes partitioned from the network for extended periods of time. In our targeted applications, we assume nodes may move into areas that increase their potential to be compromised by an adversary. Thus technologies must address the IA needs while operating over a free space wireless channel that is subject to both adversary and environmental stressors. The IA technologies must be compatible with resource-constrained mobile nodes that may be subject to physical manipulation by an adversary.

Cryptosystems and authentication protocols are employed in networks to address the previously mentioned IA concerns. Authentication is the primary service necessary to begin secure network operation. The compromise of the authentication service breaks down the entire IA system and the network cannot proceed to provide the other services without the valid identities of the communicating nodes being successfully established [6]. Once the node-to-node authentication is established, the communicating nodes can realize privacy, integrity, and non-repudiation with the Diffie-Hellman key exchange protocols [3].

Current approaches for authentication services depend on centralized management approaches by either key distribution centers or certificate authorities (CA) [7]. In cases where a specific node can be protected and is accessible by other nodes of the network, a centralized approach may be acceptable. However, for the wireless ad hoc networks that we envision for our targeted applications, a centralized approach will suffer from a single-point of service denial and may be unreachable by network nodes requiring CA services. Thus a more robust CA approach must be used. This need for wireless ad hoc networks is currently a very active research area.

In this research report, we build on current research that applies distributed CAs to wireless ad hoc networks. Several current research efforts are developing methods to distribute the functionality of CAs to groups of individual network nodes [4, 7]. Luo [7] describes a method to minimize the centralized management by requiring the nodes in the network to collaboratively self-secure themselves. Distributed approaches, however, can become resource intensive and thus create difficulty when deploying in a resource-constrained wireless network. Research presented in [7] describes a method that incorporates a more efficient centralized CA approach that operates under most network conditions and has a backup mode that is based on a distributed CA if the primary centralized CA fails. The primary objective of these research efforts is the development of methods to make distributed CA more effective in a resource-constrained wireless ad hoc network. It is this same objective toward which our research is directed: *methods to make a distributed CA more effective in a wireless ad hoc network*.

Our primary contribution includes two advances to facilitate a distributed CA approach in wireless ad hoc networks. Our first advance is a method that employs *verifiable secret shares* (VSS) to discover compromised nodes that attempt to disrupt network operation by contributing bogus certificate-shares in a network with distributed CA. The approach assumes the certificate-shares are valid until an invalid certificate is generated and thus can be more efficient than current implementations. A node will expend the necessary resources to verify each certificate-share only after an invalid certificate is generated. Our second advance is a method that uses a controlled group-cast to control the depth of propagation of a certificate-share request. This method adaptively controls the number of nodes that receive a request for a certificate-share. As the density of nodes that can contribute certificate-shares varies, the method adjusts the number of hops the request is propagated to, resulting in more efficient use of the network's scarce resources.

The remainder of the research report is organized as follows. A brief overview of public key cryptography in a wireless ad hoc network and approaches to implement CAs is discussed in Section 2. In Section 3 we provide an overview of the implementation of distributed CA services. In Section 4 verification methods and our approach to implementing the verification process is described. An initial protocol to facilitate an efficient distributed CA is presented in Section 5. Finally, future work and a conclusion are presented in Sections 6 and 7.

2. Background

Both symmetric key cryptosystems and public key cryptosystems have been successfully applied to wireless communication networks. Our targeted applications have dynamic node participation that require both forward- and reverse-confidentiality and also may suffer node compromise and thus require the removal of the node. Since nodes may have to be removed from the network, we focus on public key cryptosystems with CAs. We assume our CAs also employ certificate revocation lists (CRL) to help control network membership.

Public key cryptography uses two distinct keys – one public and the other private. Public key cryptography is used to encrypt sensitive data, establish authentication, and provide digital signatures. Public key systems by definition rely on problems that are computationally infeasible to solve and therefore cannot provide unconditional security. An adversary may choose to search

through all possible solutions till a correct one is found. Fortunately, searching through every possible solution may require an unmanageably large amount of work. Therefore, security can be measured by the amount of time and energy that needs to be invested in order to break the system.

Digital certificates are used to establish confidence in a user's public key. A *digital certificate* is a cryptographically signed proof associating a user with a public key. Digital certificates can be verified publicly, and hence provide confidence of this binding. Like all cryptographic primitives, digital certificates may be vulnerable to certain types of attack. For instance, a malicious user may try a substitution attack by substituting a valid certificate with a bogus one. One protects against this type of attack and variants of it with digital certificates that are maintained by a trusted host. In an ad hoc network this can be carried out by a trusted third party known as a *certificate authority* (CA). To maintain digital certificates of many users, a public key infrastructure is needed. A *public key infrastructure* provides certification services, including certificate issuing/renewal, certificate revocation, and database maintenance.

There are several critical design objectives that must be considered when employing a public key infrastructure in a wireless ad hoc network. These design objectives include availability, robustness against compromise of the CA, and overall scalability of the network. A brief description of each of these objectives follows.

Availability: A network is considered to have good availability if the CA can provide certification services on demand. In networks with fixed topology, availability may be easily established with a single optimized CA based on the topology of the network. In contrast, in an ad hoc network availability can be difficult to maintain since network topologies, node participation, and partitions are dynamic.

Robustness against compromise: Network nodes may be subject to intrusion, denial of service (DoS) attacks, or even damage. A network is robust if it can maintain services even after individual nodes have gone offline or become compromised. The ability to detect and take action against bad behavior is critical for making a network robust.

Scalability: Scalability describes the ability of the network architectures and protocols to support increasing node numbers.

The above design objectives imply guidelines that must be considered for our targeted applications. Depending on the applications, network IA requirements, and operating environment, there are several implementation choices for a CA in a wireless network. The implementation choices are 1) a single CA, 2) multiple CAs, and 3) a distributed CA. A brief description of each follows.

Single Certificate Authority: In a single CA design all certificate services are provided by a single node. The advantage of using this particular design is that only a single CA is responsible for all the network services and updates. For a static network with limited node dynamics and reliable reach ability, this might be an attractive option. There are several disadvantages, however, when the network is dynamic rather than static. For instance, a single CA may suffer from availability problems because of the changing dynamics of the network. A user may be out

of range or may simply have problems establishing a good communication link. Also, a single CA may be vulnerable to direct attacks. An adversary may try to jam the communication of the CA, leaving the network inoperative. Considering these factors, it is unlikely that a single CA may be the best design choice for a dynamic network operating in a hostile environment.

Multiple Certificate Authorities: Another design option is to replace a single CA with redundant copies. This fixes the availability and fault tolerance problems associated with a single CA. However, in a hostile environment, redundant CAs increase the likelihood that one of them may be compromised. If a CA is compromised, the entire network security is lost, since an adversary has access to the secret parameters of the network. If the CAs have strong protective capabilities, then multiple CAs may be a viable alternative. For example in a combat situation the network may assign the CAs to be unmanned air vehicles (UAV's) [7]. The only way to disrupt these CAs is to destroy them, which may be difficult.

Distributed Certificate Authority: A distributed certificate authority replaces a single CA with multiple nodes that share the same responsibilities of a single CA. Instead of having the network secret parameters all in one location, as a single CA, they are distributed throughout multiple nodes. This way if a node is compromised by an adversary, they learn little to no information about the network secrets. Certification services are provided by a threshold number of distributed CA nodes. If a node has a certification request, it must contact at least some fixed number of nodes, the threshold number, before its request can be answered. The certificate can be reconstructed when enough distributed CAs have responded with certificate-shares. In a dynamic network that operates in a hostile environment, distributed certificate authorities provide the most secure approach.

The disadvantage of a distributed CA is the added complexity in the design of a secure system when compared to the other approaches. Great care must be taken to insure the system does not suffer from fault tolerance issues. For instance, if an adversary compromises a single distributed CA node or if this node is damaged, it may respond to a certification request with a bogus certificate-share. If the system is not capable of dealing with bogus certificate-shares, then it may make the network inoperable. This problem can be mitigated by implementing verifiable secret sharing (VSS) [4, 12], which is described in a later section. This addresses the above fault tolerance problem by adding an additional stage when a certificate is requested. VSS provides the basis for a requesting node to determine where a false certificate-share originated. Once a compromised or defective node is identified an accusation may be filed with the network community. This protects against potential network disruption by malicious adversaries injecting bogus certificate-shares. However, current implementations of VSS create an increased burden on processing capability since the VSS process is initiated each and every time a certificate request and response occurs. Our research addresses this increased burden on the processing capability by providing an as-needed verification process.

3. Distributed Certification Services for Wireless Ad Hoc Networks

In the previous section several approaches to employing a CA are described and the advantages and disadvantages are discussed. Of the three approaches, distributed certificate authorities provide a solution that best meets the objectives described in Section 2 for a wireless ad hoc

network operating in a hostile environment. As described in Section 2, the distributed CA has the disadvantage of added complexity and requires the exchange of information among a group of nodes. This exchange of information consumes wireless bandwidth. To enhance the distributed CA to improve operation in a wireless ad hoc network, we set the following goals:

- Certification services are to be distributed among the network nodes/participants.
- The allocation of secret shares for new nodes/participants that join the network is performed distributively by a network coalition.
- Misbehaving nodes that supply corrupted certificate-shares can be detected and isolated.
- All accusations are publicly verifiable and false accusations are detectable.

The following section briefly describes the protocol model of a distributed certificate authority.

3.1 Protocol Model of a Distributed Certificate Authority

In this section we provide a general and brief description of the protocol model for a distributed certificate authority. In particular, we highlight the implementation of certificate renewals and self-initialization. For a more detailed description of the algorithms we refer to the reader to [4]. Unless stated otherwise, all communications are assumed over a wireless communication channel.

The network retains a secret key *s*; however, no node/participant in the network has direct access to it or the ability to reconstruct it unless some assumption is broken. The network secret *s* is used to sign certificates that bind the node and the node's public key, p_{node}. Every node in the network must maintain a valid certificate to participate in the network. Each certificate, *cert*, contains an expiration time t_{exp} and a statement that certifies that the node's public key is p_{node} and is valid up to time t_{exp} [4].

Given any network of size *n,* the network secret *s* will be distributed as secret shares to each node in the network using a (*k, n)*-threshold secret sharing scheme. The security parameter *k* is chosen to maximize efficiency in obtaining the necessary number of shares and to minimize the probability of recovering *s*. This value depends both on the size of the network and the probability of compromise of each node. In a low-risk environment, the value *k* may be set relatively low to maximize efficiency. However, in a high-risk environment, *k* will need to be increased to ensure the secrecy of *s* is preserved.

The public key p_{node} of the receive node is used by the source node to encrypt data. The public key of the source node is used by the receive node to verify the signature of the source node. The source node uses its private key to sign data. The receive node uses its private key to decrypt data. A node is trusted if any *k* nodes in the network verify its certificate *cert* is valid. If a new node joins the network, it must exchange certificates in order to establish trust.

Nodes will employ transport layer security to encrypt, decrypt, sign, and validate messages with keys derived from the public key infrastructure.

3.2 Distributed Certificate Renewal

An important operation for a secure network is a *distributed certificate renewal*. Every node in the network must renew its certificate before its expiration time t_{exp}. In the event that a certificate-share is corrupted, an interactive publicly verifiable secret share (PVSS) scheme is employed [4, 12].

Certificate Request: A certificate renewal is performed by any k nodes in the network. A node that requests a certificate renewal broadcasts a service request along with its unsigned digital certificate *cert*. Each node that services the request returns a *certificate-share* $cert_{si}$ by applying its secret share s_i. The recovery algorithm $Recover_{cert,k}$ and any k certificate-shares is used to obtain the certificate,

$$cert_s \leftarrow Recover_{cert,k}(cert_{s_{i_1}}, \ldots, cert_{s_{i_k}})$$

The recovered certificate $cert_s$ is the digital certificate signed by the network's secret key s.

The certificate request operation works fine when any k nodes each return a valid certificate-share. However, if a single node returns a bogus or corrupted certificate-share, the requesting node currently has no way of determining if any certificate-shares are bogus until the node attempts to construct a certificate. The requesting node may determine that one or more of the certificate-shares are bogus if it cannot construct a valid certificate. At this time the requesting node can request a new set of k certificate-shares and attempt to construct a valid certificate. The requesting node must obtain an entire new set of k certificates since it cannot determine which node contributed the bogus certificate-share. This is not an acceptable situation since a single bogus certificate-share will require the node to expend resources to obtain k new certificates. In some cases, k new certificate-shares may not be available. Our research has identified and describes a solution to mitigate this concern. The solution is described in Section 4.

3.3 Distributed Self-Initialization

Secure wireless networks used in our targeted applications are comprised of mobile nodes that may result in nodes entering and leaving the network. When a new node enters the network and does not have access to a dealer, an alternative method is necessary for this node to join the coalition of nodes able to provide secret shares. This alternative method is necessary to securely provide the node the ability to generate dynamically new secret shares that are compatible with other coalition nodes already in the network. Luo and Lu [4] propose a distributed self-initialization algorithm to address this problem. In particular, they use a coalition of members already in the network. The coalition communicates interactively to generate partial-secret shares that can be combined to generate the secret share for the new node.

Initialization: The generation of a secret share for a new node that joins the network is constructed by a coalition $\mathcal{S}$, of k nodes currently in the network. A new node broadcasts an initialization request, along with the IDs of the nodes of its chosen coalition. Each participant communicates privately with every node in $\mathcal{S}$ by exchanging secret information. Each node $j \in \mathcal{S}$,

subsequently returns a shuffled version of its secret share s_j to the new node. A shuffled version is used to protect the value of its secret share s_j and will be described in Section 4.

$$\overline{s_j} \leftarrow Shuffle_{t,k}(s_j, \mathcal{S})$$

The value s_j depends on j's interaction with the other participants in $\mathcal{S}$ and the current size of the network. Once the new node has obtained the shuffled shares it may construct its secret share s_{n+1} by,

$$s_{k+1} \leftarrow Unshuffle_{t,k}(\overline{s_1}, \ldots, \overline{s_t})$$

If every participant of $\mathcal{S}$ is honest, then the value s_{n+1} will be the same value obtained by $Share_{n+1}(s)$ extracted for the n+1 secret share. $Share_n$ is an algorithm described in Section 4.

As with the certificate renewal operation, each node must return a valid shuffled version of its secret share. If a single malicious node or multiple malicious nodes desire to disrupt the self-initialization process, they may contribute a bogus shuffled version of their secret share. Thus a verification method is necessary to validate the result, and, if the result does not validate, to identify which element is bogus. The following section describes a method.

4. Verifiable Secret Shares to Verify Integrity of Certificate-shares

In Section 3 we described necessary procedures to maintain certificates in a wireless ad hoc network. We also described the impact to the procedures if malicious nodes attempt to disrupt the distributed procedures by contributing bogus elements to the certificate renewal or self-initialization procedures. To provide protection against this, *verifiable secret sharing* (VSS) is used [2]. In the event there are malicious participants, VSS allows the user to detect a corrupted share. In this way, malicious or faulty nodes can be located and dealt with accordingly. VSS is a particular form of a cryptographic primitive known as secret sharing.

A secret sharing scheme distributes a secret s among n nodes/participants, $P_1, \ldots, P_n$. The individual pieces of s are called secret shares and, by themselves, usually do not provide any valuable information about the secret s. All secret sharing schemes are designed to recover the secret using some subset of the secret shares. If no less than k partial share combinations can be combined to reconstruct s, then the secret sharing scheme is said to be a (k,n)-threshold scheme.

Formally, a (k,n)-threshold scheme consists of two algorithms, a share algorithm $Share_n$ and a recover algorithm $Recover_k$. The share algorithm takes the secret s and returns,

$$(s_1, \ldots, s_n) \leftarrow Share_n(s)$$

with each s_i returned to node P_i, i=1,…,n. If any k nodes want to recover s, they execute the algorithm $Recover_k$ on the partial shares,

$$s \leftarrow Recover_k(s_{i_1}, \ldots, s_{i_k})$$

It should be computationally infeasible to recover s with less than k partial shares. To help illustrate the above definition, we present a polynomial version of secret sharing, first introduced by Shamir [13]. This particular secret sharing scheme is of particular interest since it is used in several of the distributed algorithms for our simulation.

Polynomial Secret Sharing: Let $\mathbf{F}$ be any finite field. We will construct a *(k,n)*-threshold scheme following [1, 13]. The secret to be shared s will be taken as any element in $\mathbf{F}$. The nodes in our network will also have a distinct representation as elements $v_i \in \mathbf{F}$, i=1,…,n. To divide s into n pieces, choose any random degree k-1 polynomial $p(x) \in \mathbf{F}[x]$ that satisfy $p(0)=s$. The distributed secret shares are defined by $s_i:=p(v_i)$. Given any k distinct pairs $(v_i, p(v_i))$, the original polynomial $p(x)$ can be reconstructed using interpolation. This follows from the fact that any degree k-1 polynomial $f(x)$ that satisfies $f(v_i)=p(v_i)$, i=1,…,k is the necessarily unique polynomial $p(x)$. Consider,

$$l_{v_i}(x) := \prod_{\substack{j=1,\ldots,k \\ j \neq i}} (x - v_j)(v_i - v_j)^{-1}$$

.

We reconstruct $p(x)$ by

$$p(x) = \sum_{j=1}^{k} p(v_j) \cdot l_{v_j}(x)$$

.

Therefore, the original secret s is recovered with $p(0)$. Given any less than k value pairs $(v_i, p(v_i))$, the secret can be recovered with no more than $1/|\mathbf{F}|$ probability of success. In other words, the best a recovery algorithm can do is to randomly guess the secret. The above scheme does not provide the user the ability to discover corrupted shares.

In the initialization stage described in Section 3.3, each new node that joins the network is represented by $v_{n+1} \in \mathbf{F} \backslash \{v_1, \ldots, v_n\}$. The secret share for v_{n+1} is $s_{n+1}:=p(v_{n+1})$. This value may be computed using any k members in the network. Using the above equation, the values $p(v_j) \cdot l_{vj}(v_{n+1})$, $j=1,\ldots,k$ add up to s_{n+1}. However, if each node v_j sends $p(v_j) \cdot l_{vj}(v_{n+1})$, then an obvious problem arises: node v_{n+1} may recover $p(v_j)$, which is the secret share of v_j. To address this problem, a shuffled version of this value must be sent instead. The shuffled version of $p(v_j) \cdot l_{vj}(v_{n+1})$ is computed based on an interactive protocol between node v_j and the other k members.

The goal of VSS is to detect faulty and misbehaving participants. This detection is usually carried out by the node making the certification request. In *publicly verifiable secret sharing* (PVSS), all participants that have access to the network may be part of the detection process.

This protects the dealer and nodes in the network from accumulating false accusations, since all accusations are publicly verifiable. Like public key cryptography, PVSS relies on communication over public channels.

Typically, PVSS schemes rely on a string *PROOF* to show that a certificate-share provided by the participant was computed correctly. The string *PROOF* is generated interactively or non-interactively by the node receiving the certificate-share. In an interactive proof both the receiving and providing node communicate with each other. The goal of the providing node is to prove to the receiving node that it correctly computed the information it sent. This is usually carried out using a zero-knowledge proof.

A *zero-knowledge proof* is a proof between two parties, a prover and verifier. In the above case, the prover is the providing node and the verifier is the receiving node. The prover has some secret information, for example the secret key used in the computation of the certificate-share. The verifier must be able to ascertain with near certainty that the information sent is correct when the providing node is honest and with high probability verify the information is incorrect when it is dishonest. Based on the interaction between the two parties, the verifier should not be able to extrapolate computationally any more information than what it could if the interaction never occurred in the first place – hence the name zero-knowledge. The same ideas follow in a non-interactive zero-knowledge proof as well, except that the verifier must use information that is publicly available in order to complete the verification.

In this research we consider networks that are comprised of nodes that may be vulnerable to compromise by an adversary. A compromised or malicious node may try to make numerous attempts to disqualify honest participants or obtain useful information during its interaction with other nodes in the network. A poorly designed network may disqualify a participant based on a single accusation. This particular flaw can be exploited under the right conditions. Therefore, considerable effort should be spent giving honest participants the ability to protect themselves from false accusations.

4.1 Verification of Distributed Certificate Renewal

Each received certificate-share can be validated using a verifiable secret sharing scheme. After the requesting node generates its new certificate, $cert_s$, by combining any k certificate-shares, it must verify the certificate by applying the network's public key *p.* If $cert=(cert_s)_p$, the node is satisfied it has generated a valid certificate. If equality does not hold, the node suspects that at least one certificate-share may be bogus.

In our proposed protocol, a challenge response is issued to each of the responding nodes, only after an invalid certificate was generated. The challenge response relies on an interactive zero-knowledge proof that is publicly verifiable. Each node publishes a signed string $PROOF_{si}$ based on its interaction. The string $PROOF_{si}$ verifies that node *i* has knowledge of its secret share s_i, and that it applied s_i correctly to the certificate *cert*. The requesting node examines each of the returned strings $PROOF_{si}$ and identifies the node or nodes that contributed the bogus certificate-share(s). The bogus certificate-share(s) are dropped and replaced with certificate-share(s) not used in the original certificate-share recovery calculation. If no extra shares are available, the

requesting node should initiate another certificate share request to any set of nodes not part of the original request. When a replacement certificate share is obtained, the combining algorithm is re-executed to calculate a renewed certificate. The process is repeated until a valid certificate is calculated or no additional certificate shares can be obtained.

In addition, when the original requesting node identifies the node that contributed a bogus certificate share, it issues an accusation against that node. If the suspect node receives additional accusations, it will be added to the certificate revocation list and thus removed from participation in the network. Since the proof is made public and signed by node *i* the other members may also verify.

The major advance in our implementation over the work proposed by [4] is that our proposed protocol will not, under normal network operation, initiate the *PROOF* request. It is only after an invalid certificate is generated that our protocol initiates the effort to discover the bogus certificate share(s).

4.2 Verification of Distributed Self-Initialization

If a single malicious node or multiple malicious nodes desire to disrupt the self-initialization process, they may contribute a bogus shuffled version of their secret share. Thus a verification method is necessary to validate the result and, if the result does not validate, to identify which element is bogus.

The verification algorithm for distributed self-initialization relies on a set W of public witness values. The elements in W are used to verify the integrity of the secret shares for each node in the network. Once a new node has constructed its secret share s_{n+1}, it runs a verify algorithm *Verify* on s_{n+1} and W [4]. *Verify* returns a single bit, that refers to whether the value s_{n+1} is correct or not. If *Verify* indicates an incorrect value, the node suspects that at least one secret share may be bogus.

In our protocol, if an invalid result is identified by *Verify*, the original requesting node issues a challenge response to each node that returned a secret share. When a challenge is issued, each node $i \in S$ publishes a signed string $PROOF_{si}$ that verifies it has knowledge of its secret share s_i and it computed the shuffled version of the secret share s_i correctly. If any of the proofs are not valid, the requesting node initiates another secret share request to replace the shares that are bogus.

Similar to the verification of distributed certificate renewal described in Section 4, the original requesting node can identify the node that contributed a bogus secret share and will issue an accusation against that node. If the suspect node receives additional accusations, it will be added to the certificate revocation list and thus removed from participation in the network. Since the proof is made public and signed by node *i*, the other members may also verify the node's secret share.

This approach results in more efficient network operation than the approach described in [4] which employs the *PROOF* operation for every initialization. A network that operates without

any adversary manipulation attempts will not expend resources on the *PROOF* operation. It is only after an invalid result is detected that the *PROOF* operation is executed.

4.3 Timing Delays for Distributed Certificate Services

When selecting or developing IA approaches for secure network architectures, it is necessary to consider the computational power required of the system. In cases where a less capable processor is selected, the network operation may be impeded by the time the processor takes to compute the IA algorithm. Our proposed approach is based on distributed cryptographic methods that can be resource-intensive when used in a mobile wireless network. In this section we provide timing delays for the distributed certificate services described in the previous sections.

The timing delays attributed to each protocol are a measure of the time it takes to complete each cryptographic algorithm. Since we are analyzing exclusively the performance of the cryptographic algorithms, network traffic delay is not considered. The times necessary to execute the cryptographic algorithms are realized using timing data obtained from Safenet Inc. data sheets [9, 10, 11]. Safenet specializes in providing proprietary information for embeddable cryptography products that are compatible with the instruction sets for all ARM processor cores. We selected an 80MHz, 32-bit ARM7 microprocessor for our timing analysis, because the ARM7 family of processors is commonly used in low-power wireless devices. The computational delay for 1024-bit addition, multiplication, exponentiation, random number generation, RSA SK-Sign, and SHA-1 are all given in Table 1. All other operations are assumed to be comparatively small and therefore are not considered.

Table 1: Computational delay for 1024-bit operations on an 80Mhz, 32-bit ARM7 processor with 1024-bit key length

Addition	Exponentiation	Multiplication	RNG	RSA SK-Sign	SHA-1
1.20µsec	20.45msec	7.63µsec	64.0µs	6.45msec	2.0µsec

Our public key infrastructure uses RSA primitives with key lengths of 1024-bits. When the secret parameters for RSA signing are known, we use the Chinese Remainder Theorem to optimize the time to take an exponentiation. When the secret parameters are not known, regular exponentiation is used. We will use SHA-1 as our standard cryptographic hash. The timing diagrams for both the certificate issuing/renewal and verification algorithms are given in Figures 1 and 2 respectively.

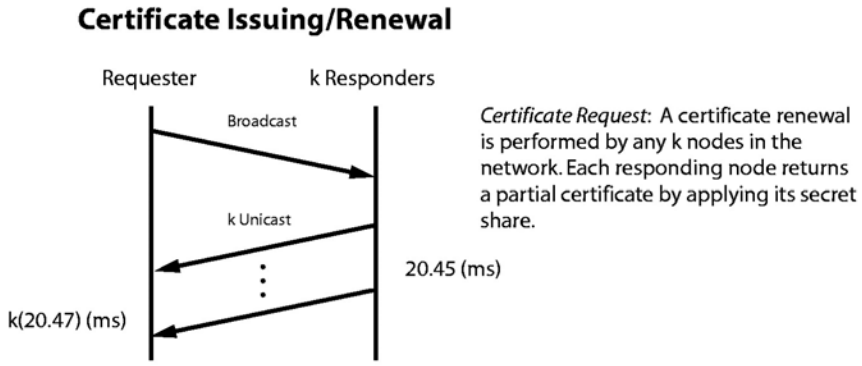

Figure 1: Timing diagram for certificate issuing/renewal with 1024 bit key length.

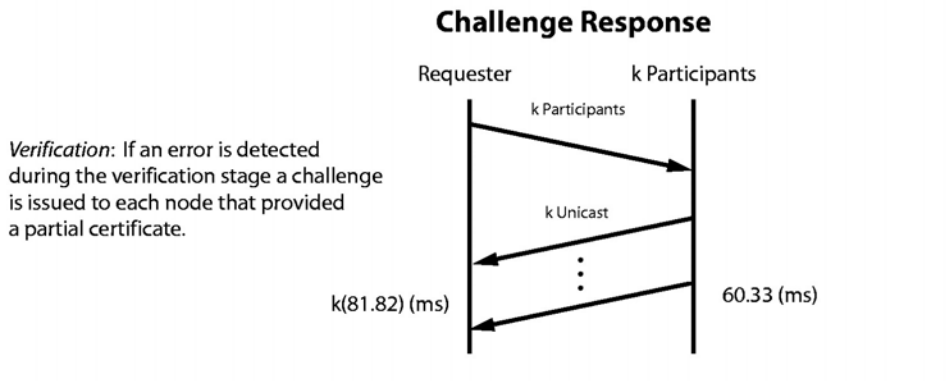

Figure 2: Timing diagram of verification stage for certificate issuing/renewal.

The computation overhead for both protocols is dominated by the time to execute an exponentiation. Therefore, any increase in performance would likely be a result of improving this timing value. The verification protocol also uses RSA signing, but since each node knows the RSA secret parameters for their public keys, they are allowed to make use of an optimized exponentiation.

The distributed self-initialization protocol occurs in several stages. In the shuffling factor stage each pair of members securely exchanges a shuffling factor. The shuffling factor is generated as the sum of two random numbers with each one of the two numbers generated individually by a member. The two numbers are then securely exchanged using a three-pass authenticated key exchange protocol. Once the shuffling factors have been exchanged, they are combined to hide the node's secret share. As illustrated in Figure 3, this combination takes approximately the time for a single exponentiation when the size of the coalition k is small. When the coalition is large, the additions used to combine the shuffling factors can have a significant impact on the timing delay. If k=100, it takes approximately 122 ms for each coalition member to compute its shuffled

share, excluding network delay. The requester requires a total of 2.7 seconds to compute its secret share after it has obtained enough shuffled shares. Table 2 lists timing values for various values of k.

In the verification stage shown in Figure 4, each coalition member tries to prove that it is not faulty. For each member this timing delay is roughly that of computing an exponentiation and RSA signature. In the final stage, the requester takes the returned proofs and verifies them by computing on the order of k^2 multiplications and k exponentiations. If we again choose k=100, the coalition members' delay is constant at 26.9ms while the requester is 2.1s. See Table 2 for additional values.

Table 2: Timing delays for distributed self-initialization using an 80Mhz, 32-bit ARM7 processor with 1024-bit key length

k	Initialization: Requester	Initialization: Coalition Member	Verification:
10	0.27 sec	121.7 ms	0.21 sec
50	1.30 sec	121.8 ms	1.05 sec
100	2.7 sec	121.8 ms	2.10 sec

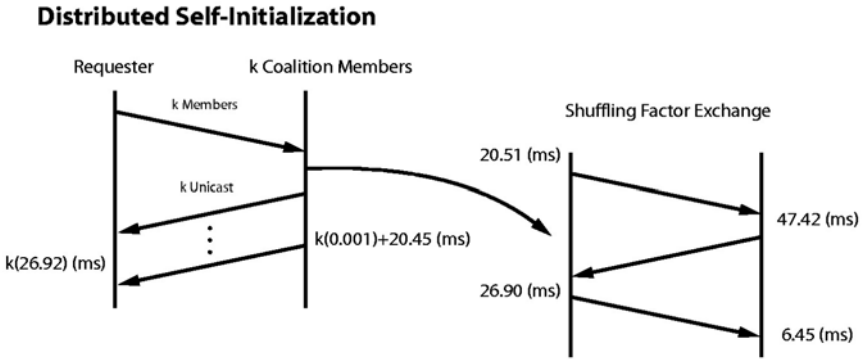

Figure 3: Timing diagram for distributed self-initialization.

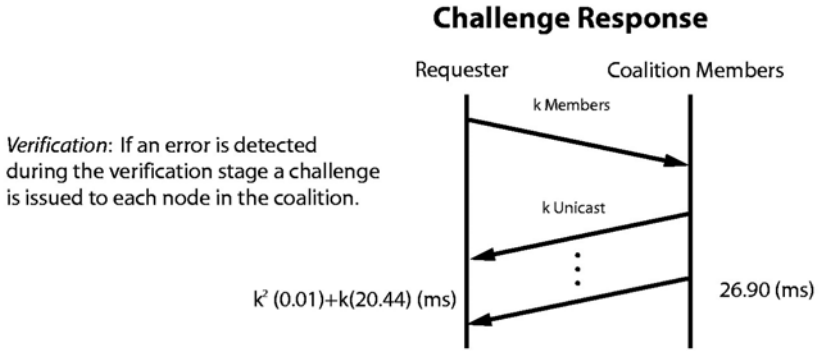

Figure 4: Timing diagram of verification stage for distributed self-initialization.

5. Distributed Certificate Authority with a Group-Cast Protocol

Our targeted mobile wireless ad hoc networks have node densities that vary over the life of the network. For example, when some networked systems are deployed they may have high node densities where each node may have direct communication with a large number of other nodes. As time progresses the network may become less dense with nodes moving away from each other to increase their spatial coverage. This results in nodes not having direct end-to-end communication with each other but multihop communication with other nodes. Since the node density varies over the life of the network, it is necessary to have a means to control the number of nodes that are requested to participate in distributed certificate renewal or distributed self-initialization.

We have developed an approach that facilitates a *distributed* CA that operates in a wireless ad hoc network with time-varying node density. Our approach supports dynamically changing network topologies that can result in a wide range of node densities around a node that requests a certificate renewal or self-initialization. The approach controls the number of nodes that should receive the certificate request operation. This number of nodes should be large enough to obtain a sufficient number of certificate share replies. Since mobile wireless ad hoc networks have changing topologies, their node densities will be variable. Thus, the network should have capability to adaptively increase or decrease the propagation depth as necessary to obtain the certificate share replies.

Our approach facilitates the distributed CA described in the previous sections. The approach facilitates a *(k, n)*-threshold signature scheme. With this scheme a coalition of *k* nodes can serve as the CA and jointly provide certification services for a requesting mobile node. Coalition of fewer than *k* players cannot produce a valid certificate. This scheme is intrusion-tolerant even in the presence of up to *k*-1 malicious players.

5.1 Group-Cast Distributed Certificate Request Description

When a communicating node is required to provide a signed certificate, it initiates a request to a coalition of nodes by sending the request as a group-cast. This group-cast is received by members of the group who collectively represents the CA. The particular group-cast is recognized to be only those nodes that have the capability to sign a certificate-share. For purpose of discussion, these nodes will be referred to as *Node Level 2 (NL2).* The requesting node, which initiates the signature request, will be referred to as *Node Level 1 (NL1).*

The first signature request will be sent out by an initiating NL1 node using a group address (that addresses all NL2 nodes within the local broadcast area). A type value field in this transmission frame will alert all NL2 nodes in the area that receive the transmission to restrict this transmission to the current receiving domain and not propagate this request to other, more distant nodes located outside of the current transmission domain. This transmission request will also have an associated sequence number that uniquely identifies this request. This sequence number will be cached for a pre-determined period of time and can also be used to detect transmission loops within the transmission domains, and prevent forwarding of identical packets. When the node that initiated the requests for a certificate receives all responses within a specific time-out period, it compares the number of responses with the needed threshold value of *k*. If the *k* threshold has been met, the node combines the signed certificate-shares to create the desired certificate needed for authentication.

If the threshold value has not been met, a subsequent transmission is sent out. The subsequent transmission is sent out using the same group address as the first transmission. It uses the same sequence number as the first transmitted frame, which allows the first group of nodes to determine whether this is a subsequent request from the original node. The type value of this frame will be different so it can be identified as a *forwarding request*. The type value when received by the original responders directs these responders to forward this *group-cast* to potential NL2 nodes that are in more distant regions than where the original request was propagated. These distant nodes, upon reception of the request, compute the signed certificate-share and transmit the result to the NL2 nodes that originally forwarded the request via a unicast transmission. These NL2 nodes then transmit the results back to the originating NL1 node.

When the initiating NL1 node receives this second wave of *k* signed certificate-shares (within an appropriate time-out period), it compares the number of additional signatures with the threshold value. If the value is met or exceeded, it combines and computes the needed certificate. If the threshold value is still not met, the node then aborts its request. Figure 5 shows this process with the first request not having received enough responses and thus requiring a second request that propagates into Domain B.

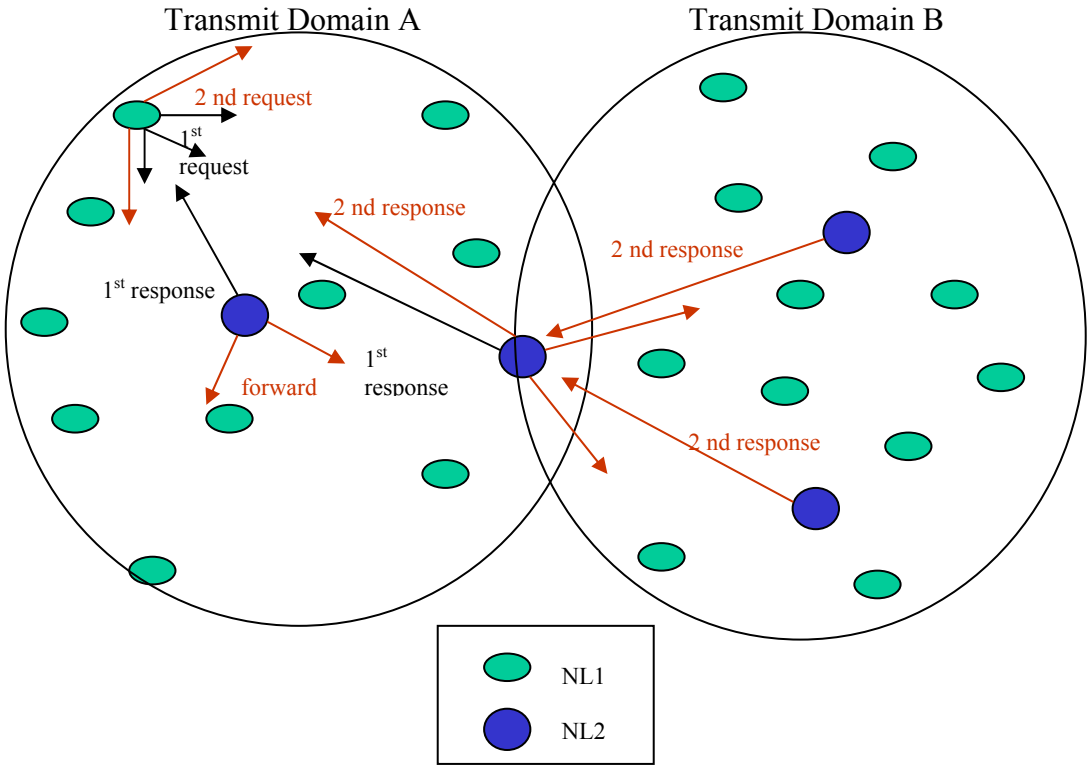

Figure 5: Group-cast certificate-share request with the first request not receiving enough responses and thus requiring a second request to propagate into Domain B

In some ad hoc networks, the operational environment and/or scenario may result in sparsely populated nodes that can provide distributed CA services. In some cases, these networks may also require a moderate or large *k* threshold value for security reasons. The need for this type of operation may be attributed to excessive amounts of environmental clutter (hills, mountains, trees, buildings, or weather) that effectively reduce the overall communication range of each participating node. This may also be due to the limited number of participating CAs due to limited deployment, large coverage area, node failures, or adversarial destruction and/or compromise.

To provide for the sparsely populated node scenario, an additional type field value is added to the CA request frame that allows the initiating NL1 node to request certificate-shares from multiple transmission domains on its first request cycle. This allows for the request frame to be propagated through a predetermined number of different transmission domains on the first request without waiting for an initial request from a single domain to time out. As with the default frame, this request frame will be associated with a unique sequence number to allow for the detection of transmission loops. Upon reception, NL2 nodes will act upon this request immediately. The original NL1 requesting node will wait a specified period of time for the required number of responses before timing out and aborting the CA request. This time-out value is necessary to prevent unserviceable requests from accumulating at the node. The dynamic

topologies of our targeted applications may at times partition, resulting in requests that may not be serviceable. Figure 6 shows this Sparse Certificate Authority Scenario (SCAS) scenario.

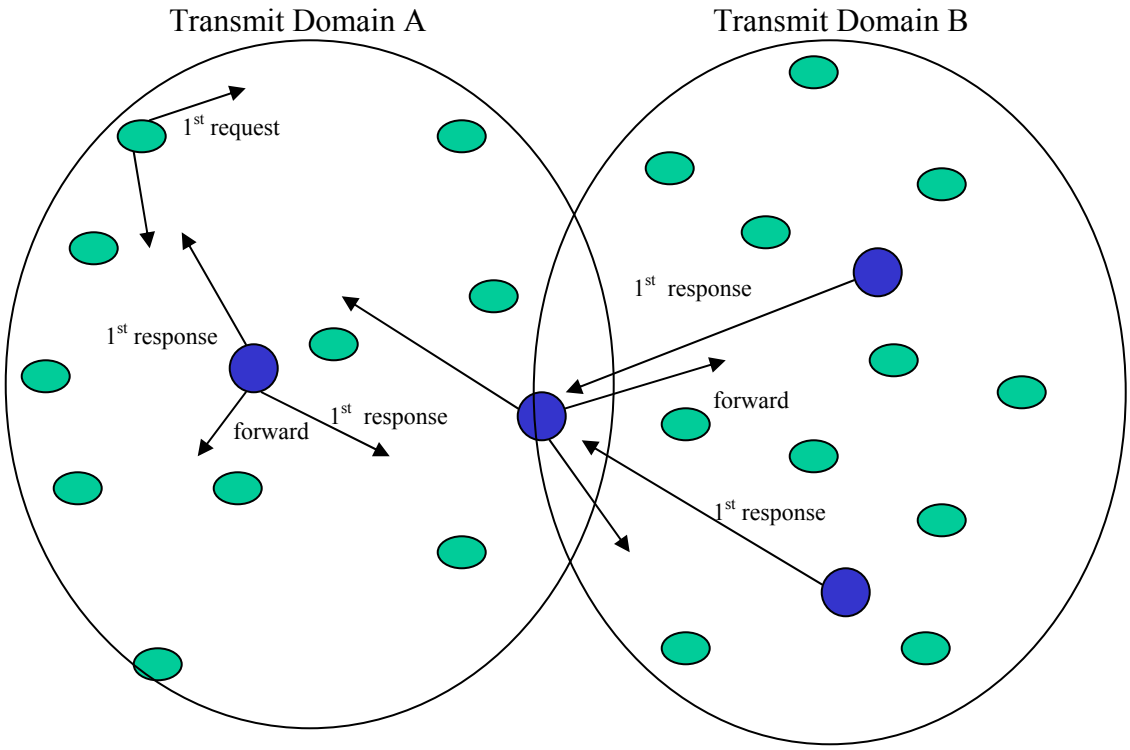

Figure 6: Sparse Certificate Authority Scenario

5.2 Group-Cast Distributed Certificate Protocol Attributes

For effective distributed CA services in mobile wireless ad hoc networks, several important operational attributes must be associated with the group-cast distributed CA protocol. These attributes will help facilitate the cryptographic algorithm needed to provide the network CA services.

- *Group-cast certificate request mode* allows for a certificate request to be sent to a group of nodes that have the capability of signing a certificate-share. This provides a much more efficient means of distributing a certificate request to all participating NL2 nodes.

- *Dense certificate request mode* is an automated process that specifically addresses individual certificate-signing capable end nodes. This allows for a more direct method of requesting a signed certificate when the *k* value of the request is small and the number of certificate-generating end nodes is large.

- *Selectable certificate request throttle* is a user-defined attribute that provides the means to change automatically from a group-cast mode to the dense-certificate-request mode. The number of nodes that respond to a certificate request is compared to a threshold value. When the threshold value is exceeded, the next certificate request cycle uses the dense-certificate-request mode.

- *Selectable outreach mode* allows for the user to define the number of forwarding domains that a certificate request will traverse. A domain is defined as the region a group of nodes can be separated and still maintain wireless communications. The protocol has a domain value that instructs appropriately addressed NL2 nodes to forward a CA request.

- *Use- selectable k threshold value* defines the number of certificate-shares that are needed to provide a signed certificate. This user-selectable feature balances authentication security with operational efficiency. Larger *k* values result in a more secure network since a larger number of nodes must be compromised to capture a key. However, larger *k* values result in increases network traffic and more processing for the certificate-share combining node.

- *Stale request detection* attribute provides a means to discard certificate requests that have expired. When a node is operating in outreach mode, subsequent forwarding transmissions must be cached until a response is received. If there is no response prior to the timed cached packet, the packet is discarded. All responses that are received after a timeout has occurred are ignored.

- *Transmission loop detection* attribute identifies all redundant certificate requests within the broadcast domain and prevents the continual forwarding of these requests. When a requesting NL1 node has set the domain byte to > 1, all receiving NL2 nodes must verify incoming request have not already been cached and forwarded.

- *Certificate Revocation List (CRL) cache and dispense function* provides a means of caching and distributing all the names of nodes that have been identified as compromised.

5.3 Group-Cast Distributed Certificate Protocol Model and Simulation

To analyze the performance of the group-cast distributed certificate protocol design, we developed and performed several simulation experiments. The simulation experiments are intended to analyze and demonstrate the operation of the protocol. Our simulations include the effects of the wireless channel, such as path loss, channel interference, and channel contention issues. We examine the performance of the protocol independent of any application that operates on the node.

The simulation software used for this research was the OPNET Modeler and Radio simulation package Version 9.1a. Optimum Network Performance (OPNET) [8] is a comprehensive engineering system capable of simulating large communication networks with detailed protocol modeling and performance analysis capability. OPNET features include: graphical specifications of models; a dynamic, event-scheduled simulation kernel; integrated tools for data analysis; and

hierarchical, object-based modeling. OPNET analyzes system behavior and performance with a discrete-event simulation engine. Discrete-event simulation is an approach that supports realistic modeling of complex systems that can be represented as a progression of related events. This approach models system behavior based on objects and distinct events such as the arrival of packets at various points in a network. Each object has associated attributes that control its behavior in the simulation.

In OPNET, a node is a collection of interconnected modules in which data is manipulated as defined by the modules. Modules represent the internal capabilities of a node such as data creation, transmission, processing, internal routing, reception, storage and queuing. The modules are used to model aspects of node behavior. A single node model is usually comprised of multiple modules. The modules are connected together by packet streams and statistic wires. Packet streams are used to transport data between the modules while statistic wires allow one module to monitor a varying quantity within another module. The ability to integrate the use of modules, packet streams, and statistic wires allows the developer to create highly realistic simulations of node behavior.

Each node module contains a set of inputs and outputs, some state memory, and a method for computing the module's outputs from its inputs and its state memory. OPNET provides a module library that represents standard functions and protocols. To support development activities, the modules are open source and can be customized to represent user-defined behavior. In addition, completely new modules can be created to represent user-developed functions and protocols. Example OPNET modules are processors, queues, generators, receivers, and transmitters. The processor, queue, and generator modules are strictly internal to a node. The transmitters, receivers, and antennas have external connections to data transmission links. We developed modules in OPNET to represent behavior of the NL1 and NL2 nodes with the group-cast protocol.

5.3.1 Group-Cast Distributed Certificate Protocol Model

A node model was created to simulate the group-cast protocol in each of the NL1 and NL2 nodes. The node model includes functions to represent some of the attributes listed in Section 5.2. Our primary objective was to examine the basic protocol operation with a wireless physical layer. Figure 7 shows a wireless node model on which this protocol is implemented.

As shown in Figure 7 the node model is comprised of a transmitter and receiver with the necessary medium access control (MAC) module that controls and manages the access to the wireless medium. A wireless LAN interface manages the interactions between the two custom modules described below and the wireless MAC.

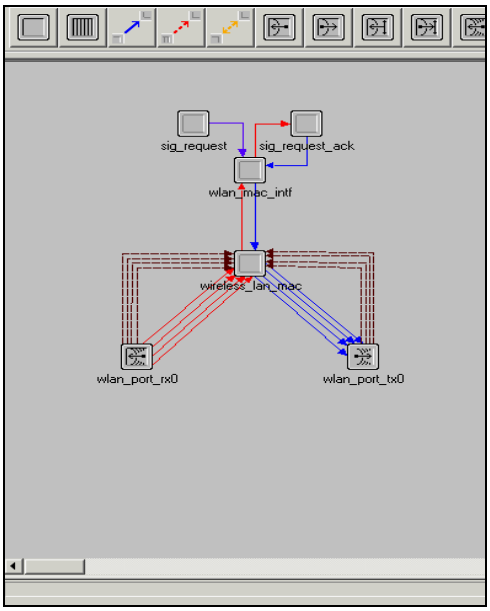

Figure 7: Wireless node model with group-cast distributed certificate protocol

The first step in the model development was to create a group-cast capability within the protocol. This required development of logic in the data-link layer of the wireless communications stage to differentiate between a broadcast transmission, a unicast transmission, and the newly created group-cast transmission. A logic filter was developed by inserting a new type field into the transmitting packet. Specific bits in the type field identify it as group-cast packet.

An additional field was created for the sequence number generated by the certificate-share-requesting node. A capability for NL2 nodes to cache this sequence number and the associated request information was also added. The sequence number uniquely identifies the transmission request needed to propagate through multiple domains. This field supports tracking the number of domains a request is forward and also supports mapping return packets to request packets. When a subsequent response is received from a previously forwarded certificate-share request, the sequence number is used to reference the cached information, and the response is sent back to the requesting NL2 node. If a request from an originating NL1 node is to remain local or within the same domain, the *out-reach mode* bit is not set and the packet is processed by a corresponding NL2 node and returned immediately. The sequence number is used at the receiving NL1 node to correlate different signature requests with their responses.

As shown in Figure 7 two custom modules implement the group-cast distributed CA protocol. The module *sig_request* (located within a designated NL1 node) sends out signature request packets at predetermined times, while the *sig_request_ack* module (located within a designated NL2 node) processes these requests.

Figure 8 shows the process model within the *sig_request_ack* module. This module is comprised of five logic states, INIT, IDLE, SIG REQ, SIG RESP, and Delay, developed to support the logic construct of the certificate request-processing block. Each of the states is described in Table 3.

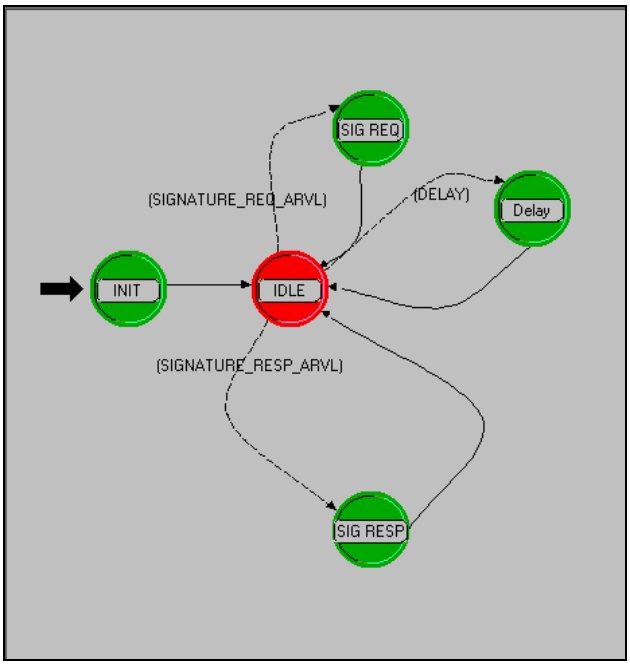

Figure 8: Process model within the *sig_request_ack* module

Table 3: Description of logic states for *sig_request_ack* module

State	Description
INIT	This state initializes the process model by reading the values of the source attributes associated with its node module. The attributes are associated with packet generation and include packet inter-arrival time, packet size, packet format, start time, and stop time. The packet-format attribute has been created to facilitate a signature request packet generated by a NL1 node.
IDLE	In this state, accessed after the initialization phase, all activities remain in an idle state until one of three interrupts occur: an arrival of a signature request packet, an arrival of an signature response packet, or the expiration of a self interrupt generated by the SIG REQ block when inserting an appropriate time delay for processing a certificate-share.
SIG REQ	This state is accessed when a stream interrupt occurs with a signature request packet. Information within the packet is read and displayed to demonstrate protocol operation. The logic differentiates between a local signature request packet (within a single domain) and an extended outreach packet. A local request packet incurs a processing delay that represents the certificate-share generation process that supports the distributed CA architecture. The actual processing delay depends on the cryptographic algorithm and the processing speed of the CPU. For a more detailed description of a distributive cryptographic certificate scheme and its associated processing delay, see Section 5.

SIG RESP	This logic state is responsible for the creation and transmission of a signature response packet back to the requesting NL1 node. The packet must contain the originating sequence number as part of the data payload. After the incoming packet is identified as a signature request packet, it is sent to the Delay processing block.
DELAY	After an appropriate processing delay has occurred (based on calculated processing delays defined in Section 5), the DELAY block calls the *ss_packet_generate_1* function that creates and sends a response packet back to the originating NL1 node.

5.3.2 Group-Cast Distributed Certificate Protocol Simulation

A simulation scenario was created that had a single NL1 node that periodically transmitted certificate-share requests. The NL1 node transmitted a group-cast packet on the wireless medium, and any nodes that were tuned to the wireless channel had the potential to receive the request. Several factors determined if a node actually received and processed the request. First, each receiving node examines the group-cast transmission for signal strength and compares it to the necessary received signal-to-noise ratio. If this condition is not met, the request is dropped. During the reception, the receiver continuously checks that no other packet transmission causes interference resulting in a disrupted attempt at receiving the request. If a packet collision occurs, the request is dropped. Finally, the receiving nodes must examine the request packet to determine whether they are a member of the group. A screen capture of the simulation is shown in Figure 9.

To examine the protocol under dynamic wireless communication conditions, all nodes were given random mobility during the simulation scenario. This included the effects of the limited transmission range of the NL1 node, resulting in not all transmitted requests being received by all NL2 nodes. Both the NL1 and NL2 nodes included a movement algorithm that was based on a random start location and random vector direction. The nodes then traversed their assigned vectors at a rate of 5 meters/second. When they reached the end of their plotted vector paths, they would pause briefly until another random vector was plotted and then proceed in a new direction. The receiver and transmitter parameters were set to have a communication range of 300 meters.

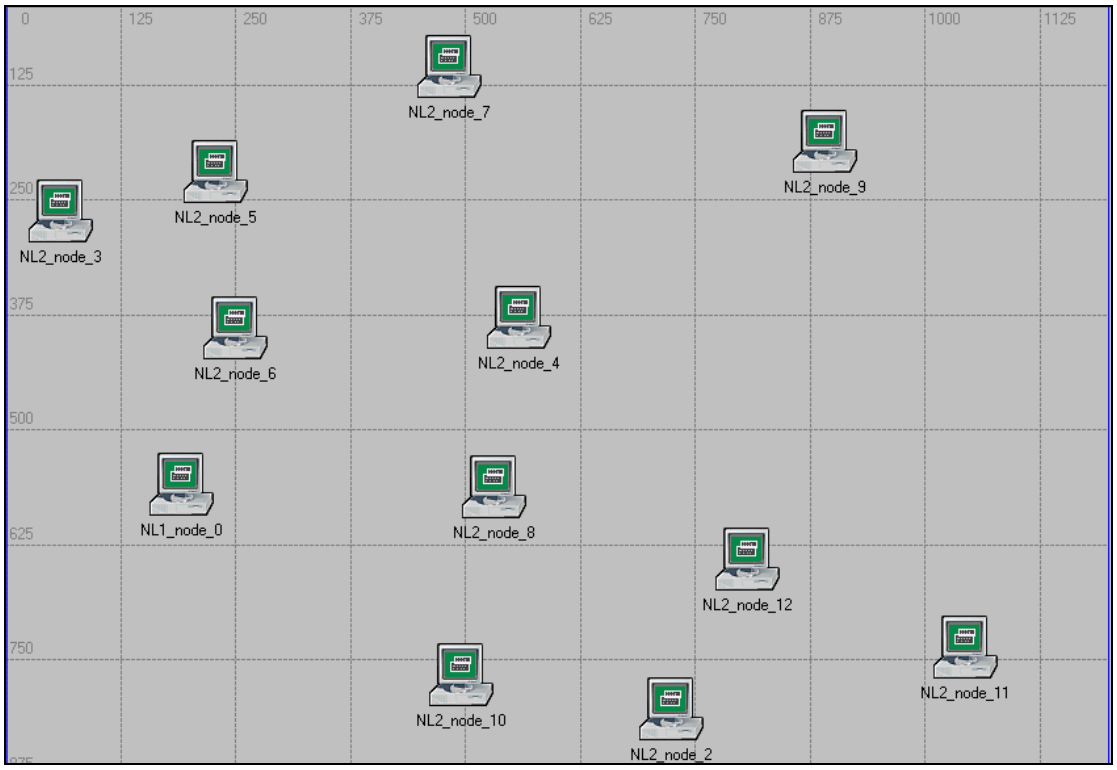

Figure 9: Screen capture of group-cast distributed CA simulation

To illustrate the dependence on node density, three scenarios were examined, each with a different NL2 node density. Each of the scenarios included 10 NL2 nodes randomly placed and moved on a two-dimensional grid. The grid dimensions were 1) 1.0-km by 1.0-km, 2) 2.5-km by 2.5-km, and 3) 5.0-km by 5.0-km. The scenarios were simulated, and the number of certificate-share responses was recorded for each periodic request.

The results of the three scenarios are plotted together in Figure 10. These results clearly demonstrate the dependence of certificate-share returns for each request on node density. For the high density case an average of 3.6 certificate-shares are returned for each request. For the medium and low density cases an average of 1.2 and 0.1, respectively, certificate-shares are returned for each request. The high density scenario demonstrates a network that would be functional if $k \leq 3$ and the certificate renewal had greater than five periods to update. The medium density scenario is operational for a value of $k = 1$; however, a more conservative design would suggest a slightly higher density. Obviously, with a $k = 1$ the certificate authority is not considered distributed and suffers from all the drawbacks of a single certificate authority. Clearly the low density scenario is non-functional since most of the time no certificate-shares are returned.

The scenarios also demonstrate the value of a group-cast distributed CA request protocol. In our targeted applications there typically are periods of time, such as when they are deployed, that node density is high. A certificate-share request may get a much larger number of responses than

that necessary to create a certificate. Each of the unnecessary responses consumes network resources and thus reduces network performance. A protocol that can control the depth of certificate-share request propagation will support increased network performance.

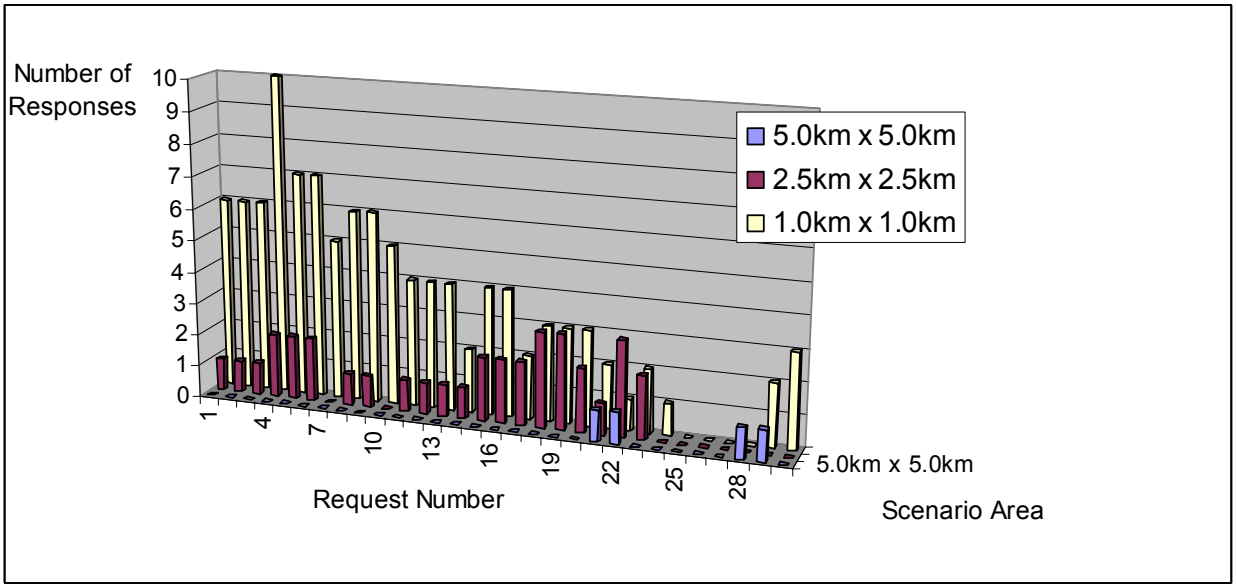

Figure 10: Number of certificate-shares returned for each request

6. Future Work

This work has developed two concepts to enhance the performance of a distributed CA in a mobile wireless ad hoc network. Our initial analysis of each of the concepts shows system performance improvements. A complete integration of the described concepts into a distributed CA is expected to realize further gains. The following future work is necessary to complete the integration into an operational network.

The group-cast distributed CA protocol should be extended to incorporate the full list of attributes described in Section 5.2. When the full list of attributes is employed in the protocol, we will be in a position to facilitate the distributed CA services described in earlier sections of this report. In addition, the verification processes described in Section 4 should be tightly integrated with the group-cast protocol.

7. Conclusion

Mobile wireless ad hoc networks that are resistant to adversarial manipulation are necessary for distributed systems used in military and security applications. Critical to the successful operation of these networks operating in the presence of adversarial stressors are robust and efficient information assurance methods. In this research, we developed enhancements for protocols that

are applicable to secure architectures employing distributed certificate authorities. We first described a means to discover a compromised node that attempts to disrupt network operation by contributing bogus shares. The method is efficient in the fact that it is initiated only if an invalid certificate is generated. Additionally, the beginnings of an efficient certificate-share request protocol are described. Timing values are presented to provide insight into the impacts of network performance when the described methods are used in a network design.

8. Bibliography

[1] B. Blakley, *Safeguarding Cryptographic Keys*. American Federation of Information Processing Societies Proceedings, pgs. 313-317, 1979.

[2] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, *Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults*. FOCS, pgs. 383-395, 1985.

[3] W. Diffe and M. Hellman, *New directions in cryptography*. IEEE Transactions on Information Theory, November 1976

[4] H. Luo, S. Lu, *Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks*. UCLA-CSD-TR-200030, October 2000.

[5] P. Feldman, *A Practical Scheme for Non-interactive Verifiable Secret Sharing*. FOCS, pgs. 427-437, 1987.

[6] L. Gong, *Increasing availability and security of an authentication service.* IEEE Journal on Selected Areas in Communications, Vol.11, No.5, June, 1993, pp.657-662

[7] J. Kong, H. Luo, K. Xuo, D. Lihui Hu, M. Gerla, and S. Lu, *Adaptive Security for Multi-layer Ad-hoc Networks.*

[8] OPNET Technologies. *OPNET Users Manual*, www.opnet.com.

[9] Safenet Inc., EmbeddedIP-23, *Public Key Accelerator*, Document Number: EIP/2002/0014, Revision 1.5, 21 February 2003.

[10] Safenet Inc., EmbeddedIP-61, *SHA-1 Accelerator*, Document Number: EIP/2002/0012, Revision 1.1, 5 September 2002.

[11] Safenet Inc., EmbeddedIP-71, *RNG Core*, Document Number: EIP/2002/0010, Revision 1, 18 April 2002.

[12] B. Schoenmakers, *A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting*. in Crypto '99, 1999.

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

This page intentionally left blank.

DISTRIBUTION:

	MS	
5	0785	W. E. Anderson, 6514
1	1004	R. W. Harrigan, 15221
1	0785	R. L. Hutchinson, 6516
1	0785	A. J. Lanzone, 6514
1	1004	M. J. McDonald, 15221
1	0785	T. S. McDonald, 6514
5	0785	J. T. Michalski, 6516
1	1004	F. J. Oppel III, 15221
1	0455	R. D. Pollock, 6501
1	1170	R. D. Skocypec, 15310
1	0455	R. S. Tamashiro, 6517
1	0785	M. D. Torgerson, 6514
1	0784	R. E. Trellue, 6501
5	0785	B. P. Van Leeuwen, 6516
1	0451	S. G. Varnado, 6500
1	0323	LDRD Program Office, 1011 (Attn: Donna Chavez)
2	0899	Technical Library, 9616
1	9018	Central Technical Files, 8945-1