A Framework for Providing Redundancy and Robustness in Key Management for IPsec Security Associations in a Mobile Ad-Hoc Environment

George C. Hadjichristofi

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Dr. Nathaniel J. Davis, IV, Chair
Dr. Scott F. Midkiff
Dr. Luiz A. DaSilva
Dr. Ira Jacobs
Dr. Ezra A. Brown

June 28, 2005
Blacksburg, Virginia
Keywords: IP Security, Key Management, MANET, Performance

A Framework for Providing Redundancy and Robustness in Key Management for IPsec Security Associations in a Mobile Ad-Hoc Environment

George C. Hadjichristofi

(ABSTRACT)

This research investigated key management in a Mobile Ad Hoc Network (MANET) environment. At the time this research began key management schemes provided limited functionality and low service availability in a highly partitioned ad hoc environment. The purpose of this research was to develop a framework that provides redundancy and robustness for Security Association (SA) establishment between pairs of nodes.

The key contribution of this research is the Key Management System (KMS) framework and, more specifically, the unique way the various components are integrated to provide the various functionalities. The KMS overcomes the limitations of previous systems by (1) minimizing pre-configuration, (2) increasing service availability, (3) and increasing flexibility for new nodes joining the network. A behavior grading scheme provides the network with a system-wide view of the trustworthiness of nodes and enables the KMS to dynamically adjust its configuration according to its environment. The introduction of behavior grading allows nodes to be less dependent on strict identity verification. This KMS was simulated with Monte Carlo and NS2 simulations and was shown to interoperate with IP Security (IPsec) to enable the establishment of IPsec SAs. The simulations have proven the effectiveness of the system in providing service to the nodes in a highly partitioned environment.

Acknowledgements

This research effort was the result of guidance and support from a number of people and I wish to thank them.

I thank my advisor, Dr. Nathaniel Davis for his guidance throughout this research as well as his invaluable help during the preparation of this document.

I thank the members of my advisory committee, Dr. Scott Midkiff, Dr. Luiz DaSilva, Dr. Ira Jacobs, Dr. Ezra Brown for their gracious service. I especially thank Dr. Scott Midkiff for introducing me to this research and helping me whenever I needed additional advice.

I thank Dr. David de Wolf for helping me and giving me advice prior to starting my Ph.D.

I thank all the friends that helped me over the years, especially Palan Annamalai, Joe Adams, Tao Lin, Unghee Lee, Karthik Channakeshava, Enrico Loddo, Matt Anderson, Kaustubh Phanse, Michelle Gong, Animesh Patcha, Dr. Jahng S. Park, Creighton Hager, Fahad Koujah, John Wells, Clark Gaylord, Michael Thompson, and Constantinos Phanouriou.

I thank my parents Costas and Chrystalla, my brother Stavros, and my sisters, Maria and Despo, for their love and support all these years. I thank my wife, Stella, for her patience, love, and understanding throughout this process.

This work was supported in part by the Office of Naval Research through the Navy Collaborative Integrated Information Technology Initiative (NAVCIITI) and Advanced Wireless Integrated Navy Networks (AWINN). This support is gratefully acknowledged.

TABLE OF CONTENTS

List of Figures

List of Tables

Glossary of Acronyms

3DES	Triple Digital Encryption Standard
AH	Authentication Header
AODV	Ad hoc On Demand Distance Vector
API	Applications Programming Interface
CA	Certificate Authority
COPS	Common Open Policy Service
CRL	Certificate Revocation List
DCA	Delegated Certificate Authority
DES	Digital Encryption Standard
DH	Diffie Hellman
DNS	Domain Name Server
DOI	Domain of Interpretation
DoS	Denial of Service
ESP	Encapsulation Security Payload
GN	Generic Node
GRE	Generic Routing Encapsulation
GUI	General User Interface
HMAC	Keyed-Hashing Message Authentication Code
ICV	Integrity Check Value
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	IP Security
ISAKMP	Internet Security Association and Key Management Protocol
KDC	Key Distribution Center
KMS	Key Management System
MAC	Message Authentication Code
MANET	Mobile Ad Hoc Network
MD5	Message Digest 5
MIC	Message Integrity Code

MIPv6	Mobile IPv6
MITM	Man in the middle
OE	Opportunistic Encryption
OLRS	On-Line Revocation Server
OLSR	Optimized Link State Routing
OSI	Open Systems Interconnections
OSPF	Open Short Path First
PEM	Privacy Enhance Mail
PFS	Perfect Forward Secrecy
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
QoS	Quality of Service
RA	Registration Authority
RADIUS	Remote Authentication Dial-In User Service
RCA	Root Certificate Authority
RSA	Rivest Shamir Adleman
SA	Security Association
SADB	Security Association Database
SAR	Security–aware Ad hoc Routing
SASL	Simple Authentication and Security Layer
SDCA	Supporting Delegating Certificate Authority
SDN	Service Discovery Node
SHA1	Secure Hash Algorithm 1
SKEME	Secure Key Exchange Mechanism
SKIP	Simple Key management for Internet Protocol
SP	Security Policy
SPD	Security Policy Database
SSH	Secure Shell
SSL	Secure Socket Layer
TCA	Temporary Certificate Authority
TCP	Transmission Control Protocol
TLS	Transport Layer security
TP	Trusted Peer
VON	Virtual Operations Network

VPN	Virtual Private Network

Chapter 1

Introduction

Data communication has undergone significant advances due to the development of wireless technology. Wireless networks can now accommodate users with higher bandwidth needs at a lower cost, which makes these networks more attractive to a wider variety of applications. However, this shift to wireless communications has spawned new demands for functionality, especially in MANETs.

MANETs are autonomous systems of mobile routers and associated hosts connected by wireless links in dynamic topologies [1]. They are deployed in environments where costs, environmental constraints, or other limitations require a self-organized solution. Users in a MANET dynamically connect to their peers to exchange information. Some of this information may be confidential and may need to be communicated to specific people or groups of people. This desire to communicate secrets gives rise to the need for encryption to keep data confidential. However, in order for two peers to establish a secure path that would provide the desired privacy, there should be a configured level of trust or Security Association (SA) between the users. This level of trust is achieved via authentication. Using authentication, people verify each other's identity by showing proof of their identity with a key or certificate. These keys or certificates can be distributed dynamically to the nodes automatically via a key management system (KMS). Traditional centralized KMSs do not operate properly in a MANET environment since MANETs have no infrastructure and are characterized by unpredictable connection links, node failures, and security vulnerabilities (see Section 2.1). This research effort focused on improving the existing methods of distributing authentication information among nodes in a MANET so that they can build SAs between them and securely exchange sensitive information.

1.1. Problem Statement

This research investigated ways of providing redundancy and robustness for key management to facilitate the establishment of Internet Protocol Security (IPsec) SAs in a MANET. Key management is the secure generation, distribution, maintenance and storage of keys on network nodes. The specific problem addressed was that KMSs, which facilitated trust establishment among nodes in a *wired* network were not suitable for MANETs. These KMSs were limited in a number of ways:

- They were not as flexible to adjust to the dynamic changes in topology as required in a MANET.
- They relied on centralized distribution of certificates or revocation lists, which was not scalable.
- Certification was mainly conducted off-line instead of dynamically as desired in a MANET.

1.2. Background and Motivation

Trust within a network is established when users show proof of identity to authenticate themselves. This proof can be a certificate, which is obtained from a Certification Authorities (CA) or it can be proof of possession of a pre-shared key. CA-certificates use asymmetric or public/private keys, whereas pre-shared keys typically use symmetric keys.

The set up and maintenance of these keys on network nodes may be done manually (off-line) or automatically (on-line) via the use of a trusted key distribution mechanism. Setting up keys manually avoids any security issues related to automatic key management on the network. However, manually setting up keys is not scalable and may not function properly in a dynamic environment such as a MANET.

This research was motivated by the lack of suitable KMSs that are required prior to the establishment of trust between nodes in a MANET and deployment of a security mechanism, such as IPsec.

The authors of [2],[3],[4], and [5] proposed that a public/private key pair be kept for the entire key management service. The private key was shared among a number of nodes, which in turn used it to generate a partial signature. A combiner then computed the certificate from the partial signatures. The authors assumed that there was a trusted authority that empowered the arbitrarily chosen nodes to behave as trusted CAs. This scheme was more applicable to less partitioned MANETs (more densely connected) in which there was a higher probability that the required number of CAs to build a certificate were reachable in a timely manner.

The authors of [6] through [10] proposed the use of a self-organized KMS that allowed all the nodes in a MANET to issue certificates for one another. This system was heavily dependent on the use of certificate revocation to mitigate attacks from compromised nodes. A user was authenticated by building chains of trust, which are inherently weak as they rely on the correctness of every authority on the chain. In addition, the system was limited by a costly initialization phase.

The authors of [11] and [12] worked on the address ownership problem in IPv6. They proposed that the public key be used to derive part of the IP address of a node in a cryptographically verifiable way. Thus, the usage of certificates was avoided. Even though this approach suggested that there may not be any need for certificates it had some limitations. It did not deal with issues such as proof of user identity, certificate revocation, and hierarchy of trust. Furthermore, it could not be applied to IPv4 addresses.

A more detailed explanation of related work is given in Section 2.9.

1.3. Research Questions

This research addressed the following questions regarding KMSs in a MANET environment.

1) What is the current state of the art of the technology (e.g., IPsec) for which the KMS is proposed?
2) What are the limitations of the existing KMSs proposed for MANETs and how can those limitations be mitigated or eliminated without significantly diminishing the strengths of those systems?

3) What are the trade-offs between security and functionality of the proposed KMS in a MANET environment and how does this system compare to existing ones?
4) What is the probability of issuing/reissuing a certificate to a node when the centralized CA is available, (without the proposed key management/distribution system in place)?
    i. How does the number of on-line CAs affect this probability?
    ii. How do the various network parameters affect this key management function?
5) What is the probability that a node will be able to obtain the certificate of another node to negotiate an SA? This functionality deals with the overall distribution of the certificates in the network based on CAs and repositories.
    i. How does the number of on-line CAs and repositories affect this probability?
    ii. How do the various network parameters affect this key management function?
6) What is the probability that a CA will be able to communicate with a given percentage of nodes over a period of time when revoking a certificate? The probability of revocation was calculated by measuring the success of distributing the revocation notices (RNs) to a group of nodes and the time required for this process.
    i. How do the various network parameters affect this key management function?
7) What is the overhead imposed by this system? The performance metrics will be the network load and number of transactions.

1.4. Objective

In a MANET, not all of its members can communicate directly with each other. Thus, they form multiple clusters that enable them to communicate with one another. If a cluster is isolated from the rest of the clusters in the network, then the network is

partitioned and the cluster is referred to as a partition in the network. The objective of this research was to develop a KMS that was suitable for a highly partitioned MANET environment. The KMS overcame the limitations of previous systems (Section 2.9). It increased service availability in a highly partitioned network environment, minimized pre-configuration of KMS's nodes, and accommodated new nodes joining the network. Moreover, it provided a means of obtaining feedback based on the network activity to dynamically adjust its configuration. In addition, it interoperated with an existing implementation of IPsec, specifically FreeS/WAN IPsec [13] .

1.5. Methodology Overview

The current state of technology of a Linux IPsec implementation and key management were investigated through deployment on a testbed. A KMS was formulated based on the findings of the IPsec deployment and the investigation of existing KMSs proposed for MANETs. The detailed transactions of the KMS were formulated and the system was simulated with both a Monte Carlo simulation analysis and a Network Simulator 2 (NS2) simulation to assess its performance and scalability. The Monte Carlo analysis utilized a uniform distribution of nodes and no node mobility, whereas the NS2 simulation utilized a non-uniform distribution of nodes and accounted for the impact of node mobility [14][15]. Analytical modeling of the KMS was impractical given the complexity of the system. The KMS was implemented on a testbed to prove its interoperability with an existing IPsec implementation and its effectiveness in increasing service availability.

1.6. Significant Results

The proposed KMS used a modified hierarchical structure. The system increased service availability in a partitioned environment by using multiple Delegated Certificate Authorities (DCAs). (For the purpose of this KMS, the name convention for the CAs was DCAs signifying the existence of a Root Certificate Authority (RCA).) Nodes communicated with one of the multiple DCAs and issued their certificates without

necessarily meeting physically to exchange keys, i.e., without ever being connected by a physical path. This functionality was achieved by assigning trust levels on the certificates based on the method of authentication with the CA. In the absence of DCAs, nodes could still establish SAs by obtaining other nodes' certificates from their Trusted Peers (TPs), thus further increasing service availability. The TPs of a node were the nodes that were trusted by that particular node (shared an SA). The notion of TPs was directly mapped to the existing IPsec implementation and facilitated integration with the KMS.

The KMS was flexible enough to accommodate new nodes joining the network by utilizing Temporary Certificate Authorities (TCAs) that generated temporary certificates for their collocated peers. In addition, the system minimized pre-configuration since any node that had a root certificate could register into the network and serve as a DCA. This approach was more dynamic as compared to other systems that required all CAs to share information about one another.

The components of the KMS were secured via integration with a behavior grading scheme. The behavior-grading scheme recorded the number of TPs of a node on each node's certificate and distributed this information to the rest of the network. Based on this information the DCAs could choose to revoke particular nodes and individual nodes could chose to trust or distrust their peers. The transactions of the KMS were structured in such a way so that no single node could maliciously change this information, thus providing balance of powers.

Monte Carlo simulation analysis demonstrated that if 10% of the nodes were DCAs then there was at least an18% increase in the probability that a node could issue its certificate as compared to a centralized system. In addition to the 10% DCAs, if a node was trusted by 10% of the nodes (TPs), then there was at least a 35% increase in probability that a node could obtain its peer's certificate and establish an SA. Similarly the NS2 simulation [16] analysis demonstrated that if 10% of the nodes were DCAs then there was at least 14% increase in the probability that a node could issue its certificate in a highly partitioned network as compared to a centralized system. If a node was trusted by 10% TPs (combined 20% TPs and DCAs), then there was at least a 25% and 43% increase in probability that a node could establish an SA with its peer for OLSR and

AODV, respectively. The overhead of the KMS could be dynamically changed depending on the network-wide and node security policies as well as on the nodes' power constraints (see Section 5.3).

1.7. Summary

This research focused on providing a system that enabled nodes to establish trust between them in a MANET. Currently proposed systems make invalid assumptions about the trustworthiness of nodes, provide limited functionality and are not flexible enough to accommodate new nodes. The research questions addressed in Section 1.3 were answered by investigating the current state of existing technology and formulating an interoperable system with existing technology. The system was simulated to investigate its effectiveness in increasing service availability. This research provided a framework for key management in a MANET that was robust and increased functionality as compared to existing schemes.

This chapter has introduced the research problem. The remainder of this dissertation describes, in detail, how the solution to this problem was developed. Chapter 2 presents background knowledge and a review of relevant literature. Chapter 3 gives the findings from the investigation of the existing state of technology related to IPsec and key management. Based on these findings, a solution is then developed and explained in detail in Chapter 4. Chapter 5 presents the analysis and results of the evaluation of the proposed KMS. Finally, Chapter 6 summarizes the research findings.

Chapter 2

Background and Literature Review

This chapter provides an overview of background knowledge and presents relevant existing literature for the proposed research problem. Section 2.1 describes the general characteristics of a MANET environment and compares them with those of traditional wireless and wired networks. The comparison is conducted on the basis of functionality and security. Section 2.2 gives an overview of cryptography. The main focus is to define the essential elements that are required in a secure environment and to describe the basic cryptographic functions and authentication methods used. Section 2.3 examines the security mechanisms available at the various Open Systems Interconnect (OSI) layers. An overview of IPsec is then given in Section 2.4 followed by a description of currently implemented network security systems in Section 2.5. Section 2.6 explains the reasons for considering IPsec for this research. Section 2.7 gives an overview of authentication in IPsec and Section 2.8 analyzes common ways of setting up, or distributing keys in a network using symmetric and asymmetric keys. Finally, Section 2.9 presents previous related work.

2.1. MANET Definition

"A MANET is an autonomous system of routers (and associated hosts/nodes) connected by wireless links -- the union of which forms an arbitrary graph" [1]. The routers move freely, changing the network's wireless topology dynamically and new nodes join or leave the network at any time.

A MANET may operate by itself as a separate entity or may operate as a hybrid fixed/ad-hoc network that is connected to the Internet. This research concentrated on a MANET being a separate entity, since connection to the Internet may infer the ability to obtain required information, such as keys, without the need of an additional KMS in

place. For the purpose of IPsec, each individual node in a MANET may be considered as a single node or a group of nodes connected via wireless links. An example of a MANET is shown in Figure 2.1. Initially nodes A, B and C are directly connected with each other and node D is connected to nodes A and B through C. After some time, node A moves out of range of nodes B and C but retains connectivity to the rest of the nodes via node D. In addition, node E joins the network and is connected via C.

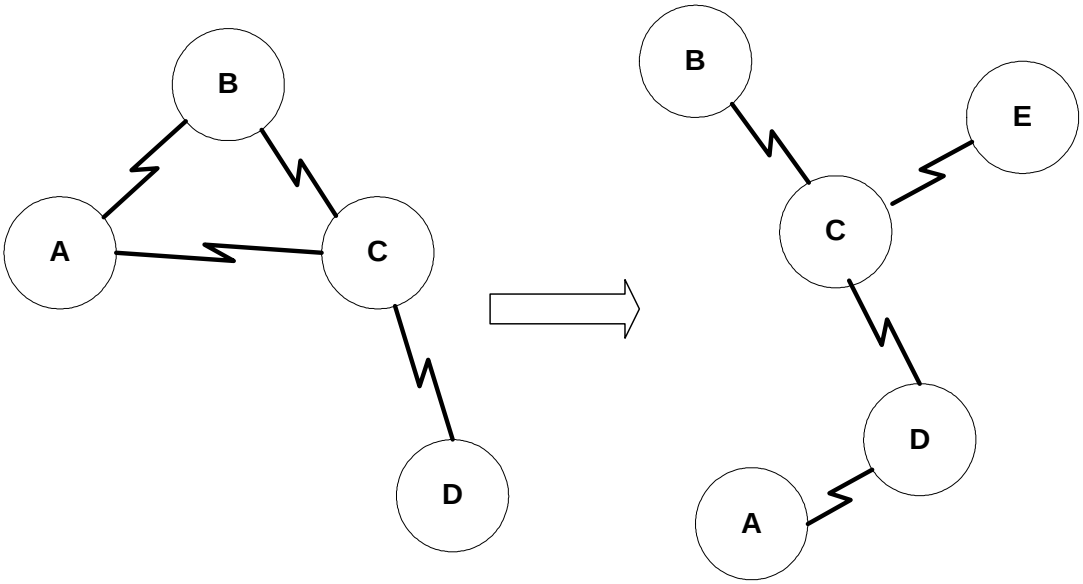


Figure 2.1. MANET topology.

A MANET can be deployed rapidly so it becomes an attractive alternative for a wide variety of applications. It can be used by soldiers in the military, for disaster relief operations, for expeditions, and by the media.

2.1.1. MANET Environment Characteristics

The inherent nature of MANETs has changed the way networks are organized and operated. MANETs have no infrastructure and require cooperation among nodes to provide the various functionalities required within the network. The links in a MANET are wireless thus, there are more bandwidth limitations and there is a higher probability of data error (versus wired links), which lowers the rate of information transfer. Software that handles traffic on the nodes needs to be aware that loss of data is no longer

necessarily due to congestion as in wired networks, but it is likely due to high data error rates. In cases where the topology is very dynamic, the stability of the links tends to be lower as compared to fixed wireless or wired links. Functions such as routing need to quickly propagate changes to avoid extensive data loss. The nodes also tend to be battery-powered, which imposes additional constraints on their functionality and processing capabilities within the network. Lastly, MANET end-users have extra responsibilities to properly set up and manage the applications on their systems for their own benefit and for the benefit of the network as a whole.

2.1.2. Security Considerations

Security is of critical importance in many networks, especially in likely applications of MANETs. In addition to the network functionality challenges imposed by the MANET environment characteristics mentioned above, there are further security challenges that need to be addressed. Networks must provide confidentiality, authentication, integrity, non-repudiation and availability [2].

Confidentiality is the ability to guarantee the privacy of the data transferred. In wired networks an eavesdropper has to be in the path between the source and destination of the communication entities in order to "sniff" on the transit traffic. Wireless links, however, can suffer from multiple points of attack as eavesdroppers can obtain the data transferred without being in the path of transit traffic, as long as they can intercept the desired radio signal. Additionally, some operations may require that connectivity information, such as routing, be confidential to avoid disclosing the location of the mobile entities.

Authentication allows two communicating entities to verify the identity of each other. If authentication is not used, an adversary in a MANET can masquerade as a node, gain access to unauthorized information, and interfere with the operation of the network while utilizing the network's limited resources.

Integrity ensures that a message transferred was not corrupted while in transit from the source node to the destination node. Compared to wired links, wireless links

can be plagued with a higher number of malicious nodes, which can intercept and alter the data.

Non-repudiation provides protection against denial of involvement in a communication by proving the origin of the data. The dynamic environment of MANETs makes the proof of involvement in a communication difficult. Non-repudiation is especially important in MANETs to isolate attackers or compromised nodes.

Availability is the protection against denial of service attacks. MANETs are more sensitive to Denial of Service (DoS) attacks. Jamming could affect the wireless link, decreasing the available bandwidth and increasing the data error rate. Attacks on the routing protocol service may disconnect the entire network. Attacks on the KMS may promote distrust among all nodes and generally disable the network.

Finally, since nodes in a MANET are mobile, they have poor physical protection and may be easily compromised. The dependence of the network on a centralized service provided by a mobile node is dangerous since the entire network may be deprived from that service, if that mobile node is compromised.

2.2. Overview of Cryptography

"Cryptography is the art of secret writing" [17]. Cryptography provides the ability to send information between communicating entities while at the same time preventing other people from reading it. A *plaintext* or *cleartext* is a message in its original form and a *ciphertext* is a message in a secret encoded form. *Encryption* is the process of producing a ciphertext from a cleartext and *decryption* is the reverse operation (see Figure 2.2). Cryptographic systems usually involve a secret value called a key and an algorithm. Possession of the algorithm alone does not enable an individual to decrypt the information. Furthermore, the cryptographic strength of a system does not depend on the cryptographic algorithm, but rather on the choice of keys.

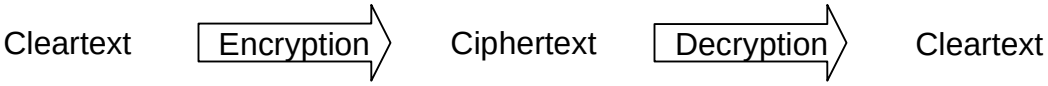


Figure 2.2. Cryptographic relationship.

2.2.1 Cryptographic Functions

There are three types of cryptographic functions: secret key functions, public key functions, and hash functions. The main difference is the number of keys the functions use. Public key functions require the use of two keys, secret key functions require one key and hash functions do not require keys [18].

2.2.1.1 Secret Key Cryptography

Secret key cryptography is also known as symmetric cryptography [18]. In secret key cryptography, one key is used to encrypt a message into a ciphertext. The recipient of the message then uses the same key to decrypt the message (see Figure 2.3).

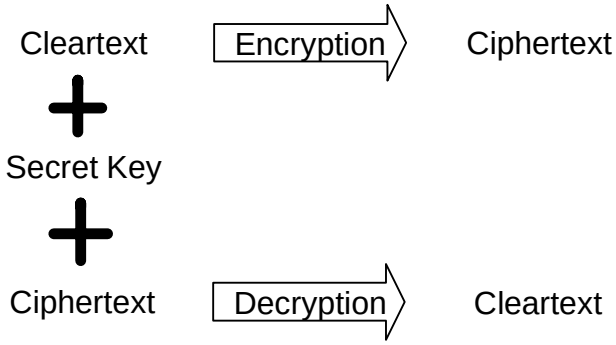


Figure 2.3. Secret key cryptography.

Secret key cryptography may be used to secure transit traffic by providing confidentiality. The information is encrypted into unintelligible data that can only be read by the intended recipient. Possession of the secret key may also be used to authenticate the sender of the information, assuming that nobody else has possession of the secret key.

Encryption may not always be needed or may not be feasible based on the processing capabilities of a node. In this case, secret keys may be utilized to provide message integrity, which requires less processing. An algorithm used in conjunction with a secret key is applied to the message to compute a cryptographic checksum, also known as Message Authentication Code (MAC) or Message Integrity Code (MIC). The MAC

and the message are sent to the communicating party, who can reproduce the checksum and check the integrity of the message.

Another use of secret keys is for storing of sensitive data on the network. In this case, the users have to be careful so that nobody obtains the key. Also, if they "lose" the key they will no longer be able to retrieve the information. Some algorithms that use secret key cryptography are the Digital Encryption Standard (DES) [19] and the Advanced Encryption Standard (AES) [20][21].

2.2.1.2. Public Key Cryptography

Public key cryptography is also known as asymmetric cryptography [18]. Individuals have two keys, a private key and a public key. The private key is not revealed to anyone, whereas the public key is made available to the public. For example, if there are two individuals, Alice and Bob, they each possess a private/public key pair. Alice has knowledge of Bob's public key and vice versa. If Alice wants to send a message to Bob, she will encrypt it with his public key and he will decrypt it with his private key (see Figure 2.4). Bob will reply to Alice by encrypting the information with Alice's public key, which Alice can decrypt with her private key. Unlike secret keys, anyone who has Bob's public key can send private information to him. There is no need for each pair of communication entities to share a key. Thus, public key cryptography is more easily configurable compared to secret key cryptography. However, public key cryptographic algorithms are significantly slower than secret key algorithms. A combination of secret and public key cryptography may be used during a communication.

Public key cryptography can provide the same functionality as secret key cryptography mentioned above, as well as digital signatures. With digital signatures, Alice generates a cryptographic checksum or "message signature" using her private key and sends it to Bob. Bob can then decrypt the message signature using Alice's public key (see Figure 2.5). This method is different from the integrity method using shared keys described above, because it guarantees integrity, as well as non-repudiation. Since Alice is the only one who possesses her private key then she is the one responsible for sending a particular message. If Alice and Bob shared a secret key, then it would be difficult for

Bob to prove that Alice signed the message even though he may be certain he did not sign it.

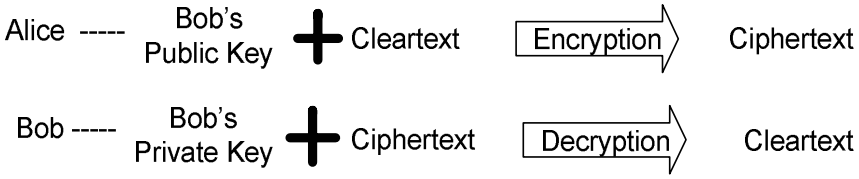


Figure 2.4. Public key cryptography

Figure 2.5. Digital signatures.

Two algorithms that use public key cryptography are Rivest-Shamir-Adleman (RSA) [22] and Diffie-Hellman (DH) [23]. DH predates RSA and it is the oldest public key cryptosystem still in use [18]. Unlike RSA, DH does neither encryption, nor digital signatures. DH enables two parties to establish a secret key between them by exchanging messages in public. A successful DH exchange does not imply that the parties trust/authenticate one another. The only accomplishment is the ability of two parties to agree on a shared key. The two parties may authenticate each other by using pre-shared keys or certificates from a trusted third party.

During a DH exchange, Alice picks two numbers *p* and *g* (public values), where *p* is a large prime number and *g* is a number smaller than *p*. She then makes them available to Bob. Alice and Bob generate random numbers *A* and *B*, respectively. Alice then calculates $M_A = g^A \bmod p$ and Bob calculates $M_B = g^B \bmod p$ and they exchange the results obtained. Alice then computes $M_B{}^A \bmod p$ and Bob computes $M_A{}^B \bmod p$. Thus, they both end up with the same number (or secret key). The proof of this exchange is shown in Equation 2.1.

$$M_B{}^A = (g^B)^A = g^{BA} = g^{AB} = (g^A)^B = M_A{}^B \bmod p. \qquad \text{EQN. 2.1}$$

In order to use the RSA algorithm, two prime numbers p and q are chosen. An odd value, e, is selected such that $1< e < pq$. Odd value, e, and $(p-1)(q-1)$ should be relatively prime which means that they should have no common prime factors. The private key, d is computed such that $(de-1)$ is evenly divisible by $(p-1)*(q-1)$ or $de=1(\text{mod } (p-1)(q-1))$. The public key is the pair (e, pq) and the private key pair is (d,pq). The private key is never revealed whereas the public key can be freely distributed. There are no known easy methods of calculating e, p, or q given only (e, pq). Equations 2.2 and 2.3 show the calculations carried out during the message exchange.

$$Alice \quad Ciphertext = (Message)^d \bmod pq \qquad \text{EQN. 2.2}$$

$$Bob \quad Message = (Ciphertext)^e \bmod pq \qquad \text{EQN. 2.3}$$

2.2.1.3. Hash Functions

Hash functions are also known as message digests [18]. A hash function takes a message of any length and uses it to compute a short, fixed-length number. A hash of a message is relatively easy to calculate compared to the previous algorithms. Hash functions operate on the conjecture that it is computationally infeasible to find a message that gives the same hash value.

Hash functions can be used for password hashing. Hashes of users' passwords can be stored in a computer instead of the actual password. When a user enters his password, the hash of the password is computed and is compared with the already stored value. This method hides the users' passwords from possible attackers.

In the case where Bob and Alice have a shared secret, hash functions can also provide message-integrity by using a *keyed hash***,** as shown in Figure 2.6. Alice can concatenate a message she wants to send to Bob with the secret key and then calculate its hash value. Alice can then send the hash value and her message to Bob. Bob can verify the integrity of the message by concatenating the message with the same secret key, re-computing the hash value and comparing the two hash values. If the hash values are the same, then the message has not been altered.

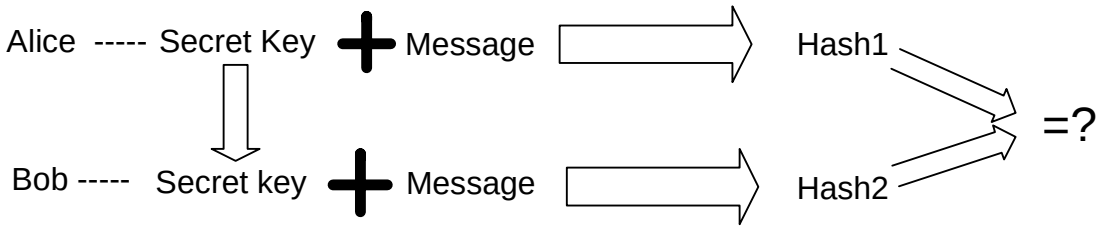


Figure 2.6. Keyed hash.

A hash function can also improve digital signature's efficiency. Bob can use his private key to sign the hash value of the message, instead of signing the entire message. Since the message digest is shorter than the message, it requires less processing to obtain the digital signature of the message [18].

2.2.2. Authentication Algorithms

Authentication algorithms may utilize any of the functions stated above in a variety of combinations to authenticate the parties involved. Authentication can be classified into three types: password-based authentication, address-based authentication and cryptographic authentication.

Password-based authentication is conducted by showing proof of knowledge of a secret piece of information. This type of authentication does not use any cryptographic algorithm [18]. A limitation with using passwords is that every time users want to connect to a server they have to input their passwords. If they want to connect to multiple servers they have to repeat this process multiple times. Even though this problem may be solved with a distributed system, other problems may arise such as password synchronization and storage. An additional problem with password-based authentication is that it is vulnerable to off-line and on-line password guessing attacks. Off-line attacks, also known as dictionary attacks, occur when an attacker acquires a message and uses a software program to guess a password that will produce the same message. On-line attacks occur when users try to guess the passwords while the system is functioning. On-line attacks can be mitigated by controlling the number and frequency of tries that a user can input his password.

Address-based authentication is more convenient and secure than password-based authentication and it is widely deployed in distributed systems. It provides proof of identity by using the network address of a computer to identify the source of the data received. Each computer can accommodate multiple user accounts and allow access to its resources. Even though this authentication is safe from eavesdropping, (that impacts password-based authentication), it is vulnerable to network address impersonation. An attacker can place herself in the path between two communicating entities and conduct a man-in-the-middle (MITM) attack. An example of the MITM attack is given in Section 3.3.2.1.

Cryptographic authentication is more secure than the previously described methods. However, it can be more processing intensive and may require more computer resources. Cryptographic authentication is provided by the cryptographic functions explained in Section 2.2.

2.3. Which Layer for Security?

Security protocols can be implemented at different network or OSI layers, depending on the application and user requirements. There are various advantages and disadvantages for implementing security at different layers. This section analyzes security and functionality at each OSI layer.

2.3.1. OSI Layers

Figure 2.7 shows the seven OSI layers. The physical layer transmits unstructured bits of data over the communication links. The data link layer collects the bits of data into frames and distributes them accordingly on the shared link. The network layer handles the routing of packets among nodes over multiple links. The transport layer implements the exchange of messages by establishing a reliable data stream and re-transmitting lost packets. The session layer ties together different transport streams, which are needed by a single application. Many systems do not use the functionality of this layer, or it is combined with the application layer, so it is not considered in this

security layer analysis. The presentation layer handles the format and encryption of the data exchanged between peers and it is typically combined with the application layer. The application layer constitutes the applications that use the network, such as file transfer applications. Security may be implemented at the data link, the network layer, the transport layer, and the presentation or application layer.

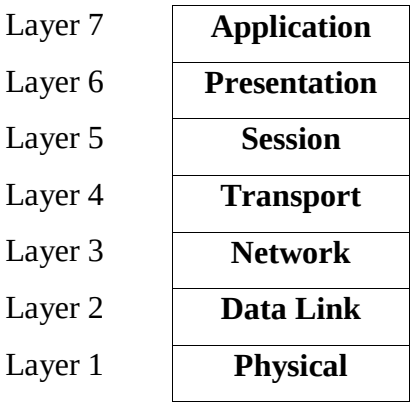


Figure 2.7. OSI layers.

2.3.2. Presentation or Application Layer Security

Presentation or application layer security removes the need to rely on the operating system to provide the required security services. Presentation or application security can provide different levels of security depending on the data exchanged. Furthermore, it allows more flexibility to satisfy the specific security needs of users such as verifying credentials of communicating entities and checking for non-repudiation.

The major disadvantage of presentation or application layer security is that each application needs to be enhanced to incorporate security. This process is complicated and prone to errors. Examples of presentation or application security services are Pretty Good Privacy (PGP) [24][25] and Privacy Enhanced Mail (PEM) [26].

2.3.3. Transport Layer Security

Like presentation or application layer security, transport layer security does not rely on security provided by lower layers. Higher level applications can interface with

the transport layer security protocol with minimal changes. Unlike presentation or application layer security, transport security does not provide user-specific services to support multiple users per node. Examples of transport layer security services are Transport Layer security (TLS) [27] and Secure Socket Layer (SSL) [28], and Secure Shell (SSH) [29][30][31][32][33].

2.3.4. Network Layer Security

Network layer security has many advantages. Network layer security can provide subnet-based security and create Virtual Private Networks (VPNs). VPNs are networks with subnet-to-subnet connectivity via public networks such as the Internet. Another advantage is that the overall key negotiation overhead is lower, since the transport protocols and applications working above the network layer can share the key management infrastructure. Also, network layer implementation of security avoids the explosion of complexity in the implementation of security at higher layers, since the security implementation at higher layers tends to be more complicated than at lower layers [34]. Finally, network layer security removes the need to implement security separately in every higher layer protocol or application because data is secured at the network layer.

Like transport layer security, when implementing security at the network layer, it is much more difficult to handle issues such as user-based security control on a multi-user system. Despite this weakness, it is assumed that, within a subnet, other mechanisms can be used to control users by securing appropriate higher protocol layers. IPsec is an example of network layer security [35].

2.3.5. Data Link Layer Security

Security at the data link layer is implemented in hardware and is, therefore, the fastest. It can be used when there are dedicated links between different entities. However, this solution is not scalable, as two entities have to be directly physically connected.

2.4. IPsec Overview

IPsec is a suite of protocols for protecting IP datagrams and was developed by the Internet Engineering Task Force (IETF). It provides connectionless data integrity authentication, data confidentiality, anti-replay protection, data origin authentication, and limited traffic flow confidentiality [35]. (Traffic flow confidentiality is the service that conceals the external characteristics of communication, such as source and destination addresses). To protect IP datagrams, the IPsec protocol suite provides security by defining header extensions to standard IP. The header extensions support two traffic security protocols, the Authentication Header (AH) protocol and the Encapsulation Security Payload (ESP) protocol. The security of AH and ESP is dependent on the cryptographic algorithms used, such as DES. These services require shared keys, which can either be negotiated manually or automatically. Manual keying scales poorly, so a dynamic way of negotiating security and generating shared keys, defined by the Internet Key Exchange protocol (IKE), is typically used.

2.4.1. IPsec Architecture

IPsec is a suite of protocols consisting of (1) protocols for securing packet flows (e.g., AH and ESP) and (2) key exchange protocols for setting up those secure packet flows (e.g., IKE) [34]. IPsec defines how these different components interact with each other to implement the required functionality. Figure 2.8 shows how the different components in IPsec are linked.

The ESP and AH documents define the protocols, the header formats, the packet processing rules and the various services that can be provided. However, they do not describe the transforms used to provide these capabilities. Different hash functions, such as MD5 [36][37], and encryption algorithms, such as DES [19], are used to authenticate and encrypt the data. The parameters that are used for encryption and authentication are defined in the IPsec Domain of Interpretation (DOI) for the Internet Security Association and Key Management Protocol (ISAKMP) [38]. The Security Policy (SP) is also an important part of a network. It determines which transforms two entities should use to

communicate with each other. Based on the security policy, the key management generates and manages a key by using IKE. All of the parameters used by the negotiating entities are defined in the DOI.

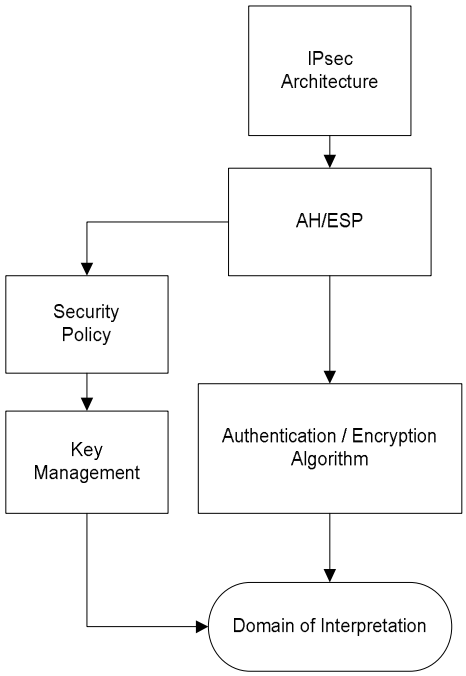


Figure 2.8. IPsec components and their relationship.

2.4.2. IPsec Modes

IPsec can be implemented in either the transport mode or the tunnel mode [35]. The transport mode in AH and ESP protects the IP payload, whereas the tunnel mode protects the full IP packet. The transport mode is used when the desired security is end-to-end. Figure 2.9 shows the format of packets for the transport mode.

IP header	Authentication Or Encryption Protocol	IP Payload (TCP+ Data)

Figure 2.9. Transport mode packet format [35].

The tunnel mode is used when the final destination of packet can be different from the security end point. Figure 2.10 shows the packet format for the tunnel mode.

Outer IP header	Authentication or Encryption Protocol	IP Packet (Inner IP+ TCP+ Data)

Figure 2.10. Tunnel mode packet format [35].

2.4.3. Security Associations (SA)

Security associations are contracts between two communicating entities that provide inherent security features. In IPsec, an SA defines the IPsec protocols, the transforms, the keys, and the key management used. An SA is unidirectional, meaning that different SAs are kept for inbound and outbound processing. All SAs are stored in the SA database (SADB), which works in conjunction with the Security Policy Database (SPD) (see Section 2.4.5).

For the purpose of this research, it is implied that two nodes trust each other if they establish an SA between them. Therefore, the term "SA" is used interchangeably with "trust establishment." If IPsec was deployed, an SA is referred to as an IPsec SA.

2.4.4. SA Management

SAs can be created and deleted either manually or automatically. With manual keying, two parties have to agree on the parameters of an SA offline. The negotiation of all the parameters required for IPsec is done manually and the parties involved have to periodically change those keys manually. Manual keying is usually used on a small scale or for testing. It is considered less secure and prone to errors compared to automatic keying.

Automatic keying is deployed through an Internet standard key management protocol such as IKE [39]. The IPsec kernel calls IKE when there is a need to establish a secure connection. The IKE negotiates the SA with the desired destination, and creates the SA depending on the policy used [34].

2.4.5. Security Policy (SP)

The SP determines the kind of services that can be used for a packet. It defines how IP packets are treated, which protocols to use, in which modes, and with which transforms. The SP can be different for inbound and outbound packets. Management of the SP is usually done by modifying the SPD, which is stored in the kernel. The IPsec implementation provides an interface to manipulate the SP.

2.4.6. IPsec Protocols

IPsec uses two traffic control protocols: ESP [40] and AH [41]. ESP can provide confidentiality, data origin authentication, integrity, anti-replay protection, and limited traffic flow confidentiality. Confidentiality is usually applied with authentication and integrity so that the traffic is not vulnerable to certain type of attacks, such as reading encrypted data and hijacking sessions, as described by Bellovin [42]. The anti-replay service is only effective if the receiver checks the sequence number.

AH is used to provide data integrity, data origin authentication, and optional limited anti-replay services to IP. Unlike ESP, AH cannot be used to provide encryption. In an IPsec implementation, if a network needs to use both authentication and encryption it can either use ESP to do both, or use AH for authentication and ESP for encryption (see Figure 2.11). The primary difference between AH and ESP authentication is the extent of coverage. In the tunnel mode, ESP does not authenticate any IP header fields of the outer IP header. AH can provide a better check of integrity, if required, since it extends its protection to predictable fields of the outer IP header.

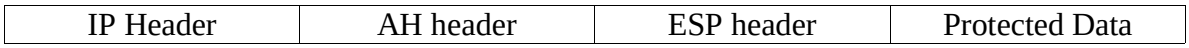


Figure 2.11. AH and ESP packet format [34][35][40].

There is an ongoing debate on whether AH is required. Some people argue that AH is unnecessary because there in no need to authenticate the AH header [43] and even if there is a need to protect the IP header, this protection can be provided in tunnel mode.

Furthermore, AH increases network overhead. On the contrary, previous research that I conducted showed that the overhead impact of AH compared to ESP authentication can be considered negligible, especially for large file transfers [44].

2.4.7. Authentication Algorithms

The authentication algorithm used for the Integrity Check Value (ICV) is usually specified by the SAs. Two parties that share a secret key may use MACs to validate messages sent between them. Suitable authentication algorithms include Keyed-Hashing Message Authentication Codes (HMAC), based on symmetric encryption algorithms [45]. HMAC provides a framework to incorporate any cryptographic hash function such as MD5 [36][37] and SHA1 [46]. The cryptographic strength of the HMAC mechanism depends on the security provided by the underlying hash function [34]. Even though MD5 has been found to be vulnerable to some attacks such as the collision search attack [47], the use of MD5 with HMAC is not compromised [48]. SHA1 is a cryptographically stronger function since it produces a larger message digest than MD5 (160-bit versus 128 bit) but it requires longer computational time.

2.4.8. Encryption Algorithms

Encryption provides data confidentiality and limited traffic flow confidentiality. Two algorithms, DES and 3DES were originally used for securing Internet traffic. The cryptographic community now considers DES [19] insecure because its 56-bit key can be discovered by a brute-force exhaustive attack in a relatively short period of time [49]. Triple-DES or 3DES has been used to replace DES [50]. 3DES uses a key size of 112 bits in applications, as opposed to 56 bits for DES. The disadvantage of 3DES is that the encryption and decryption time per block is three times that of DES. However, 3DES is more secure to brute force attacks than DES.

2.5. Alternative Security Systems

This section describes the functionality of alternative security systems as compared to IPsec.

2.5.1. Transport Layer Security Systems

Two transport layer security systems are the SSL and the TLS. SSL was first deployed in Netscape Navigator [18]. TLS was introduced by the Internet Engineering Task Force (IETF) [27]. TLS is similar to SSL but offers additional cryptographic capabilities. TLS does not interoperate with SSL and is not as widely deployed as SSL. SSL and TLS are used to authenticate two parties and establish a key that cryptographically protects the communication session. They protect streams of data instead of individual packets.

SSL and TLS inherit the transport layer characteristics, as described above, and operate "above" the Transport Control Protocol (TCP). A security problem with operating above TCP is that TCP cannot check whether the packets are valid until they reach the SSL and TLS service. TCP will drop the valid packets received that may arrive after bogus packets, since they will be considered duplicates [39]. IPsec can avoid this problem by dropping all invalid packets at the network layer. In spite of the fact that SSL and TLS are inferior to IPsec, SSL and TLS are more widely used than IPsec simply because IPsec requires Operating System configuration and more complex key management compared to SSL and TLS, which are hidden from the user.

2.5.2. Secure Shell (SSH)

SSH is a protocol that can provide secure remote login, file transfers and other security services [29][30][31][32][33]. SSH authenticates the users and encrypts all traffic (including passwords) providing confidentiality and perfect forward secrecy (PFS). PFS ensures that the compromise of a session key does not compromise any earlier sessions.

The functionality of this protocol is different from IPsec as it uses a client-server configuration. A user must have an account to be able to connect to the server, or any other node, which assumes a pre-existing level of trust from the other node. Although this protocol may seem relevant to IPsec, it is not, as they differ in functionality. SSH is implemented at the transport layer, and thus it inherits the properties of security at this layer as explained in Section 2.3.2.

2.5.3. Pretty Good Privacy (PGP)

PGP is a secure mail protocol. It can also provide integrity and confidentiality for any file that is sent via electronic mail [24][25]. PGP uses public key cryptography for authentication. The key infrastructure of PGP is explained in Section 2.8.2.3. PGP is specific to a particular functionality and inherits the security properties of the application layer.

2.6. Why IPsec?

Overall, security at the network layer has many advantages compared to security at the other layers. It avoids the complexity of implementation at the higher layers, provides subnet-based security for mobile subnets, and decreases key management overhead. Based on this analysis of security at each layer, IPsec is an attractive security mechanism to use in a network, since it is implemented at the network layer.

2.7. Authentication in IPsec

Automatic authentication in IPsec is done using IKE. IKE is a hybrid protocol for establishing SAs dynamically and populating the SADB [51]. The creation of IKE began with the proposal of two protocols, Photuris [52] and Simple Key management for Internet Protocol (SKIP) [53]. The IETF's IPsec group could not decide which protocol to incorporate into IPsec for some time. Eventually, two new protocols were introduced: the ISAKMP [54] and Oakley [55]. ISAKMP provides a syntax guide for peer

negotiation. Oakley provides a specific mechanism for exchanging keys through the definition of various key exchange modes. Most of the IKE key exchange process is based on Oakley. Oakley combines the advantages of both Photuris and SKIP. These two protocols slowly evolved into IKE. IKE integrated the foundation of ISAKMP, the modes of Oakley, and the keying techniques of SKEME. SKEME is a key exchange protocol that was proposed by Krawczyk [56].

IKE uses the two phases of ISAKMP. Phase 1 establishes an IKE SA and Phase 2 uses that SA to negotiate SAs for other protocols, such as IPsec. IKE has two Phase 1 exchanges, one Phase 2 exchange and two extra exchanges. The two Phase 1 exchanges are the *main mode* and the *aggressive mode*. Currently, there are eight variants of Phase 1 because there are four "types" of keys that can be used for each Phase 1 mode: secret keys, public signature keys, old-style public encryption key, and new style public encryption key. It is important to clarify that using a weak public or secret key does not affect the cryptographic strength of the keys derived during the IKE negotiation that will provide security for transit traffic. A weak key just compromises the authentication of the parties involved.

The main mode uses six messages as shown in Figure 2.12. These include three main steps:

1) a SA negotiation,
2) a Diffie-Hellman and nonces exchange, and
3) a peer authentication by showing proof of identify.

The curly brackets, "{}", in Figure 2.12 imply that the data is encrypted. A nonce is a random number that is generated in order to be associated with a particular message and is used to guarantee the freshness of the exchanged messages.

The aggressive mode requires half the number of exchanges of the main mode. In the first exchange, the user sends his DH public value, his nonce, his identity and his list of protection suites. The negotiation party replies with his DH public value, his nonce, his identity, his authentication payload, and his selected protection suite. Finally, the initiator then replies with his authentication data.

The aggressive mode limits the rich negotiation capabilities of IKE, such as not being able to offer different Diffie-Hellman groups in different protection suites.

However, it can be used in cases where the two parties know each other's policy and want to create an IKE SA more quickly.

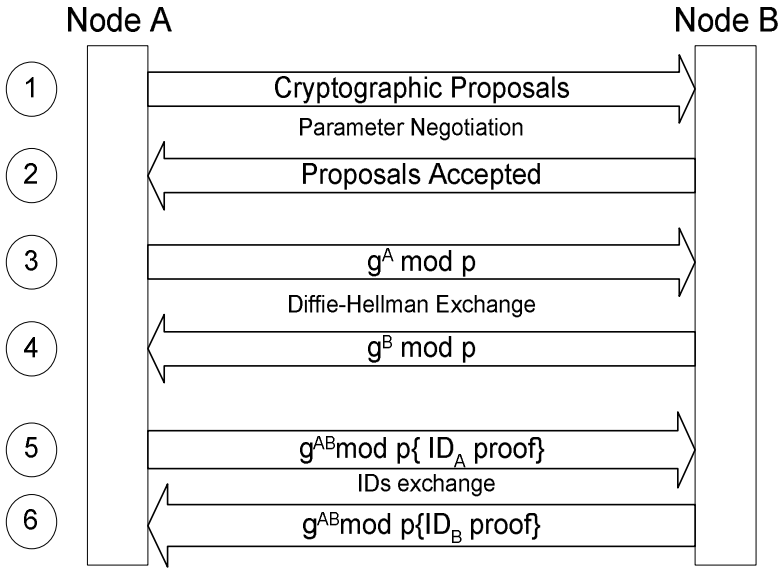


Figure 2.12. IKE Phase one main mode negotiation [18][51].

The Phase 2 exchange uses the *quick mode* to generate other SAs for other protocols such as IPsec. A quick mode negotiation uses three exchanges. The main reason for these transactions is to provide a liveness test similar to the three-way handshake of TCP. The optional and required attributes that IKE uses to negotiate in a Phase 2 exchange are defined in the DOI.

Thus far, the Phase 1 exchanges create the IKE SAs and the Phase 2 exchanges create the IPsec SAs. IKE also offers two extra exchanges. The first extra exchange allows communicating parties to send information related error messages and SAs' status. The second extra exchange defines a new group exchange and allows communication entities to negotiate private groups and group identifiers for their own use [34].

2.8. Key Distribution Services

This section describes ways of distributing keys within a network to provide authentication among entities. The operation and capabilities of the distribution centers are dependent on whether symmetric or asymmetric keys are used for authentication.

2.8.1. Symmetric Keys

A network with *N* nodes using symmetric keys (also known as "secret keys") and no central distribution center would require that each node have knowledge of *N*-1 keys, so that each can establish peer-to-peer trusted communication. If a new node joins the network, then *N* keys would need to be generated and distributed securely to all other nodes. This approach is not scalable.

A solution to this problem is to use a central distribution center (see Figure 2.13). If Alice wants to talk to Bob, she contacts the Key Distribution Center (KDC) and asks for a key to talk to Bob. The KDC authenticates Alice and then encrypts a nonce, $K_{AB,}$ that is used as a secret key with Alice's password, K_A. The KDC also encrypts K_{AB} with Bob's password, $K_{B,}$ and gives it to Alice to forward to Bob. This is usually called a *ticket*. Alice can then retrieve the secret key, and send a request to communicate to Bob together with her ticket.

There are several problems with this solution. First, the KDC is a single point of failure. If the KDC is unavailable, the network nodes cannot authenticate each other. Second, if the KDC is compromised, it can impersonate any node in the network or it can decrypt any conversation between two parties. Third, the KDC may lack the processing capability to serve all nodes, unless there are multiple KDCs. An example of this system is Kerberos.

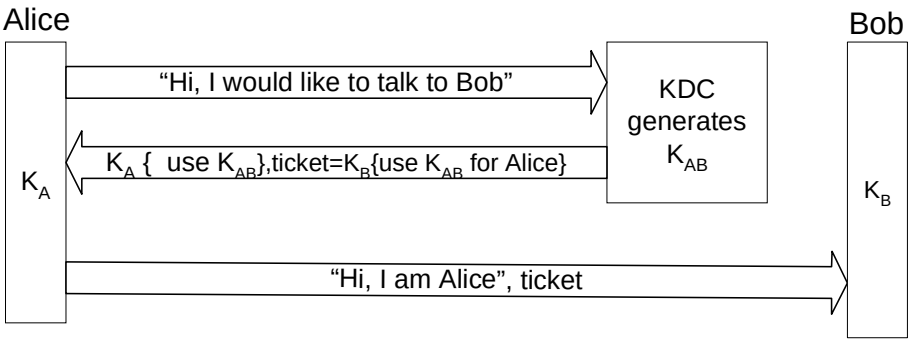


Figure 2.13. Secret key KDC.

2.8.2. Asymmetric Keys

Asymmetric keys make key distribution simpler. Each node only has to know its private key and obtain the public key of its communicating peer. Even though freely distributing the public keys does not give access to the private key, it introduces the MITM attack problem (see Section 3.3.2.1). An intruder, Chris, can claim another node's identity during the authentication process, tricking other nodes to establishing trust with her. This problem can be solved with a CA. A CA is a trusted entity or organization that signs certificates for all users, thus associating their identity with their public key. X.509 defines certificate formats [57][58][59]. A typical certificate includes the user's name and public key, a serial number, and an expiration date.

Nodes do not trust any certificates unless a trusted CA signs them. This approach assumes that nodes are pre-configured with the CA's public key so that they can verify the authenticity of the certificates. The certificates are stored in a central repository so that all the nodes on the network can access them. Usually the CA is not connected to the network, which secures it against any type of network attack and allows it to be a simpler system. Unlike the KDC with symmetric keys, if the CA fails it does not affect the performance of the network, since nodes can still obtain their keys or certificates from the central repository, assuming it is available. Furthermore, if the CA is on-line and becomes compromised, an attacker can impersonate any node in the network, but cannot decrypt communication between two nodes, as is the case with symmetric KDCs.

A disadvantage of this system is the inability to control a user's access to the network resources. Even though certificates expire after a period of time, there may be a need to deny access to a user before the certificate expires. This problem introduces the notion of certificate revocation. A certificate revocation list (CRL) enumerates all the unexpired certificates that have been revoked. This list is posted periodically in a repository to be accessed by all network users. To avoid scalability problems with very long CRLs, delta revocation may be used instead. With delta revocation only the periodic changes to the CRLs are posted. An alternative method to posting the CRLs in a repository for public access is to have a server that the nodes can query over the network

to check the status of a particular certificate. This server is known as an On-Line Revocation Server (OLRS).

Public Key Infrastructures (PKIs) utilize asymmetric keys. There are various PKI models that can be used in a network. Some common trust models are the monopoly, the oligarchy and the anarchy model [60]. There are other approaches such as the top-down and bottom-up models, but these are not explained here since they are beyond the scope of this research.

2.8.2.1. Monopoly

In the Monopoly model, a single trusted organization generates the certificates for all other organizations [60]. A problem with this model is that it is infeasible to change the key of a single CA when used by a large number of organizations. It is also difficult to find a universally trusted CA.

A variation of the monopoly is to use Registration Authorities (RAs) with the CA. The RAs collect the people's or nodes'credentials and then send that information to the CA to issue the certificate. This model facilitates the registration of more people or nodes. Another variation is to give RAs the authority to act as DCAs by having the central or root CA vouch for their trustworthiness. Trust chains can then be used to verify the authenticity of a user. For example, if Alice trusts Bob then Chris can trust a certificate issued from Bob as long as she can verify that Bob is trusted by Alice. These two model variations still suffer from the monopoly limitations and problems.

2.8.2.2. Oligarchy

In the Oligarchy model, multiple trusted CAs can verify the authenticity of a certificate [60]. Computers come pre-configured with a list of trusted CAs. Users can edit the list and choose to trust only a subset of the listed CAs. This model is widely used in web browsers. It allows for more flexibility through replication. If one CA is compromised it does not put at risk the whole community. However, this model is considered to be less secure than a monopoly model. It also raises questions on the trustworthiness and authority of product vendors to set up the list of CAs. In addition,

users have no way of examining the set of trusted CAs to determine whether an intruder has altered the CA list.

2.8.2.3. Anarchy

The Anarchy model is currently used by PGP [60]. In this model, there is no centralized, trusted CA. All nodes in a network can sign certificates by collecting the credentials of other nodes using some out-of-band method. They can then share their certificates and build chains of trust among them. A disadvantage in this model is that it is not scalable. If all the nodes have the authority to sign certificates, then the total number of certificates will increase rapidly, especially in a large network. This model is also the least secure compared to the oligarchy and monopoly model, since users cannot verify the trustworthiness of all the nodes in a trust chain.

2.9. Previous Work

The previous work presented in this section discusses key management in MANETs. PKI schemes, and replicated services that were proposed and were not applicable to MANETs are not analyzed. Also, research conducted that relates to group key management is not analyzed, as the focus of this research was to provide trust between two peers. In addition, authentication protocols are not described as they are indirectly related. An authentication protocol is a mechanism for exchanging keys and authenticating peers provided a key infrastructure is in place.

Zhou and Haas [2] proposed a partially distributed key management service for MANET. Their proposal involved the use of a threshold cryptographic key. A system-wide public/private key pair, K/k, was created for the entire KMS. The system's public key, K, was known by all nodes in the network but the private key, k, was divided among *n* chosen servers into *n* shares (s_1, s_2 … s_n), assigning one share for each server. In addition, each certificate server was pre-configured with the public key of the other servers. When signing a certificate, a server used its partial private key to generate a partial signature. The partial signature was then sent to an arbitrary selected server,

which computed the certificate from the partial signatures and sent it to the node that requested the certificate.

Distribution of trust was achieved using threshold cryptography with an (n, t+1) configuration. The service could not be compromised, as long as the number of compromised nodes, t, were less than the servers, ($t < n$). This configuration implied that t+1 compromised nodes were needed to sign a certificate. The service periodically computed new shares of a private key from the old ones (share refreshing) to tolerate compromised servers and to adapt to changes in the network.

This threshold cryptography scheme did not provide sufficient functionality. It assumed that there was a trusted authority that authorized the arbitrarily chosen nodes to behave as trusted CAs. Each server had to *know* and *trust* the public keys of the rest of the servers before the network deployment, which might not be feasible in a MANET environment. This system was implemented in a wired network [61], but was not tested in a MANET environment.

Kong, et al. [3] proposed a similar threshold cryptography system that allowed new nodes joining the network to obtain a share of the KMS' private key. The advantage of this scheme as compared to the scheme presented by Zhou and Haas [2], was that it increased availability since any t+1 nodes in the local neighborhood of the requesting node could issue or renew certificates. In addition, the load of the key management service was spread over a higher number of servers.

Kong, et al. [3] implemented both implicit and explicit revocation. All the nodes in the network maintained a CRL. Similar to [2], it was assumed that the initial group of t+1 nodes was empowered/initialized by a trusted authority. It was also assumed that all new nodes joining the network would be authenticated through the use of reliable out-of-band physical methods, such as human perceptions and biometrics, before obtaining a secret share of the system. This assumption lowered the flexibility of the system, as new nodes had to be properly authenticated before they could contribute to the key management process. It seemed that the results presented did not take into account the delay required to authenticate each new node through out-of-band methods to be considered a fully functional entity of the KMS. This omission gave overly optimistic results for the nodes' initialization phase.

Based on work by Capkun, Buttyan, and Hubaux [6], this system seemed to be vulnerable to the Sybil attack [62] because of the network-wide distribution of the private key. (The Sybil attack is an attack during which mobile adversaries forge the identities of enough nodes to be able to reconstruct the private key of the distribution scheme.) It was also not clear whether the group size of t+1 nodes that were required to provide key management services to a single node changed with the network size. A large group size could provide more robustness but decrease the service availability. On the other hand, a small group size could increase availability but decrease robustness.

Yi and Kravets [3] extended the work done by Zhou and Haas [2] for application in a MANET. They defined tunable parameters that could be used in the operation of their KMS. For revocation, they implemented a simple CRL approach. Nodes built full revocation certificates using partial revocation certificates transmitted via a network-wide flood. The authors stated that a more efficient support for revocation would be investigated.

Bechler, et al. [4] continued this line of investigation and applied the KMS proposed by Zhou and Haas [2] on a cluster-oriented network. Cluster heads were assigned the role of signing certificates for other nodes. The authors introduced the idea that nodes must present a certain number of warranty certificates verifying their credentials before they could obtain a certificate from a cluster head and become full members of the network. Those warranty certificates were obtained from existing full members of the network.

Current research in threshold cryptography schemes [2][3][4][5] does not fully address two important issues: information propagation and verification across multiple DCAs, and revocation authorization and response. The first issue is that new nodes joining the network have to present their information (e.g., public key) to multiple CAs to be able to obtain a sufficient number of partial certificates. Their information needs to be communicated to the CAs via out-of-band methods so that the authenticity of the information is verified by other CAs. In a highly dynamic environment, certificate reissuance can be hindered if a node does not communicate via out-of-band methods with a sufficient number of CAs. This problem can be eliminated to some extent by having the new node obtain a certificate from a trusted off-line RCA. However, in an open system

where nodes leave and join the network, enforcing that all new nodes obtain an RCA certificate is unworkable. Bechler, et al. [4] dealt with this issue through the usage of warranty certificates but assumed that the warrant issuers were trustworthy. In addition, the requirement of collecting a number of warranty certificates with out-of-band methods was not very flexible.

The second issue deals with revocation authorization and response. Revocation is a critical obstacle to the operation of a KMS in a MANET. It is used to revoke the certificate of a compromised node or CA. The limitation with implementing revocation with threshold cryptography is that a certain number of CAs has to be *made aware* and be *convinced* of the malicious activity of a particular node before they can issue partial RNs. Current research has not addressed how this revocation process is carried out. Furthermore, some solutions did not discuss revocation [2][4]. Other approaches [3][5] were still susceptible to the high mobility and low network capacity of the MANET environment that lowered the responsiveness of the revocation process.

The authors in [6] through [10] proposed a PKI anarchy model similar to PGP. The KMS allowed all the nodes in a MANET to issue certificates to each other. The nodes kept databases of expired and updated certificates, and cooperated with each other to build chains of trust among them. The issued certificates were stored among all nodes. The issuance and revocation of certificates were performed consciously by the users, whereas all other operations such as authentication were automatic and required no user intervention. The system relied on both explicit and implicit revocation executed by all nodes to provide sufficient security in the certificate chains. In their later work, Capkun, Hubaux, and Buttyan [63] noted that this system required a costly initialization phase in terms of both overhead and time since each node had to build its local certificate repository to be able to use the services of the system. Moreover, this scheme utilized transitivity of trust, which had the fatal problem of the untrustworthiness of certificate chains that is inherent in PKI anarchy models. For example, if there is a chain of trust from A to B to C to D to E, A may trust B to sign a certificate, but since it does not know C, it cannot assume that C is trustworthy enough to sign a certificate. A CA may verify the identity of the other nodes but it does not infer that the verified node information should be trusted. Trust information applies only to the first link in the chain (e.g., A to

B) [60]. It was also suggested that the users exchanged their keys whenever they met via a secure channel, such as infrared, to set up a higher level of trust among them. As mentioned above, this was not always feasible in a MANET.

To evade the transitivity of trust presented in their prior work [6] through [10], the authors presented an idea that each node could build SAs with the help of its existing friends [63]. They demonstrated that mobility increased the number of SAs established. SA establishment was dependent on the existence of a group of friends for every node in the network. In addition, the authors did not investigate revocation for this scheme.

O' Shea and Roe [12] worked on the address ownership problem in IPv6. They proposed that the public key be used to derive part of the IP address of a node in a cryptographically verifiable way. The system provided non-repudiation since the origin of the data could be proved using the public key of the communicating node. This research was intended to provide experimentation with Mobile IPv6 before transition to the IPsec infrastructure.

Montenegro and Castelluccia [11] did very similar work to O' Shea and Roe [12]. However, their communication protocol was structured differently. Their solution was also applied to Mobile IPv6 (MIPv6) nodes to set up trust between mobile nodes and corresponding nodes. These nodes could then establish an IPsec SA to secure MIPv6 binding updates.

Approaches described in [11] and [12] indicated that certificates were not necessarily required but they had some limitations. Unlike CA certificates, the IP address of a node could not be revoked in case the private address of a node was compromised. Furthermore, showing proof of address ownership did not provide authentication or hierarchy of trust. Hierarchy of trust is the ability to assign levels of trust based on the authenticated identity of the user. Another limitation was that these solutions could not be applied to IPv4 addresses. O' Shea and Roe [12] pointed out that a global PKI, such as the one proposed in this research, could be used to provide authentication.

Summarizing, previous work was limited by a number of weaknesses. Threshold cryptography KMSs [2][3][4][5] assumed a certain level of pre-configuration to allow nodes to act as CAs. This level of pre-configuration provided less flexibility in terms of initializing the KMS and dynamically setting up CAs during the network lifetime. In

addition, all of the schemes, except the one of Kong, et al. [5], were unsuitable for MANETs with relatively high mobility and low connectivity, since a single node was dependent on a group of servers to obtain partial certificates. The low availability of these schemes is shown in our analysis in Sections 5.4.3.3 and 5.2.3.3. Moreover, these schemes relied on out-of-band methods to authenticate new nodes [3][4] and did not address issues such as information propagation and verification across multiple CAs, and revocation authorization and response. A scheme that was based on the anarchy model [6] through [10] increased availability, but suffered from high initialization cost and relied on the establishment of chains of trust among all nodes in the network. Other schemes were inflexible and offered limited functionality [11][12] with regards to certificate revocation, user authentication, and hierarchy of trust. None of the authors, except Montenegro and Castelluccia [11], investigated the interoperability of their solution with the existing IPsec architecture.

2.10. Summary

This chapter gave a background overview and presented relevant literature for the proposed research problem. The proposed KMS would operate in a MANET environment. A MANET is a collection of mobile routers that move dynamically in unpredictable directions. The links connecting the nodes are wireless, which means that they are not as dependable as wired links and they have more capacity constraints. This environment is characterized by more security threats compared to wired networks, because the wireless links are more vulnerable and the nodes have less physical protection. To counteract these threats, a network needs to provide authentication, confidentiality, non-repudiation and increase the network services availability. Cryptography can provide these services by utilizing different cryptographic functions in a variety of combinations. Even though authentication can also be provided by password-based or address-based authentication protocol, cryptographic authentication is the most secure.

Currently various security mechanisms exist that can provide security in the network. IPsec was selected for this research because it is implemented at the network

layer and provides more advantages compared to its competitors. IPsec employs two traffic protocols, AH and ESP, and various authentication and encryption algorithms. The set up of trust among peers is achieved through IKE. Two nodes authenticate one another by showing a valid certificate or a (pre-shared) secret key that is negotiated with some out-of-band method. The keys can be automatically distributed to nodes by using a KMS. KMSs based on asymmetric keys are less complex compared to symmetric keys.

Various asymmetric key or PKI models exist depending on the functionality and security required in the network. Functionality increases and security decreases as the PKI model implementations transition from a centralized one to a hierarchical one and finally to an anarchical one. Previous work related to this area of research has revealed a series of limitations: (1) low service availability, (2) pre-configuration for CAs, (3) transitivity of trust, and (4) inflexibility in accommodating new nodes.

Chapter 3 presents the current state of technology of key management and IPsec in a MANET. It describes specific issues that are used to further identify and define the research problem of distributed key management, and investigates the feasibility of using implementation as an evaluation technique for the proposed KMS.

Chapter 3

Key Management and IPsec Technologies

This chapter presents the preliminary work conducted for the proposed research problem. Section 3.1 describes the objective for conducting the preliminary work. Sections 3.2 and 3.3 present two testbeds that were deployed to investigate the current state of technology. Section 3.4 discusses the findings of this research as related to this proposal. Based on the findings, the research problems related to key management are identified in Section 3.5.

3.1. Objective

The main objective for investigating key management and IPsec in a MANET was to gain an understanding of the current state of technology and extract generic problems from the specific issues raised. The issues related to key management functionality and interoperation of key management with IPsec. This objective was achieved by deploying IPsec in two testbeds. Testbed 1 investigated interoperability of IPsec over multiple platforms and Testbed 2 investigated interoperability of IPsec with different emerging technologies and standards of interest such as dynamic routing, Quality of Service (QoS) and real-time applications, and network management. Another objective of this research was to assess the maturity of the software used in order to determine the feasibility of using implementation as an evaluation technique. The impact on key management for all the solutions implemented in the testbeds was evaluated with regards to scalability, security, and functionality. The presented results place more emphasis on key management and less emphasis on describing the detailed solutions to the various problems solved.

This research evolved from work on the NAVCIITI project, which investigated the network infrastructure for a Virtual Operations Network (VON), which is a rapidly-deployable internetwork of naval vessels at sea. The computer network security area of this task focused on the problem of enabling interoperability between heterogeneous networks that belonged to and were managed by different allies and coalition partners. The research addressed the security aspects of establishing a VON. A VON required the creation of tunnels to secure data between different coalition partners. Since IPsec is at the network layer, it provided subnet-based security, which meant that all the data exchanged between two distinct subnets (ships) were secured.

3.2. IPsec Interoperability on Multiple Platforms

Testbed 1 was deployed to investigate interoperation of IPsec and key management on different platforms. More specifically the scenarios considered were:

1) interoperation of a Cisco router with Windows 2000,
2) interoperation of Linux with a Cisco router, and
3) interoperation of Linux with Windows 2000.

3.2.1. Implementation Overview

Figure 3.1 depicts Testbed 1. Each gateway had a subnet attached to it. A 10-Mbps Ethernet local area network was used to connect the various network components. A third computer, the "Sniffer," collected the data that were communicated between the different systems. A 128-bit encryption image was installed on the Cisco router. FreeS/WAN IPsec, a freely available commercial-off-the-shelf implementation of IPsec was installed in the Linux gateway [13]. This implementation was the only available full implementation of IPsec in Linux at the time of deployment of this testbed. The latest version of Linux has another built-in version of IPsec, which is less mature compared to FreeS/WAN IPsec.

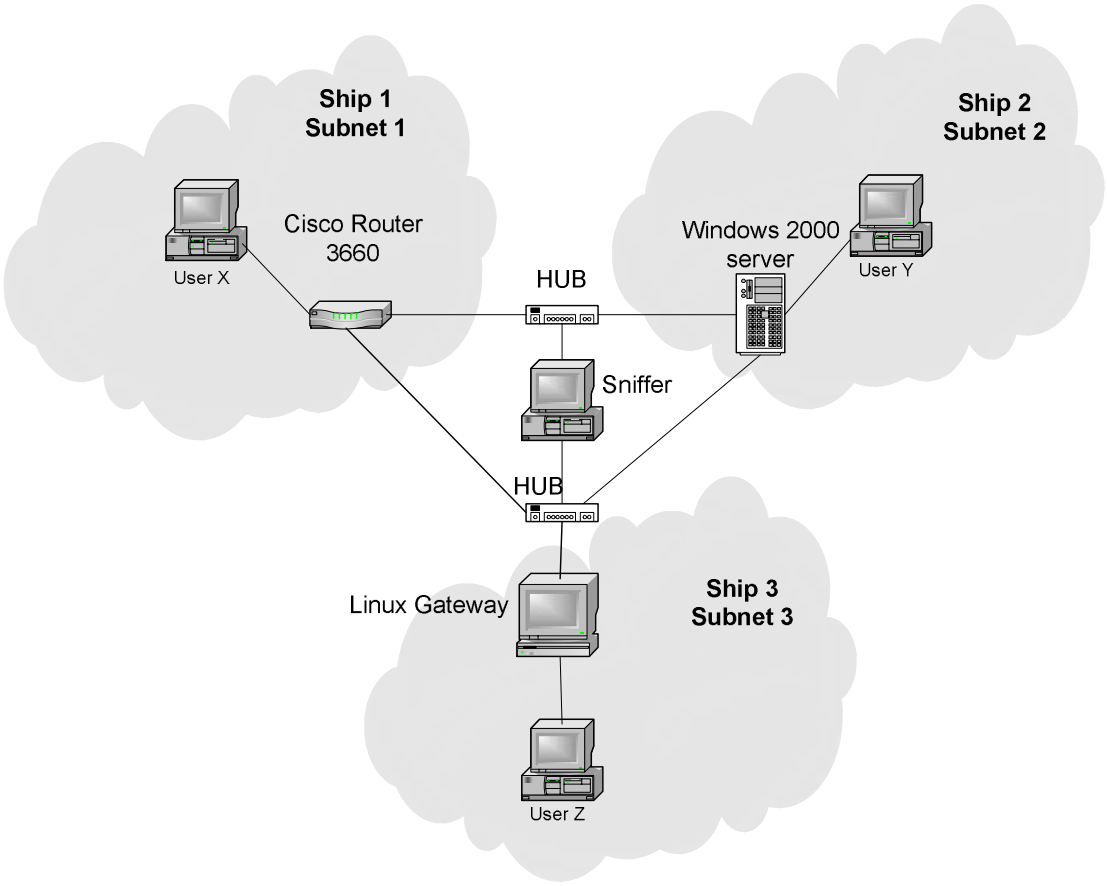


Figure 3.1. Testbed 1.

In previous work I had conducted [44], keys were installed on the nodes using manual keying. IKE was not used and there was no actual exchange of keys. Manual keying is useful for testing and can be employed in some implementations that offer no auto-keyed mode. However, manual keying is prone to errors and it scales poorly. An attacker that manages to subvert some network entities can use the manual key to read all data, both new and old, encrypted with those keys. To avoid the limitations of manual keying, automatic keying was used instead. Automatic keying utilizes the IKE protocol. An IKE negotiation provides Perfect Forward Secrecy (PFS), which means that an attacker cannot read previous messages. The attacker who obtains the secret key may conduct a successful attack, such as the MITM, which gives him the encryption key to be able to read the messages. However, with automatically re-keyed connections, the

encryption keys change often. An opponent who gets one key does not get a large amount of data, unless he can still deploy the same attack. It should be emphasized that the secret keys merely provide authentication. They are not the actual keys used in the encryption of data.

Automatic keying in FreeS/WAN could use either secret keys or RSA public/private keys. Secret keys were installed on each gateway, as they were simpler to test and set up. Authentication using secret keys in FreeS/WAN IPsec was carried out by verifying the pre-shared secret key and its corresponding IP address. This method did not authenticate anything other than the IP address of a node. Certificates could be introduced during the IKE negotiation to provide more information about the user of a peer node during the authentication phase of IPsec. However, software patches had to be applied to FreeS/WAN IPsec to enable the introduction of certificates. These software patches modified the IKE negotiation parameters and inhibited IPsec interoperation with Cisco and Windows 2000.

Another limitation was that the information received concerning the other parties during the IKE negotiation could not be communicated to the higher layer applications. A solution to this problem was to implement an Applications Programming Interface (API) that could be used to provide this information. The API would give applications the ability to observe and influence how IPsec protects the application's traffic. This API would also remove the need to deploy two levels of authentication, one by IPsec IKE and one by the application. It could also be used for an external Simple Authentication and Security Layer (SASL) mechanism implemented on top of IPsec [64]. Even though the IETF's IPsec Policy group was in the process of defining the requirements for such an API [65], progress in achieving this task has been slow. The main difficulty was the complexity of modeling IPsec security policies based on the operations of the higher layer applications.

3.2.2. Results

IPsec implementations on the Windows 2000, Linux, and Cisco router platforms could interoperate using automatic keying, which was more secure and scalable than

manual keying. This testbed contributed to an in-depth knowledge on the intricacies of configuring IPsec on all platforms. Cisco, Linux and Windows 2000 made different default assumptions about IPsec. Various parameters had to be adjusted for the tunnels to work properly. Some examples were the DH group, connection timeout, and the use of PFS.

In addition, the FreeS/WAN Linux implementation had some peculiarities and limitations. If the key negotiation for a tunnel failed, an SA could not be renegotiated without restarting the IPsec service. Restarting the service meant that all existing IPsec tunnels had to be shut down. Furthermore, the IPsec implementation was inflexible with regards to dynamically modifying the security policies of the system. If a tunnel was brought down, then the two parties could not dynamically switch to cleartext communication since the security policy assumed that communication was no longer trusted between those parties. In addition, the cryptographic parameters between two parties could not be adjusted dynamically. Moreover, if the secret keys of new connections were installed, they were not applied to the IPsec mechanism until the service was restarted, which meant shutting down existing tunnels, as previously mentioned.

The use of automatic keying avoided the manual installation of encryption keys, but did not avoid the manual set up of pre-shared authentication keys on every node. Kerberos, a KDC, could make the management process more scalable, but was not likely to interoperate due to modifications by Microsoft in Windows 2000. An alternative solution was the Remote Authentication Dial-In User Service (RADIUS), which could have been used to combine user-machine authentication and allow for trust level hierarchy. However, this system was not suitable for a decentralized MANET environment; it was more applicable to a star network topology, where users log in and set up an SA with a centralized gateway. A better solution was provided by the latest implementation feature in FreeS/WAN called opportunistic encryption (OE) [13]. This feature provided more flexibility in key distribution, IPsec configuration and user authentication and it was more suited for dynamic topologies, such as MANETs. This feature is investigated in the next phase of this research, which follows.

3.3. IPsec Interoperability with Alternative Technologies

The goals for deploying Testbed 2 were twofold:

1) investigate key management based on the findings of Testbed 1, and
2) integrate/interoperate IPsec with routing, QoS and real-time systems, and network management.

3.3.1. Implementation Overview

Testbed 2 was comprised of a number of gateways as shown in Figure 3.2. It was assumed that all gateway routers would have a subnet attached to them and form a backbone as a MANET. This testbed was not a standard form of MANETs since the gateway routers could direct traffic for a group of subnet nodes that did not contribute to the overall operation of the network.

Some gateways were connected by wireless links and others were connected by wired links via a dynamic switch [66]. The dynamic switch emulated the topology changes of a MANET and allowed specification of channel properties, such as bandwidth and packet drop rate. The wireless links operated at 10 Mbps. The testbed incorporated nodes with Linux, Cisco and Windows 2000 operating systems. FreeS/WAN IPsec was installed on all the gateways (G1-G10 in Figure 3.2). The IPsec mechanism on each gateway obtained the required authentication keys from either of the two available Domain Name Servers (DNSs). Each DNS was set up in a different subnet and was protected by the IPsec gateways. The DNSs were implemented using the BIND software for the Linux operating system [67]. The DNSs were not deployed in Windows 2000 environment because the Windows 2000 DNS application did not support public key storage.

In constructing the testbed, different technologies, such as the dynamic switch, dynamic routing, QoS and real-time systems, and network management, were contributed by other researchers at Virginia Tech [68][69][70][71][72][73]. The dynamic routing protocol utilized by the nodes was similar to the Open Short Path First (OSPF) routing

protocol and provided routing information to the gateways. The QoS mechanism provided classification for network traffic in order to meet the requirements of real-time systems. The network management service was provided by TopoView, a tool which showed the network backbone topology as it dynamically changed. More details on the various aspect of the testbed are reported in [74].

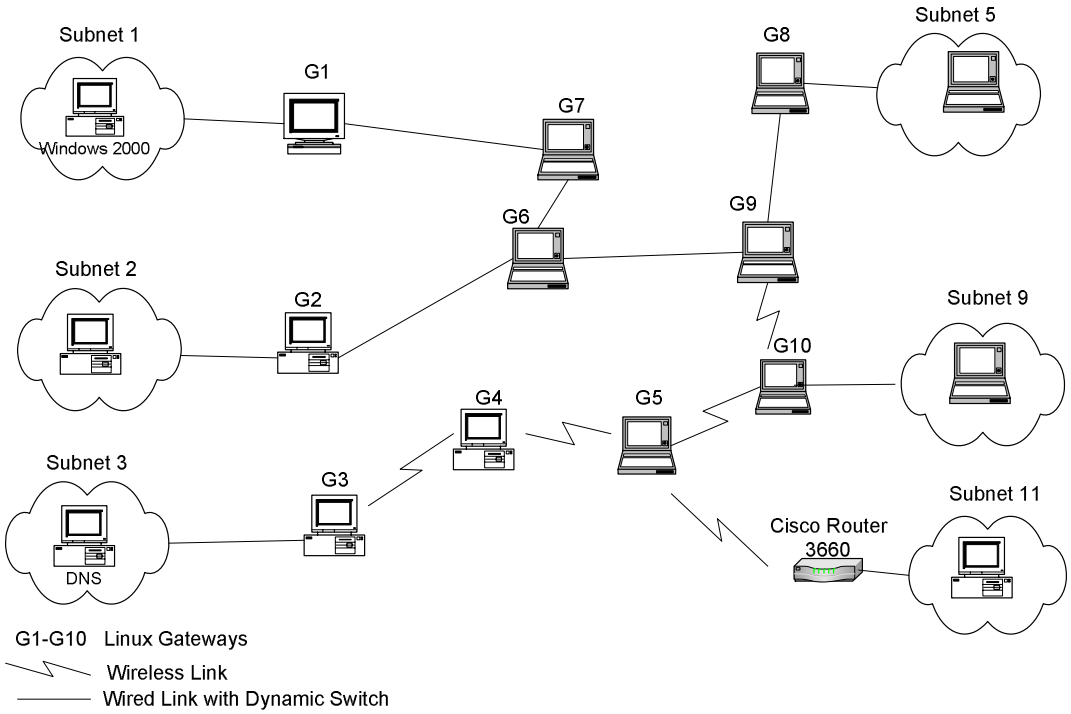


Figure 3.2. Testbed 2.

3.3.2. Results

The investigation related to this testbed was carried out in a series of four steps:

1) investigate Key Management,
2) investigate interoperation of IPsec and routing,
3) investigate interoperation of IPsec, QoS and real-time systems, and
4) investigate interoperation of IPsec and network management.

The solution from each step was analyzed so that it did not impact the solution of the previous step(s). If the solution of one step conflicted with the operation of the

previous step(s), then the previous solution(s) were revised. The objective was met when IPsec operated with all the network components. The following subsections describe and analyze the results of each step. More emphasis is given on key management.

3.3.2.1. Key Management

In Testbed 1, the focus with regards to key management was to avoid the usage of manual keying and use automatic keying instead. Scalability issues were not investigated because the number of testbed nodes was small. However, Testbed 2 was a much larger network and the impact of using either symmetric (shared) or asymmetric (private/public) keys had to be investigated. If every gateway had to establish SAs with the rest of the network gateways, forming a mesh topology of SAs, then N gateways would require $N(N\text{-}1)/2$ shared keys. In the case of asymmetric keys, N gateways would require only N public keys. Therefore, key management with asymmetric keys was a better alternative. However, the key administration set up of asymmetric keys (without the aid of a KMS) was slightly more complicated than with secret keys. Installing secret keys required setting up N-1 [IP address + secret key] pairs on every gateway whereas installing public keys required setting up $N(N\text{-}1)/2$ short scripts consisting of all the possible combinations of SAs that could be requested by a gateway. Each short script contained information about the IDs of the two communicating parties and their public keys.

Regardless of the type of keys used, a new configuration file had to be generated every time a new node joined the network. Once the configuration file was modified, the IPsec service had to be restarted in order for the keys to be installed. This requirement proved to be unworkable for a MANET environment. These key management problems were mitigated with the use of OE, which provided more flexibility in key distribution, IPsec configuration, and user authentication.

OE allowed any two systems to encrypt their communication without requiring a pre-shared key negotiated through out-of-band methods. Opportunistic encryption utilized the DNS to store the public keys of all the nodes. This feature removed the need to set up the public keys in the configuration file of IPsec. In addition, nodes did not

have prior knowledge of the IP address of their peers. They could dynamically obtain each other's public key during the IKE negotiation and set up SAs between other nodes.

A disadvantage of OE was that it was vulnerable to the MITM attack. For example, if Alice wanted to establish an SA with Bob utilizing OE, she would need the DNS record and IP address of Bob. If Chris had a server and was located between the path of Alice and Bob, he could intercept Alice's request to update her public "KEY" record and replace it with his own RSA key. Whenever a request is sent for Alice's record, Chris could reply and send his DNS records. In this way, Bob would receive Chris' "KEY" records when he requests Alice's "KEY" record. If Bob wanted to communicate with Alice, Chris could pretend to be Alice, spoofing her IP address, and negotiating an SA with Bob. Likewise, Chris could initiate a connection with Alice, while spoofing Bob's IP address. If Alice wanted Bob's key information from the DNS, Chris could send his own "KEY" information. As a result, Chris would establish an SA with Bob and Alice and both Bob and Alice would think that they have an SA with one another. Therefore, Chris, as the MITM, could potentially manipulate all the information that he received from both Alice and Bob.

The use of a secure DNS could avoid the MITM attack since Chris would not be able to alter any information unless he had the DNS private key. DNS security extensions (DNSSEC) that use public key cryptography and provide security for querying responses, dynamic updates, and zone transfers are currently available. However, DNSSEC features were not utilized in the testbed as they did not interoperate with the IPsec implementation. DNSSEC is available in recent versions of BIND [67]. IETF's DNSEXT working group was working on DNSSEC and may make more changes to it [75].

3.3.2.1.1. SA Negotiation

SA negotiation required user intervention since a series of checks had to be completed before starting the key negotiation. More specifically, the user had to:

- check the IPsec service status,
- check connectivity to the party of interest, and

- check connectivity and operation of the DNS to ensure public keys can be received.

A program was written to address this problem by executing these checks automatically and negotiating an SA. However, this program only provided a partial solution to the problem because it was not integrated with IPsec. As a result, it was not aware of failed negotiations to take the appropriate corrective actions. Moreover, IPsec did not provide any support for obtaining this information. The system administrator had to manually check for the success of the IPsec SA establishment. The IETF's IPsec working group proposed drafts [76][77] that would allow network administrators to perform operational level monitoring of the IPsec portion of their network but they have not been standardized.

3.3.2.1.2. Authentication with CA certificates

Authentication of RSA public keys embedded in certificates was not implemented on the testbed. Even though FreeS/WAN users' patches were available that could allow the use of X.509 certificates, the certificates could not be used in conjunction with OE because the certificates could not be stored on the DNS. If certificates were deployed, they had to be manually stored on each node instead of the DNS, thus increasing management complexity. This approach could increase availability of authentication information and facilitate the establishment of SAs. However, it suffered from one major drawback; the validity of the certificates could not be verified during the lifetime of the certificates because there was no mechanism to manage and revoke invalid certificates.

3.3.2.2. Interoperability of IPsec and Routing

FreeS/WAN IPsec required the most adjustments, compared to the other network functions, in order to interoperate with routing. Transport mode in FreeS/WAN IPsec did not operate when the data transferred between two gateways in the same subnet had to be redirected through another node. Therefore, tunnel mode was the only mode that could be deployed.

Furthermore, the operation of the dynamic routing protocol affected the operation of IPsec. FreeS/WAN IPsec creates a virtual interface for any IPsec negotiated connection so that packets can be routed through that interface. This functionality caused interoperability problems, because the dynamic routing protocol would replace IPsec's virtual interface route entries with its own entries. As a result, IPsec would cease to function. The solution to this problem was to use different subnet masks to avoid conflict with the routing protocol and at the same time to comply with the IPsec policy. To some extent, configuring IPsec resembled configuring packet filtering, as carried out in firewalls, and then adding in the additional complexities related to key management. (An alternative to using subnet masks would have been to use firewalls to direct traffic to different interfaces, which required additional software set up per gateway.)

An important issue with IPsec interoperability was the responsiveness in adjusting the security policies associated with an SA in order to route information over different IPsec tunnels. The IPsec virtual interface did not allow traffic to go through it if the traffic did not comply with the security policy associated with that interface. The traffic could be allowed through that virtual interface only after renegotiating SAs with the communicating entity via IKE. This IPsec functionality created problems in a MANET when utilizing dynamic routing at the VPN layer, as connectivity changed much faster compared to the time it took to negotiate an SA. For example, in Figure 3.3 nodes have point-to-point IPsec tunnels (i.e., SAs) between them. If A would like to route traffic to E then it can send data either through B or D. However, the SPD controls traffic by routing traffic only through the nodes specified during the SA negotiation.

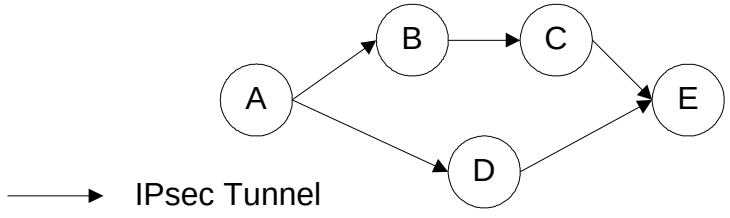


Figure 3.3. Dynamic routing over IPsec tunnels.

Wang and Touch [78] and Touch [79] proposed a method for the Xbone whereby the routing information was adjusted before being delivered to the IPsec service so that it was routed through the desired path. This functionality was achieved with the use of an additional encapsulation method and transport mode IPsec. Some aspects from the work in Xbone has been recently standardized [80] and incorporated in the new version of the IPsec architecture standard [35][81].

Even though the implementation above provided a solution to routing at the VPN layer, it made the system vulnerable to routing security vulnerabilities and it did not eliminate the problem of dynamic IPsec policy management of SAs. Moreover, IPsec applicability to a variety of applications makes the security policy management more complicated as networks scale up. Networks are characterized by different kinds of traffic controlled by various entities and different security requirements. The high number of cryptographic parameters and the use of firewalls (as mentioned above) to control VPN access, further complicate the problem. Even though IPsec IKE could negotiate SAs, IPsec did not provide a mechanism for managing, negotiating and enforcing the security policies under which SAs operate [82]. The IETF's IPsec policy working group has recently proposed the interoperation of IPsec with policy management tools, such as the Common Open Policy Service (COPS) [83][84]. These tools will be able to provide the desired functionality in the future.

3.3.2.3. Interoperability of QoS and Real-Time Systems

FreeS/WAN IPsec was not implemented by strictly following the IPsec standard, which introduced interoperability problems. More specifically, the FreeS/WAN IPsec implementation dropped any packets that utilized certain IP header fields in tunnel mode. These IP header fields were required for the proper functionality of a real-time systems investigated as part of the NAVCIITI project [85]. In order to enable the transfer of those fields across the network, GRE headers were introduced. The GRE headers added 24 bytes per packet of overhead and increased set up complexity because GRE tunnels had to be initiated in a symmetric manner (both end-points of a connection).

Another issue that was considered in this investigation was the priority assignment of IPsec traffic by the QoS application. IKE and DNS traffic had to be assigned higher priority compared to other traffic so that higher guarantees could be provided for the delivery of that information. In this way, negotiating entities could obtain each other's key and establish an SA.

3.3.2.4. Network Management

A network management General User Interface (GUI), called TopoView, was installed on a subnet node and operated by querying the SNMP agents installed on the gateways. The subnet-to-gateway communication requirement of TopoView increased the IPsec management complexity because FreeS/WAN IPsec allowed only subnet-to-subnet communication and automatically restricted subnet-to-gateway traffic. As a result, there was a need to negotiate extra SAs, so that IPsec could allow traffic from the subnet node on which TopoView was "running" to each gateway and vice versa. This procedure would have been less complicated with the existence of a security policy management mechanism and/or an IPsec API. TopoView could dynamically specify its communication requirements to the security policy management mechanism, which could dynamically negotiate additional SAs to allow for specific traffic flows. The network management operation had no impact on key management or the IKE negotiation.

3.4. Discussion

This research generated the following conclusions based on the specific issues discussed above.

1) SA establishment with automatic keying (using IKE) avoided the complexity of maintaining encryption keys on each gateway and was less prone to errors as compared to manual keying. However, it did not significantly lower the key management complexity because authentication keys still had to be pre-configured on each node. The usage of DNSs with automatic keying minimized the key set up complexity.

2) Key negotiation frequently failed to establish SAs in a MANET environment, due to the inability to obtain authentication keys from the DNSs.
3) Only asymmetric keys could be used with the key distribution mechanism (i.e., DNS), in the existing FreeS/WAN implementation. Asymmetric keys were easier to distribute with the DNS since ownership of a public key did not compromise security.
4) The DNSs were responsible for managing the asymmetric keys and authenticating the validity of those keys. Mechanisms that dynamically validate the authenticity of the asymmetric keys throughout the network lifetime were absent.
5) DNSSEC did not interoperate with IPsec in order to secure transactions between the DNS and the IPsec gateways. Significant modifications were needed to interoperate it with FreeS/WAN.
6) There was no automatic management of IPsec policies to allow more flexibility in dynamically setting security policies among nodes and no mechanism to distribute those policies within the network.
7) Higher-level protocols could not communicate with IPsec to observe and influence how IPsec functioned. In addition, there was no network management tool to obtain the status of the IKE SAs.

The integration of IPsec with the various network technologies required a large number of adjustments to obtain the desired functionality. Some of the difficulties were due to deviation of the FreeS/WAN implementation from the IPsec architecture, as stated in RFC 2401 [35], in conjunction with FreeS/WAN implementation limitations. Additional difficulties incurred due to the inability to efficiently control the IPsec security policies, and assess the state of the SAs. Even though different mechanisms were proposed, they will not be available in the immediate future.

One of the objectives of this research was to investigate interoperation of the proposed KMS with the existing IPsec architecture. Based on the preliminary work presented in this chapter, the IPsec implementation could facilitate the implementation and testing of the proposed KMS. The integration would require the augmentation of the

DNS application with DCA functionality and the introduction of a KMS client application at each gateway. The main purpose of the client application would be to make IPsec aware of the various KMS functionalities that would increase the availability of the authentication information in the network.

3.5. Identification of Factors Impacting Research Goals

The generic problems and limitations encountered served as a guide for defining the architecture of the proposed KMS. These research problems and limitations were as follows.

1) The existence of multiple DNS minimized the key set up complexity. However, the DNSs acted both as repositories and CAs, and provided limited availability in a MANET that caused the IKE negotiation to fail. The proposed KMS should deploy additional mechanism to act as repositories and/or CAs to enable nodes to obtain the authentication information of their peers.

2) The DNSs were responsible for maintaining the public keys of nodes. The maintenance of the public keys on the DNS had to be done manually by the network administrator. In a dynamic environment it was unworkable for an administrator to dynamically carry out this maintenance. For example, new nodes always had to authenticate via out-of-band methods. The KMS should provide a number of methods to facilitate nodes joining the network by decreasing the dependence on out-of-band authentication.

3) Certificates were not distributed with the DNSs. As a result, the network nodes could not identify the user of a node but only the IP address of that node. The proposed KMS should use certificates to avoid this limitation.

4) The DNS did not provide information to the administrator with regards to node behavior in the network. In addition, the DNS did not provide to the administrator the ability to inform the network nodes of any malicious nodes with which they had SAs. At best, the network administrator could only delete the public keys of the malicious nodes. The KMS should adopt a mechanism that automatically and dynamically obtains information related to the behavior of nodes in the network. That information would provide the administrator or CA with justification for revoking the certificates of malicious node. The administrator or CA could then send RNs and inform the nodes in the network.

5) Unlike the DNS, the KMS should dynamically change its configuration based on the security requirements of the network as well as the network characteristics. Therefore, the KMS should utilize a security policy management mechanism.

6) Even though DNSSEC could not be deployed, the KMS should provide the DNSSEC functionality by guaranteeing the integrity of information transmitted among nodes.

3.6. Conclusion

This chapter presented preliminary work conducted for the research problem. The key management and IPsec technologies were investigated to gain an understanding of their current state. Testbed 1 allowed us to establish interoperation between the different operating systems using automatic keying for key negotiation. Testbed 2 investigated interoperability of IPsec by integrating technologies such as routing and QoS. Every aspect of each derived solution was analyzed with regards to key management. The overarching problems extracted from a variety of specific issues pointed the need for functionalities that increase availability, flexibility, and automate the key management functions, such as revocation.

Even though IPsec was superior compared to other security systems it offered limited functionality and flexibility to systems and end-users. The mechanisms proposed in the Internet drafts as well as in the recently standardized drafts, will likely increase the marketability of IPsec. The effectiveness of the KMS proposed for this research would be demonstrated via implementation. Chapter 4 describes in detail the architecture of the KMS.

Chapter 4

Proposed Key Management System

This chapter describes the proposed KMS solution that evolved from previous work conducted by other researchers and preliminary work as described in Chapter 3. Section 4.1 describes the objectives of the proposed KMS. Section 4.2 describes the structure of the KMS. The flexibility provided to the nodes in joining the network is analyzed in Section 4.3. Section 4.4 presents the concept of non-repudiation as utilized in the KMS and describes the behavior grading mechanism of the KMS. Section 4.5 covers SA establishment and the notion of TPs. Section 4.6 presents the synchronization requirements of the KMS. Section 4.7 describes the procedure for spawning new DCAs in the network. The service discovery requirements of the KMS are discussed in Section 4.8. Section 4.9 investigates interoperability of the KMS with the existing IPsec architecture. Section 4.10 discusses the applicability of the KMS to other network systems. A security analysis of the system is presented in Section 4.11.

4.1. Objective

The objective of this research was to develop a framework for providing redundancy and robustness in key management for IPsec SAs in a MANET. This framework would enable nodes in a MANET to set up different levels of trust among them. The system should be flexible enough to allow sufficient functionality for the nodes, while simultaneously providing security criteria to the nodes in order to establish trust between them. This research has been motivated after the investigation of previous work (see Section 2.9) and the investigation of the state of technology.

This KMS aims to:

1) increase service availability for existing nodes with regards to the ability to obtain a valid certificate,
2) introduce various mechanisms to allow new nodes to join the network,
3) incorporate a scheme that allows it to dynamically adjust the parameters of its functions, such as revocation, in order to match the activity in the network environment, and
4) minimize pre-configuration and reliance on pre-existing trust during its deployment.

4.2. Structure of the KMS

This KMS is flexible enough to provide sufficient functionality for existing as well as new nodes, while simultaneously providing reasonable security criteria to the nodes in order to establish trust between them. Functionality of the KMS refers to the ability of the KMS to provide services to the network nodes, the level of pre-configuration required for the KMS' nodes, and flexibility of the KMS to adjust to changes in the network environment (e.g., connectivity, malicious activity). Security of the KMS refers to the KMS' ability to provide guarantees of the correctness of the information supplied and its ability to trace the behavior of malicious nodes and respond appropriately.

In our network environment, we assumed that nodes could leave and join the network at any time. Nodes could generate their own cryptographic keys and were capable of securing communication with other nodes. This KMS used a modified hierarchical model of three levels, shown in Figure 4.1. The roles that were undertaken by the nodes in the hierarchical model were RCA, DCA, and TCA.

The nodes that volunteered to be DCAs had more important positions within the network (e.g., administrators) and were not constrained by battery power or processing capabilities. This assumption was reasonable since not all the nodes in a network have the same position or function. Any node that was not a DCA could assume the role of a TCA. Nodes could be TCAs regardless of their position or reputation in the network.

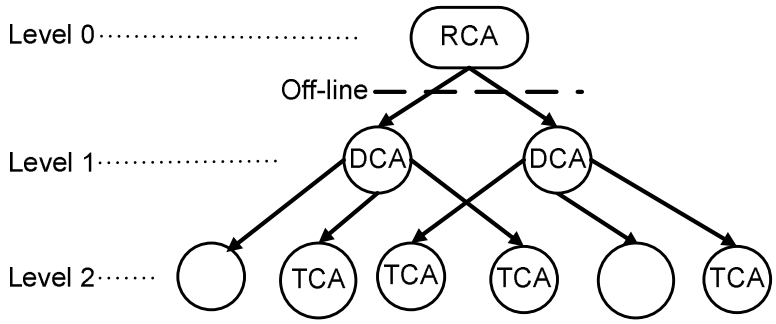


Figure 4.1. Modified hierarchical PKI model.

In the next subsections, we describe the KMS functions of the nodes according to their particular role in the hierarchical model. We focus mainly on certificate issuance and revocation.

4.2.1 Root Certificate Authority (RCA)

An RCA node was off-line so that it would be protected from malicious attacks. Rather, the dynamic nature of a MANET did not allow it to actively participate in the network and restricted it from contributing to the certificate management functions of the KMS, such as reissuance. Therefore, the RCA's main function was limited to providing certificates for nodes planning to register into the network in the future. This function was significant because any node that had an RCA certificate could register into the network and serve as a DCA. Registration was carried out dynamically and a node was authorized by sending a request with its RCA certificate to the existing control plane of DCAs. The notion of utilizing an RCA certificate to serve as a DCA differed from the work carried out in [2] and was based on work carried out in [3].

In [2], Zhou and Haas assumed that CAs were pre-configured with the public keys of one another meaning that there was pre-existing trust between those CAs. In a real world scenario, this pre-existing trust cannot exist unless the group of CAs' users physically meet prior to the KMS deployment and authenticate one another via out-of-band methods. Clearly, this is unworkable considering the nature of a typical MANET environment. Zhou and Haas [2] assumed that there is an off-line RCA that issued

temporary certificates for CAs. We believed that this method provided more flexibility compared to [2] because it did not rely on some level of pre-existing trust among a group of CAs. However, it was limited by the requirement that all nodes joining the network served as CAs. Thus, all nodes had to authenticate using out-of-band methods with an off-line RCA before obtaining a share of the private key of the KMS and serving as CAs. In our KMS, only a subset of nodes had to possess RCA certificates to serve as DCAs compared to [3], which made the method in [3] more applicable to our system. Therefore, by incorporating this method in our system, RCA certificates promoted minimal pre-configuration of the DCAs at the initial steps of KMS's deployment and facilitated dynamic deployment of additional DCAs during the network lifetime, as described in Section 4.7. An example of an RCA is VeriSign [86].

New nodes that possessed an RCA certificate and did not intend to be DCAs could enjoy increased availability when joining the network by temporarily establishing SAs with other nodes using that certificate. In addition, possession of an RCA certificate allowed the nodes to register into the network without physically encountering a DCA. Possession of an RCA certificate was noted on the nodes' DCA certificate. In this way, higher confidence was placed on the authenticity of the certificate by existing nodes in the network (see Section 4.3).

4.2.1.1. RCA Certificate

An RCA certificate was obtained via out-of-band methods. Examples of out-of-band methods are face-to-face communication and location-limited channels [87]. The RCA certificate format was similar to the X.509 v3 format [57][58][59]. A typical RCA-certificate contained the user ID, public key, and expiration of the certificate. It could also contain additional information such as the role of the user in a company.

4.2.1.2. Revocation at the RCA Level

RCAs utilized implicit revocation for their certificates. The life of the certificates was short and enabled nodes to establish short-term SAs at a lower trust level as described in Table 4.1. Using a shorter period forced the nodes to register into the

network with DCAs and accounted for the fact that the off-line RCA could not inform the network nodes of a revoked certificate. The certificates expired after a period of time long enough for new nodes to enter the network and register with a DCA. Once a node registered in the network and obtained a DCA certificate, it did not need to periodically update its RCA certificates. This requirement would be unworkable given that the RCA was off-line.

The lifetime of RCA certificates issued to nodes that intended to serve as DCAs was longer compared to other nodes in order to retain the validity of the certificate chains throughout the network lifetime. Even though those certificates had longer lifetime they did not allow for longer SA lifetimes and only provided a lower trust level. Since the RCA was off-line and the certificate validity was long, revocation of the DCA role in the network was handled by the entire control plane of DCAs, which checked and balanced one another, as discussed in Section 4.4.

4.2.2. Delegated Certificate Authority (DCA)

The KMS was comprised of a number of DCAs with the main objective of providing high service availability to the network nodes. Threshold cryptography schemes [2][3][4] decreased the availability of a KMS in a partitioned network due to the dependence on a group of CAs, which could be unavailable or compromised. In this KMS, we increased availability by allowing a single CA to generate a certificate. The KMS enforced the verification and detection of invalid certificates through the use of non-reputable transactions spanning over more than two nodes and through the use of the behavior grading scheme (see Section 4.4; Figure 4.2). DCAs issued certificates to nodes after presenting their credentials either via out-of-band methods or via online connectivity through peer nodes. More information on this functionality is presented in Section 4.5.

4.2.2.1. DCA Certificate

The DCA certificate format was similar to the X.509 v3 with some extensions [57][58][59]. The extensions were:

1) the method of registering in the network,
2) possession of an RCA certificate,
3) the node's KMS role in the network (e.g., DCA),
4) the TCA-signed certificates validity period (see Section 4.2.3.1), and
5) the behavior metrics of the node in the network, as discussed in Section 4.4.

The KMS utilized a different ruleset for the period of validity field of the certificate, in order to comply with the modified requirements of routine revocation as presented in Section 4.2.2.2. Instead of including the date of issuance and the date of expiration of the certificate, the date of registration into the network and the date of reissuance of the certificate were recorded. The date of registration indicated the total period of time that the node had been active in the network. The date of reissuance of the certificate indicated the last time that information on the certificate was modified. The certificates' expiration was dynamically imposed by the DCAs via routine revocation, the explanation of which follows.

4.2.2.2. Revocation and Security Alerts at the DCA Level

The DCAs utilized two revocation schemes to promote robustness and increase security in the network. As suggested in [3], nodes during revocation received a number of RNs from more than one DCA before taking the appropriate revocation action (e.g., dissolve an SA). (An example of this scheme is shown in Chapter 5 Figure 5.20.) The two methods of revocation were immediate and routine revocation. Immediate revocation was the process through which DCAs explicitly revoked the certificates of particular nodes. Immediate revocation was carried out at the DCA level based on security policies that were applied to the entire network. Certificates were revoked when certain levels of malicious activity were reached.

Routine revocation was introduced in order to increase certificate availability in a highly partitioned network environment. Routine revocation relaxed the time constraints that were imposed through traditional periodic reissuance of the certificates, also known as implicit revocation. With routine revocation, DCAs advertised a certain certificate

serial number or time of issuance before which all certificates would be invalid, as well as the time of expiration. The idea of advertising a serial number or time of issuance, called First Valid Certificate was suggested by Kaufman, Perlman, and Speciner [18]. The objective of the First Valid Certificate was to keep CRLs short and allow certificates not to have a predetermined expiration time. In this KMS we extended this notion by adding the *window of time* field, which indicated the time certificates would expire. The expiration date allowed nodes a window of time to reissue their certificate and reestablish their trustworthiness in the network.

Routine revocation was implemented because we felt that it was inappropriate to implicitly (periodically) revoke certificates for our network environment. With implicit revocation, all the nodes periodically reissue their certificates. The certificates are valid for a period of time indicated on their certificates, and nodes have to reissue their certificates before the end of that period of time. The CAs set this period of validity at the time of issuance or reissuance of a certificate. The inability to dynamically adjust this period of validity during the lifetime of a certificate introduces two problems. First, the CAs are unable to dynamically balance the overhead imposed by implicit revocation with the security of the system. We argue that a cooperative environment, such as a MANET, tends to be healthy most of the time and the majority of nodes abide by the network rules. Therefore, periodically revoking certificates introduces extra overhead in a resource-limited network when not necessarily needed. Setting a short validity period increases the overhead induced by implicit revocation. Increasing the period of validity decreases this overhead. However, a CA has to wait for the reissuance of the nodes' certificates to set a shorter validity period and thus use stricter security policies.

The second problem with implicit revocation is that it can decrease availability. Nodes may be unable to communicate with a DCA before their certificate expires. One way to diminish this problem is to have nodes reissue a certificate some time before the certificate expires. However, based on the dynamic nature of the network, nodes may not be able to accurately assess connectivity in the network and accurately predict the time needed to reissue a certificate. The utilization of the routine revocation introduced more flexibility by informing nodes that their certificate would expire and that they had to initiate a reissuance procedure. With the usage of the *window of time* field, nodes were

given enough time to locate a DCA and reissue their certificate. Therefore, routine revocation in our KMS provided higher availability as opposed to implicit (periodic) revocation.

DCAs carried out routine revocation based on the system-wide security policy. Stricter security policy implied that a greater percentage of certificates in the network would be reissued within a period of time. The value of the *window of time* field was determined according to the DCAs' ability to communicate with other nodes/DCAs.

Unlike other schemes, the revocation schemes in this KMS were complemented with security alerts in order to increase their effectiveness in informing nodes of malicious behavior. Security alerts were carried out at both the DCA level and at the node level. The DCAs utilized security alerts to inform the nodes when certain threshold levels of malicious activity were reached. At the node level each node had its own individual security policies that were reported to the DCAs during registration or certificate reissuance. The node's security policies specified the behavior values (see Section 4.4) that it could tolerate for its TPs in the network. Once those behavior thresholds were reached, the DCAs informed that particular node. The node could then decide on the action it would take, such as breaking the SA with its peer. The security alerts introduced more checks and balances into the network since nodes were made aware of the levels of malicious activity throughout the network lifetime. The thresholds of malicious activity were based on behavior grading (see Section 4.4). The security provided by these revocation schemes and security alerts were reinforced through the use of tools, such as non-repudiation (see Section 4.4).

4.2.3. Temporary Certificate Authority (TCA)

In an open system, new nodes that enter the network need to establish SAs with other peers when the DCAs are unavailable. A new node could obtain an RCA certificate with a short life and establish temporary SAs. This mechanism does aid new nodes joining the network and was incorporated in this KMS. However, it provided some level of inflexibility because it forced new nodes to register with an RCA using out-of-band methods. In order to further facilitate new nodes joining the network, we utilize TCAs.

In contrast to DCAs, any node in the network could serve as a TCA regardless of their position or reputation (see Section 4.4) in the network. TCAs signed temporary certificates for physically collocated new nodes to enable them to establish temporary SAs with existing network nodes. TCAs authenticated the new nodes using out-of-band methods or through other similar methods [87]. Even though the TCA functionality required out-of-band methods, there was a higher probability that at least one of the network nodes would be physically collocated with another one at any point in time as opposed to relying on a group of DCAs. An analysis of this probability is given in Section 5.4.3.4.

The TCA mechanism is different from the warrants scheme in [4]. In that paper, Bechler, et al. suggested that a node collected a series of warranty certificates from existing nodes via out-of-band authentication and sent them to threshold CAs to obtain partial certificates. We argue that this approach is not very dynamic and limits availability in highly partitioned networks because a node would have to authenticate out-of-band with a group of other nodes. In our KMS, possession of one or more certificates from collocated nodes was not required to obtain a DCA certificate. In addition, the certificate generated by a DCA after showing possession of one or more TCA certificates was not necessarily *fully-trusted*. When a certificate was generated, possession of TCA certificates was simply noted on the DCA certificate indicating that one or more other nodes verified the node's ID information. This functionality enabled each node to place a different trust level on its peer's certificate (see Table 4.1).

4.2.3.1. TCA Certificate

The TCA certificate format was similar to the X.509 v3 certificate but was extended to include the DCA certificate of the TCA that signed the certificate. Since the TCA's trustworthiness was displayed on its DCA-signed certificate via behavior grading, this extension provided the network nodes with an indication of the trustworthiness of the certificate issuer. By default, a TCA-signed certificate was given a lower trust level than a DCA-signed certificate. The level of trust of a TCA-signed certificate could be

translated by other nodes as being equal to, or less than, that of the TCA that signed the certificate, as shown in row 7 of Table 4.1.

The validity period of the TCA-issued certificates was controlled by the DCAs based on their network-wide security policies. (This period was recorded on the TCAs' certificate when issued or reissued by the DCAs.) The validity period acted as an indicator of invalid certificates issued by the TCAs, since TCAs attached their own certificates to the temporary certificates that they issued to the new nodes.

4.2.3.2. Revocation at the TCA Level

TCAs only utilized implicit (periodic) revocation. Certificates issued to new nodes were given short validity periods and allowed to expire. SAs established using TCA certificates had a short lifetime forcing /motivating nodes to register with DCAs and establish permanent credentials in the network.

4.3. Flexibility in Joining the Network

New nodes that possessed an RCA or TCA certificate could establish temporary SAs that had short lifetimes. In this way, new nodes were encouraged to register with a DCA using out-of-band methods to increase their trustworthiness, as previously mentioned. The KMS system introduced further flexibility for new nodes joining the network by allowing them to obtain certificates via DCAs in two ways: (1) via out-of-band methods or (2) by sending their credentials through peer nodes via online connectivity. A new node did not have to physically encounter a DCA because the authentication method was recorded on its certificate. The information served as an initial indicator of trust that could be placed on the certificates (i.e., the nodes' identity) during an SA negotiation.

Table 4.1 shows an example of how the combination of authentication method and certificate possession information was translated by a node to various trust levels. These trust levels could vary according to each node's security policy. Trustworthiness of each node might be loosely classified using a scale from 1 to 100 where 1 is the least

trusted level and 100 is most trusted level. Out-of-band authentication was considered more trustworthy as it implied face-to-face contact and earned nodes higher trust from their peer nodes during SA establishment. In this way, a node was motivated or forced to authenticate with a DCA via out-of-band methods in order to increase the trustworthiness that could be placed on its certificate by peer nodes. However, connecting through peers to obtain a certificate promoted more flexibility for new nodes, as described above. Based on row 1 of Table 4.1, if node X during an SA establishment presented its DCA certificate to node Y, which was obtained by using its RCA certificate and authenticating via out-of-band methods, then this registration process provided the highest level of trust with regards to the authenticity of the node's ID. As a result, node X was given an initial trust level of 100 by node Y. However, in row 3 of Table 4.1, authenticating via peer connectivity and possessing an RCA certificate provided less confidence on the trustworthiness that could be placed on a certificate because of the inability of the DCAs to connect and dynamically obtain revocation information from an RCA. Even though an RCA certificate did not expire, it could have been revoked by the RCA. Therefore, the node was given a trust level of 90 by its peer. Furthermore, in rows 4 and 5 of Table 4.1, if node X obtained its DCA certificate by connecting to a DCA through peer nodes and did not possess an RCA certificate, then it was given lower trust levels by node Y. Node Y trusted node X's certificate more in the scenario of row 4 compared to the scenario of row 5 because node X showed a TCA certificate to the DCAs, proving that another node in the network had verified the same credentials. Authenticating via the DCA through peer nodes without possessing any type of certificate would reasonably yield the lowest level of trust.

Thus, this scheme of trustworthiness assignment based on the authentication method and possession of TCA or RCA certificates promoted more flexibility because a new node that had no behavior grading (see Section 4.4).could establish SAs with peer nodes at various trust levels.

Table 4.1. Trust Levels of Node X after Establishing an SA with Node Y

	Node X Cert.	Authentication Method with a DCA		Certificate Possession		Certificate Duration/ SA Lifetime	Y's Initial Trust level
		Out-of-band	Peer Connectivity	TCA	Root		
1	DCA	√		N/A	√	Long	100
2	DCA	√		N/A		Long	95
3	DCA		√	N/A	√	Long	90
4	DCA		√	√		Long	80
5	DCA		√			Long	65
6	RCA				√	Short	85
7	TCA			√		Short	≤TCA's
1 => lowest trust level				100 =>highest trust level			

4.4. Non-repudiation and Behavior Grading

To balance the flexibility and the increased availability of the KMS, security was provided by introducing two concepts in addition to revocation and security alerts: *non-repudiation and behavior grading*.

First, all transactions of the KMS were verified by at least two other nodes or DCAs in a non-reputable manner. The originating nodes signed the transactions of the KMS, providing proof of the origin of each transaction. This functionality was important because it prevented any node or DCA from modifying the data transferred and allowed the detection of malicious activity by those nodes. An example of this procedure is certificate issuance, which is shown in Figure 4.2. In step 1, Node A digitally signed its personal information and sent it to DCA1. DCA1 authenticated node A out-of-band and generated its certificate. In step 2, DCA1 forwarded node A's signed information and/or node A's certificate to DCA2 and DCA3. In step 3, DCA 2 and DCA3 sent an acknowledgement to node A, which could be a hash of its signed information. In this way, DCA1 could not modify the information because it was signed by node A. DCA1 could elect to communicate with one or more DCAs, based on the security policy of the network as well as the network environment. Since the certificate issuance involved more than one DCA, it provided a balance of power and ensured information propagation. Any

malicious behavior was recorded by the behavior grading scheme, the description of which follows.

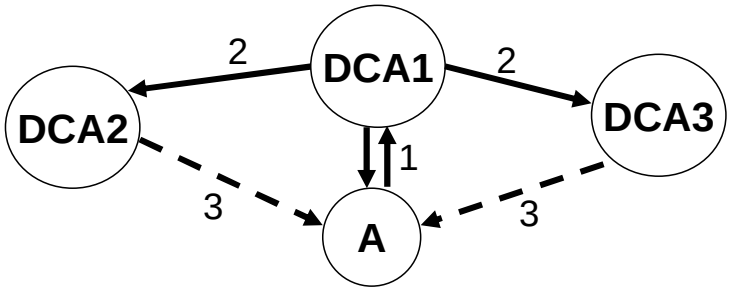


Figure 4.2. Certificate Issuance over multiple DCAs.

The KMS maintained sufficient levels of security by combining node authentication with an additional element, *node behavior*. A behavior grading scheme required each node to grade the behavior of other nodes. It was envisioned as a central, data processing layer, as shown in Figure 4.3. At the lower layer, an intrusion detection system (IDS) [88] or monitoring scheme [89][90] could provide periodic performance observations to the network nodes. In a self-organized environment, such as a MANET, each node could utilize both the IDS information and the behavior grading information from the KMS to decide whether to trust or distrust a peer. In [91], we provided a method that a node could use to aggregate feedback from the behavior grading scheme of the KMS and from an IDS to produce a reputation index. The node could use that reputation index and decide whether to trust its peer based on its individual security policy. Once the node decided to trust or distrust its peer it reported its trust decisions to the behavior grading mechanism of the KMS. The behavior grading scheme would then collect this information and tie it to the certificates of the nodes. Existing nodes could in turn use these credentials to negotiate with their peers and decide whether to establish or dissolve SAs.

In addition, the KMS utilized this information to set its security policies related to reissuance, revocation, and security alerts thresholds. Thus, the KMS could dynamically assess the malicious activity in the network and initiate revocation or utilize security alerts. This layered approach meant that the behavior grading system was independent of the type of IDS since nodes could utilize any type of feedback generated by an IDS, such as routing activity.

The premises of the behavior grading scheme were based on existing concepts that were deployed in reputation management systems [88][90]. However, the significance of the behavior grading scheme was that its parameters recorded the results from the aggregate input collected about nodes' activity instead of grading a particular activity (e.g., a node does not forward packets). Nodes could collect different inputs from one or more IDSs or any other observations that could be aggregated and recorded in a binary form: trusting or distrusting a node. The notion of utilizing different types of feedback information corresponds to real life situations. In real life, we choose our trusted friends by considering a number of different situations and experiences with an individual rather than one specific experience. Each individual has different criteria when deciding whether to trust someone and can give more emphasis on different experiences. Another important aspect of aggregating input in binary form was that the overhead imposed by the behavior grading scheme was not high except during the initial deployment of the network when all nodes desired to establish SAs with one another. Furthermore, in a network that is healthy most of the time, fluctuation of trust and thus reporting of trust/distrust would tend to be low. This concept is demonstrated with the pyramid shape in Figure 4.3. Information flowed from IDS to the behavior grading scheme to the KMS. The higher the layer of a function on the pyramid, the less information needed to be communicated to a fewer number of DCAs.

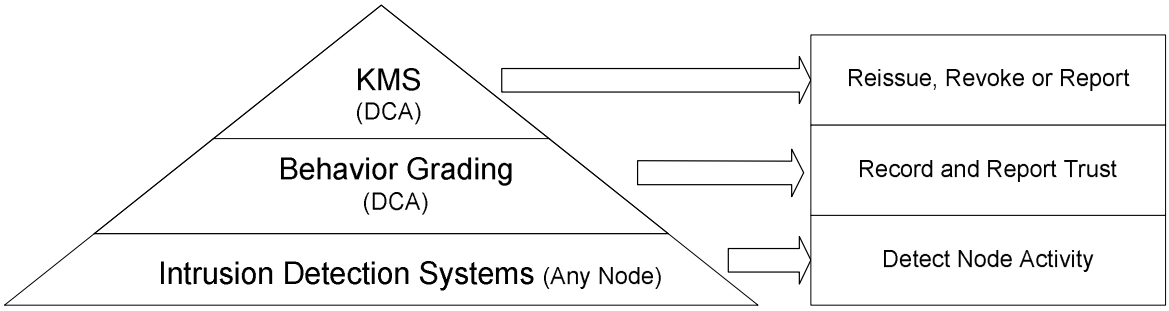


Figure 4.3. Incorporating node activity into the KMS via behavior grading.

Overall, the behavior grading scheme provided the nodes and the KMS with a layer of abstraction of the trustworthiness of nodes, which was based on the overall activity of the nodes in the network. Through behavior grading, the nodes were motivated

to do what was best for them while at the same time contributing to the entire network. Nodes were not as dependent on strict identity verification since they also had the ability to judge the trustworthiness of a peer node based on its behavior in a network as perceived by other nodes in the network. As a result, the need to periodically reissue certificates via routine revocation to enforce higher security in the KMS was not as frequent.

The behavior grading scheme recorded the nodes' level of trustworthiness using three parameters: positive reputation, negative reputation, and a complaint counter. *Positive reputation* indicated the number of TPs of a node. After an SA was established between two nodes, the two nodes reported the ID of their new TP to a DCA as shown in Figure 4.4. The nodes reported their trust to a different DCA than the one that distributed the certificates to them prior to the SA establishment. This functionality allowed DCAs to check the integrity of the certificates distributed by other DCAs, and prevented any DCA from modifying the reputation or any other information of a particular node. Furthermore, it was required that both nodes registered their SA to get a positive reply that their trust had been recorded on their certificates. This mechanism enforced the notion of having more than two nodes be involved in a transaction and discouraged selfish nodes, since their peer was informed if they did not report their trust. Overall, positive reputation motivated the nodes to collaborate and improve their reputation and allowed re-socialization of nodes that may have being wrongly victimized in the network. Messages T1 and T2 in Figure 4.4 represent the report of trust from each party. The messages were signed from each party providing non-repudiation. Message T3 was comprised of messages T1 and T2 as well as the updated certificates of each peer node. Receiving the updated certificates was optional, as discussed in Section 5.3.

A node's *negative reputation* indicated the number of peers that no longer trusted a particular node. It was motivated by the notion that nodes in a network, like people in our society, have a natural tendency to complain about other nodes or people. If a node deemed that its peer was no longer trusted it could elect to break the SA with its TP and inform the DCA. In this KMS, a complaint inferred that the complaining node had no SA with that particular peer.

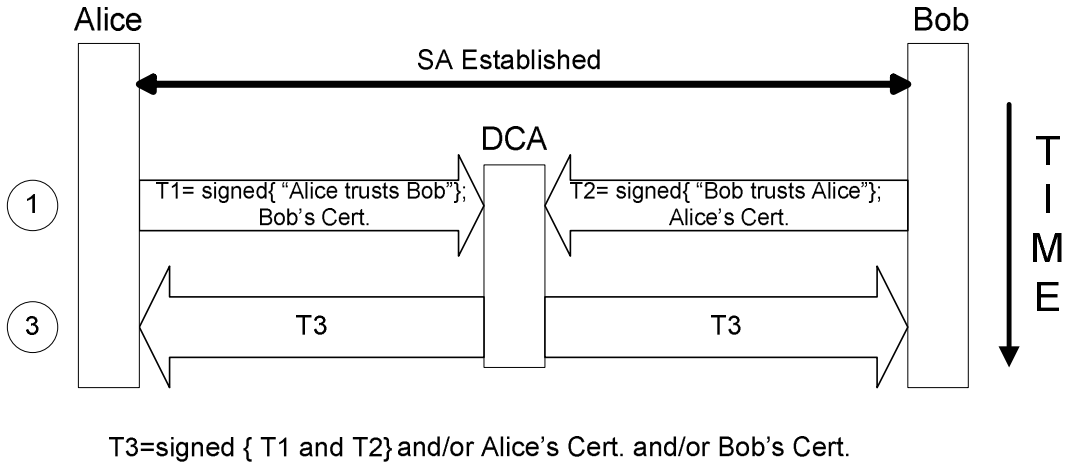


Figure 4.4. Positive reputation recording.

Once the DCA obtained the complaint from a node, it recorded the complaint and sent a copy of the reply and/or updated certificates to both nodes that had an SA (see Figure 4.5). In this way, the malicious node was in effect isolating itself by decreasing the number of nodes that trusted it in the network. In addition, the notification sent to the malicious node indicated immediate repercussions by negatively affecting that node's reputation.

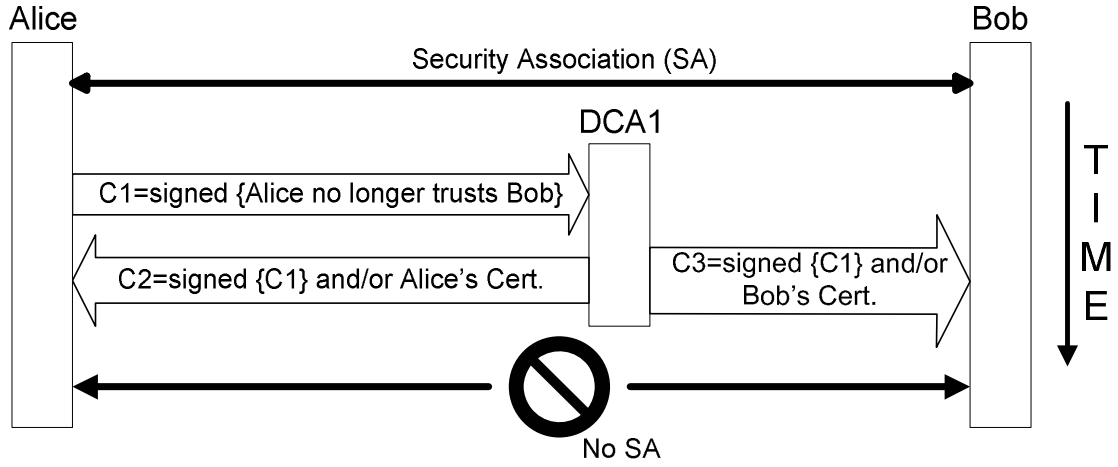


Figure 4.5. Recording of negative reputation.

However, if a malicious node complained about a "good" node, then the notification from the DCA would inform the "good" node to stop communicating with

the malicious node that complained, in case they still had an SA. Thus, in effect the complaint isolated the malicious node. Even though, in this scenario the behavior record of a "good" node would be falsely modified, the KMS would not revoke the certificate of that node based on a single complaint. The KMS judged revocation based on the overall behavior of the node in the network with regards to the network-wide security policy. Furthermore, the ability to give a "bad" grade by reporting distrust was independent of the existing reputation of the nodes. In this way, "good" nodes turning "bad" were prevented from "attacking" the rest of the network nodes.

A malicious node roaming the network could only complain about other nodes with which it had an SA. Therefore, the number of nodes that could complain about a particular node was limited to the number of its TPs. Furthermore, a malicious node could only submit one complaint for the peers with which it had SAs, thus avoiding a stacking attack. (A stacking attack occurs when a node keeps complaining about its peer, and builds up that peer's negative reputation.) In Figure 4.5, message C1 represent the complaint of a peer (Alice). Messages C2 and C3 were comprised of C1 and an optional updated certificate for each peer. The messages of the complaint process were again signed providing non-repudiation.

The *complaint counter* was recorded at the same time that the negative reputation occurred. The complaint counter showed the number of times that a node complained about its TPs and, thus, discouraged a malicious node from roaming around the network, establishing SAs with peer nodes and then complaining about them. A node would not associate with a peer node that complained about a large number of its TPs.

The criteria used for revoking a certificate reflected both the node's behavior history as well as its recent behavior. The decision to revoke was therefore based on both the total number of complaints as well as the frequency of complaints about a node. These values were modified by the DCAs to reflect the dynamic changes in a MANET based on the existing complaint frequency of the nodes in the network. Allowing for a high frequency of complaints made the system less sensitive to sudden changes in behavior of nodes and vice versa.

DCAs set the network-wide policies for the various parameters of the KMS, such as the frequency of reporting malicious activity via security alerts, by negotiating a set of

policies with each other. These policies were based on each DCA's view of the network and determined by factors such as network environment and malicious activity in the network.

To provide a *balance of power* among DCAs and increase the security of the system, each DCA could also grade its peer DCAs. The grading was based on the correctness of the DCAs' functions. Malicious activity could occur in the form of incorrectly displayed information by the DCAs and it was identified by the lack of proof that justified the information modifications. This requirement for balance of power among DCAs required a network to have at least three DCAs.

DCAs were given some immunity by the behavior grading of the KMS, as a motivation for their services in the network. This immunity protected them against a number of malicious attacks. The level of immunity was based on the network size and the type of environment over which the network was deployed (e.g., hostile versus friendly). The immunity was again set up by a series of negotiations among DCAs.

4.5. Facilitating SA Establishment

The certificates of nodes were required during an SA establishment so that nodes could authenticate each other. The certificates in any KMS can be stored on CAs or on repositories. Storing the certificates on CAs simplifies the management of the certificates and provides more control because expired or revoked certificates can be immediately deleted by the CA. However, this approach imposes a higher workload on a CA because the CA needs to store and send the certificates to the nodes. Furthermore, it may provide limited service availability in case all of the CAs are unavailable due to network partitioning. Specific nodes can be selected as repositories, in addition to the CAs, to distribute the certificates on behalf of CAs, increase availability, and decrease the workload of the CAs. However, utilizing repositories introduces communication overhead because CAs have to periodically update the revoked certificates stored on the repositories. In addition, as the various functionalities of the KMS are spread over more nodes the KMS becomes more vulnerable to security attacks. Another limitation is that the selection of a node to be a repository should be carefully assessed so that a node is not

malicious and it can be trusted to provide the certificates to the nodes in place of the DCAs. In our scheme, we utilize TPs to serve as repositories. (TPs were the nodes that trusted a particular peer, i.e., shared an SA with that peer.) TPs only stored the certificates of the nodes that they trusted. This approach provided higher guarantees with regards to certificate distribution because those TPs were more likely to distribute certificates for their peers. By default, the TPs automatically obtained the certificates of peer nodes during SA establishment, thus simplifying the repository selection process and dynamically assigning repositories. In this way, the workload of managing the certificates was spread among all the nodes in the network.

At the beginning of an SA establishment, an *existing* network node queried any of the available DCAs. If all of the DCAs were unavailable due to network partitioning, a node obtained a peer's certificate by utilizing the TPs of its peer, as shown in Figure 4.6. In this example, node C wanted to establish an SA with node F. Since all DCAs were unavailable, node C queried node F for its list of TPs. Once node C received that list containing the addresses of node A, node D, and node I, it could query any of them for F's certificate. The selection of which TPs to query was based on node C's perception of their trustworthiness. First, node C would query any "mutual" TPs, in this case node D, that it shared with the peer node. If node D was unavailable, the second approach would be to query the rest of the TPs in node F's list, nodes A or I, as those were unknown and presumed less trustworthy. Node C could also elect to obtain node F's certificate from one or more TPs, if available, and select the most recent one with the latest behavior grading of node F.

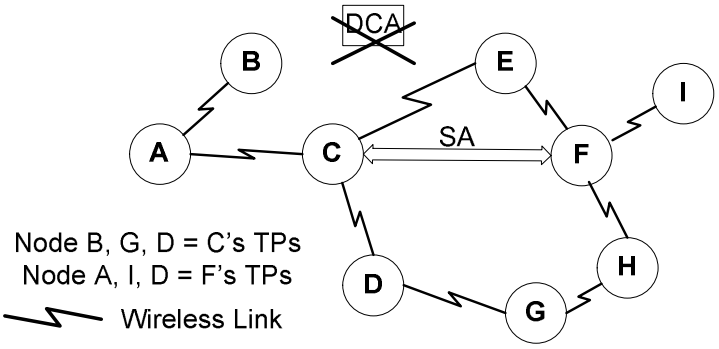


Figure 4.6. Establishing SAs without the help of DCAs.

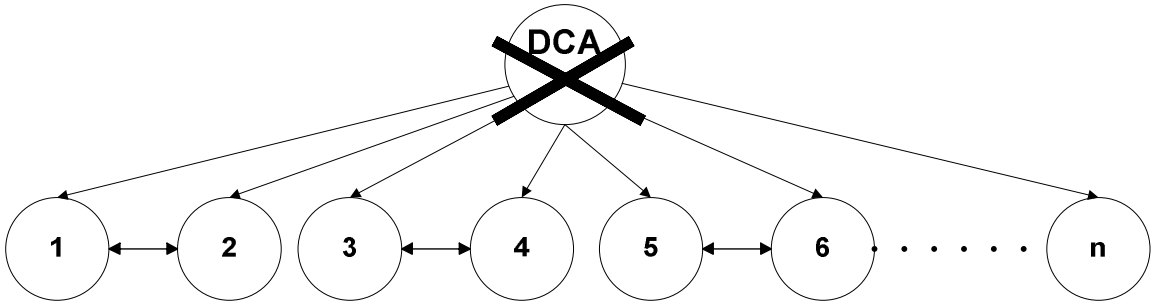


Figure 4.7. Base example for Equation 4.1.

A high number of TPs per node increased the probability that a node could establish SAs with its peers, an advantage that can be proven mathematically. In a network where every node trusted at least one other node, as shown in Figure 4.7, and all DCAs were unavailable, the number of additional SAs that were established could be calculated using Equation 4.1.

$$\frac{N(N-1)}{2} - \left\lceil \frac{N}{2} \right\rceil = \left\lfloor \frac{N(N-2)}{2} \right\rfloor \qquad \text{EQN. 4.1.}$$

In the equation above, N is the number of nodes in the network, excluding DCAs. Equation 4.1 assumed that each node required the establishment of an SA with every other node in the network when no DCAs were available. The ceiling value of $N/2$ accounted for the scenario that the network might have an odd number of nodes. Taking the ceiling value ensured that all nodes have at least one SA with a peer node. For example, in a network of 5 nodes 3 SAs would be required for every node to have at least one SA: (1) an SA between node 1 and node 2, (2) an SA between node 3 and node 4, and (3) an SA between node 4 and node 5. Node 4 would end up with two SAs.

Table 4.2 was derived from Equation 4.1 and shows the effectiveness of the proposed scheme with regards to availability. For example, in a network of 80 nodes, 3,120 additional SAs could be established without requiring the presence of a DCA. The effectiveness of this functionality did not imply that the presence of DCAs was not required at all, but rather that the system could be independent of DCA availability during SAs establishment, if needed.

The notion of "friends" that was used in [63] is different from the notion of TPs used in our KMS. Capkun, Hubaux, and Buttyan [63] presented the idea that each node could build SAs with the help of its existing (i.e., pre-configured) friends. Based on the results presented, they assumed a number of pre-existing friends for each node and equated SAs with complete trust. According to [63], two friends "trusted each other to *always* provide correct information about themselves and they had already established SAs between each other." Friends signed certificates for other nodes with which they shared SAs. In our KMS, two TPs did not *always* trust one another to provide the correct information even though they shared an SA. We assumed that absolute trust did not exist between friends and adopted a behavior grading scheme that integrated a node's behavior with its identity. In addition, TPs did not sign certificates for one another but only acted as repositories when the DCAs were unavailable. The number of a node's TPs was not fixed but fluctuated according to its reputation/behavior within the network.

Table 4.2. Additional SAs Established

Nodes	SAs	Nodes	SAs
2	0	**20**	180
3	1	**30**	420
4	4	**40**	760
5	7	**50**	1200
6	12	**60**	1740
7	17	**70**	2380
8	24	**80**	3120
9	31	**90**	3960
10	40	**100**	4900

It is important to note that the ruleset of the certificates used in the KMS differed from the X.509 v3 ruleset because multiple versions of a single certificate could exist within a window of time. That window of time and thus the number of versions of a single certificate was controlled by KMS through routine revocation (see Section 4.2.2.2). Each version of the certificate of a node could *only* potentially contain different behavior grading information, but not different personal information (e.g., public key or ID). The DCAs held the certificates with the most up-to-date behavior grading

information. The TPs held less recent versions of a certificate depending on when they established an SA with a node.

A node could establish an SA with a peer according to its individual security policies or trust thresholds, if it was satisfied with the reputation of that peer that was recorded on the peer's certificate. If the node did not trust its peer based on that certificate and the certificate was not the most recent one, the node could obtain the latest certificate from a DCA, or if the DCAs were unavailable, obtain the certificate from other TPs of that peer. Since the possession of the node's most updated certificate by its TPs increased its chances of successful future SAs establishment with other nodes, a node was encouraged or motivated to periodically obtain and distribute an updated copy of its certificate to its existing TPs. In addition, through frequent updates the node reinforced its status as a trustworthy node. The frequency of this update was based on the node's security policy.

Copies of the certificates of a node including behavior grading were stored on all the DCAs as well as on the TPs of that node. If a node's certificate was requested from a DCA and the DCA had recently received updated behavior grading information for that node, then the DCA reissued an updated certificate to reflect the latest behavior grading prior to distributing it in the network.

4.6. DCA Synchronization

DCA synchronization contributes to service availability by keeping all records/certificates of the nodes up-to-date and distributing them to all the DCAs in the network. A detailed explanation of DCA synchronization together with related optimizations is beyond the scope of this research. It is important, however, to specify the requirements of synchronization related to this KMS. The KMS required a method to ensure that the information was passed among nodes/DCAs in the network. This functionality was achieved by having a DCA communicate directly with a subset of other DCAs, the Supporting DCAs (SDCAs), to inform them of new information such as behavior grading, reissuance, and revocation. The SDCAs would in turn sent proofs of the received information to the corresponding nodes. For the case of revocation this

information in the form of RNs would also be used to trigger revocation. (A node might dissolve an SA after receiving more than one RN.) To limit the overhead imposed by this security requirement only a subset of DCAs, would need to send proof of the information to the nodes. The number of SDCAs would depend on the policies of the KMS. A higher number of SDCAs would increase the overhead because more proof would be sent to nodes. However, it would also increase availability and provide more checks of information propagation among nodes and DCAs. Figure 4.8 shows a generic structure of the transactions related to this requirement. Information flows from node A to the DCA, to the SDCA, and then back to node A. Examples of this structure are also shown in Chapter 5 Figure 5.20 for revocation and Figure 5.21 for certificate issuance. Information propagation between SDCAs and the rest of the DCAs could be carried out via schemes such as SyncML described in [92].

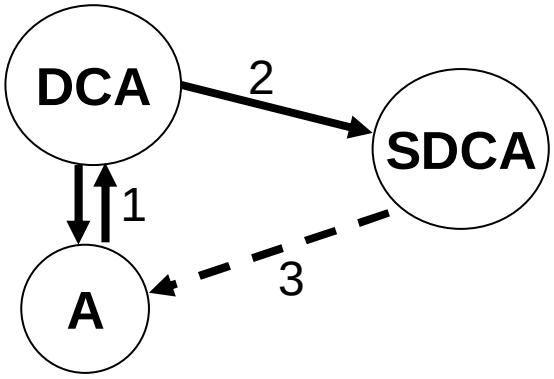


Figure 4.8. Enforcing information propagation through synchronization.

4.7. DCA Spawning

This section provides a general structure for carrying out DCA spawning. Similar to other functions reliance to multiple parties is required to ensure proper functionality. DCA spawning allowed new or existing nodes to dynamically carry out the role of a DCA. This functionality facilitated: (1) the distribution of the DCA role, from a DCA with depleting power resources to a node with sufficient resources, and (2) the dynamic increase in the number of DCAs in the network. If a particular DCA was unable to

contact other DCAs over a period of time, it would spawn a new DCA to ensure a balance of power in the system. The period of time was depended on the malicious activity in the network (e.g., the number of complaints) as well as the connectivity of the network. New DCAs were spawned in three steps as shown in Figure 4.9. In Step 1, existing DCAs broadcasted a message to the network's nodes requesting nodes to volunteer to be DCAs. The assumptions for being a DCA were discussed at the beginning of Section 4.2. Once a node volunteered to be a DCA, it obtained an authorized certificate and could assume the role of a DCA. In Step 2, the new DCA queried three random DCAs and obtained the certificates of the network nodes, as well as any other unaccounted incoming or individual updates to those certificates. The new DCA applied any updates, compared the certificates received and discarded duplicates. In Step 3, the new DCA showed its authorization certificate to the rest of the DCAs together with any required synchronization information (e.g., certificate issuance date) so that they could start sending it their own updates.

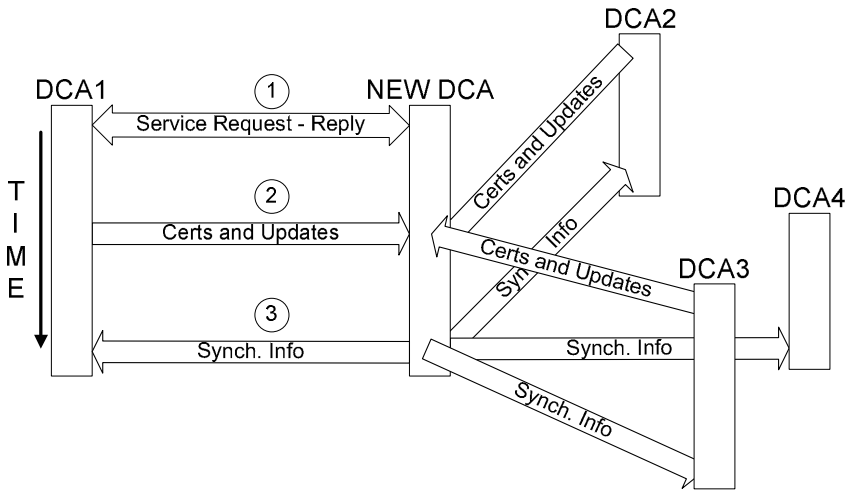


Figure 4.9. DCA initialization phase.

The set up phase for new DCAs included a short period of "immunity" to protect them against complaints from malicious nodes and to balance any minor connectivity or synchronization discrepancies that might lead to bad grading. This immunity was either a limited number of "bad" complaints or a certain frequency of received complaints. The existing DCAs set the immunity parameters based on their perception of the level of

malicious activity in the network. Even though this functionality was relatively costly compared to the other KMS functions it facilitated the dynamic set up of DCAs.

4.8. Service Discovery

The primary function of a service discovery mechanism with regards to this KMS would be to inform nodes of the DCAs' identities in the network, as well as to perform some load balancing by distributing RNs or CRLs on behalf of the DCAs. The DCAs could authorize nodes to serve as service discovery nodes (SDNs). SDNs could broadcast their own identity in the form of signed authorizations from the DCAs. The network nodes could then query the SDNs for the required information. All the information transmitted by the SDNs would be signed by the DCAs, thus preventing malicious SDNs from transmitting wrong information into the network. Similar to the structures of other schemes, such as revocation, an SDN could refrain from transmitting information such as RNs until receiving justification or approval from multiple DCAs. This KMS could be extended to grade SDNs, using a method similar to that of grading network nodes.

This research discusses only the functionality related to service discovery required by this KMS. Developing, optimizing, and analyzing the impact of service discovery mechanisms is beyond the scope of this research.

4.9. IPsec Interoperation

Testbed 2 in the preliminary work was not a standard form of a MANET, due to its usage of subnet nodes that did not contribute to a MANET operation (see Section 3.3). However, the KMS may be applied to this form of a MANET as well as to other standard forms. The basic difference in the deployment of IPsec in this environment would be the IPsec modes used.

We assumed that nodes would have sufficient bandwidth and battery power to support the KMS. This was a reasonable assumption because the overhead of setting up an SA and sending this reputation information to the KMS would tend to be lower than

the overhead of encrypting and exchanging information with peer nodes via IPsec tunnels.

Preliminary work identified the current state of the technology and pointed out that the KMS and the current IPsec architecture should use asymmetric keys stored in multiple KDCs. This KMS provided these multiple KDCs with the use of DCAs and TPs. In the preliminary work, multiple DNSs were serving as repositories by storing the public keys of the nodes. The public keys of the nodes were managed by the system administrator. The KMS would require that the DNSs act both as repositories and CAs. The management of keys would be automated for most of the KMS functions, except for functions such as out-of-band authentication. The KMS would utilize X.509 v3 certificates removing the need for two levels of authentication as discussed earlier in the preliminary work. We assumed the existence of monitoring tools to provide information on the status of SAs or to negotiate security policies among nodes.

An alternative approach to obtaining the required CA functionality would be to replace the DNS with independently programmed CA application software. However, for the purpose of this discussion we assume that the functionalities of the DNS were augmented to facilitate the KMS.

4.9.1. Behavior Grading

The reputation information of the behavior grading scheme of the KMS was directly mapped to the operations of the current IPsec implementation. Positive reputation could be reported to the KMS once an IPsec SA was successfully negotiated with a peer node. Negative reputation could be reported to the KMS after the node brought down an IPsec tunnel that was shared with a malicious node. The negative counter would be automatically updated once a complaint was reported to the KMS.

4.9.2. Certificate Issuance/Revocation

The certificate issuance/reissuance could be performed through the DNS. The DNS would generate the certificates and distribute them to the network nodes. Existing DNSs also have a built-in synchronization mechanism, which might be used to

synchronize information updates. Certificate revocation would be carried out by the DNS. The DNS and SDNs would then inform all of the TPs of the compromised node.

4.9.3. Key Negotiation

IPsec nodes negotiated an IPsec tunnel by establishing SAs using IKE. To ensure interoperation of the KMS with the IPsec implementation, the IKE would have to be modified to be made aware of the different levels of redundancy that the KMS offered.

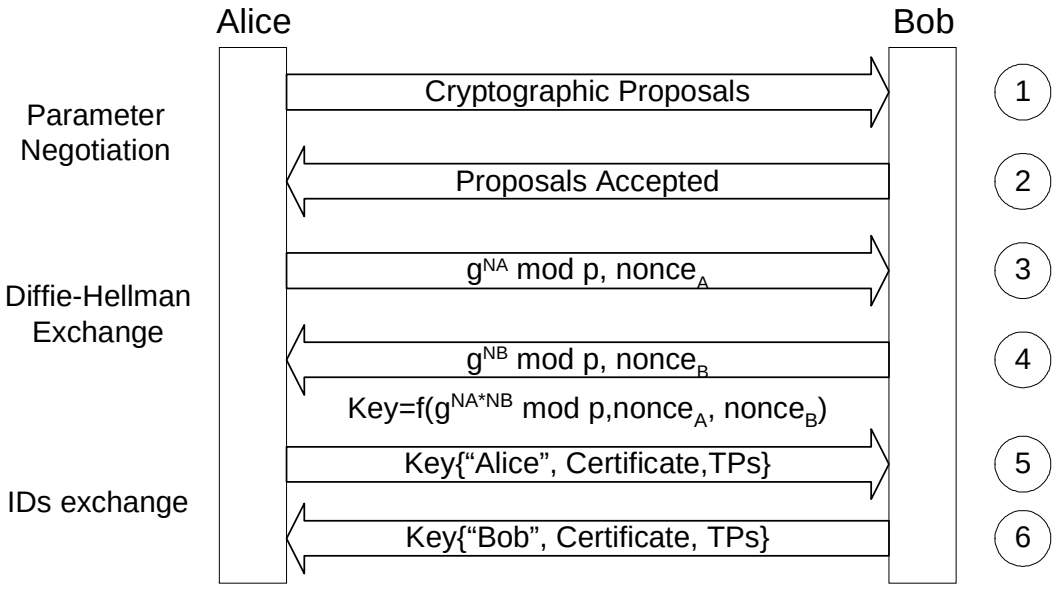


Figure 4.10. IKE phase one main mode with certificates and TPs.

Prior to an SA establishment, nodes would exchange their list of TPs and their most recent certificate (optional). This exchange would take place during the 5^{th} and 6^{th} transaction in main mode as shown in Figure 4.10. A software patch exists that enables IPsec to utilize X.509 v3 certificates. However, IKE would have to be further modified to accommodate the TPs field. If the nodes did not exchange their certificate, or wanted to get the certificate with the latest behavior grading information, they could query any of the DNSs. If all of the DNSs were unavailable, then the node might query any of the TPs of a node. Once authentication succeeded, the IKE phase 2 mode would establish IPsec SAs. Nodes would negotiate the various IPsec protocols and algorithms and generate the keys to be used for data encryption.

4.10. Additional Applicability

This KMS may be incorporated into other network operations such as routing. Secure routing protocols that were proposed, assumed the existence of a mechanism that distributed keys to the various nodes [93][94][95][96][97].

The trust relationships among nodes derived from this KMS can also be applied to make routing decisions. Yi, Naldurg, and Kravets [98] proposed a routing technique called Security-aware Ad hoc Routing (SAR) that was capable of incorporating trust levels into the route discovery process. SAR assumed the existence of organizational hierarchies that could associate a value with a privilege level. The KMS of this research is well suited for SAR since the behavior of a node could be used to determine privilege levels. SAR's simple comparison operator could sort these levels and assign the nodes a position within the hierarchy.

Furthermore, this KMS, in conjunction with IPsec, could be used to provide security services to routing protocols. In [99], Papadimitratos and Haas proposed a route discovery mechanism that mitigated route discovery vulnerabilities and provided correct connectivity information. The only requirement of that protocol was the existence of an SA between the nodes initiating the query and the desired destination nodes. The KMS with IPsec could provide those SAs.

4.11. Security Analysis

This section analyses the security of the proposed KMS and discusses the functionality that was deployed to provide the security of the system. The analysis does not concentrate on the cryptographic aspects of the algorithms that were utilized, but rather on the architectural aspects of the system. The analysis was conducted by considering the way a malicious node affected the various entities of the KMS by attacking any of the services that they provided.

Figure 4.11 shows the various services that might suffer from security attacks. A malicious node could have the role of a DCA, a TCA or a Generic Node (GN). A GN was a node that was not an RCA, DCA, or TCA. The RCA was not considered in the

analysis, as it was off-line. For the purpose of this security analysis, a malicious node could have access to the private key of a peer node or physical access to that node.

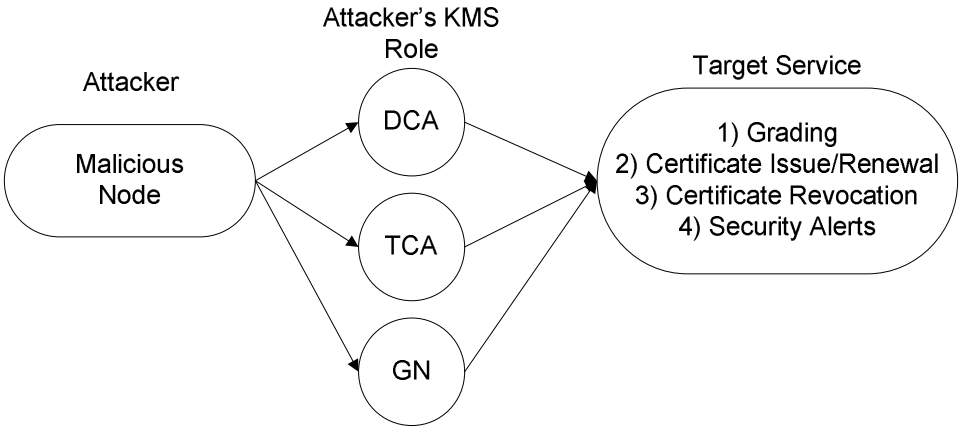


Figure 4.11. Roles of a malicious nodes in theKMS.

4.11.1 DCA Security Analysis

The DCAs had the most important role in the MANET. The DCAs could not change any of the behavior grading parameters of the other nodes because they had to show proof of the transactions to the SDCAs, which verified the validity of the respective modifications. These proofs were signed by the nodes using their private keys, thus providing non-repudiation.

A DCA could not by itself revoke the certificate of another node. If it issued and distributed an invalid RN or security alert, it was given a "bad" grade by the rest of the DCAs. Moreover, two or more RNs were required to convince nodes to dissolve SAs with their TPs, which made this attack ineffective.

A DCA could not issue or reissue an invalid certificate because the certificate generated by the DCA was verified by the node that requested the service. In addition, a node could complain about the invalid certificate by sending the invalid certificate to other DCAs to validate it. Similar to behavior grading, a DCA had to show proof of the certificate issuance transaction to the rest of the DCAs, which prevented it from maliciously modifying a certificate.

In the case of collusion between a DCA and a node this system could suffer from the existence of an invalid certificate, i.e., node with invalid ID, circulating the network. To mitigate this attack a node could be forced to authenticate out-of-band with more than one DCA or possess one or more TCA certificates before validating its certificate across the control plane of DCAs. However, as is the case with any system, e.g., threshold cryptography, stricter security policies for this system would decrease availability.

A compromised DCA was restrained from distributing invalid certificates with higher positive reputation during an SA establishment because nodes sent their TPs' certificate to other DCAs during positive reputation reporting, as described in Section 4.4. The DCAs could then detect invalid certificates distributed by the compromised DCA and take appropriate DCA grading or revocation actions. In addition, nodes could obtain a certificate from multiple DCAs or TPs before making their trust decisions, which further mitigated this attack. If an SA failed due to an invalid certificate with high negative reputation then the nodes involved in the key negotiation exchanged their certificates. The node that was rejected by its peer due to the invalid certificate could complain to other DCAs by showing its altered certificate with any other signed proofs of previous behavior. The DCAs could again take appropriate actions.

A DCA could deny service to the nodes. However, due to the nature of a MANET environment, a node could not know the reason for the DCA's unavailability, e.g., the DCA might be rebooting or it might be compromised. Therefore, the only alternative solution for a node to obtain service was via other DCAs. If DoS occurred during the synchronization of updates with the SDCAs, then the node involved in a transaction received no acknowledgements from any of the SDCAs, as described in Section 4.6. The node could reinforce the distribution of its certificate updates by communicating its certificate information to other DCAs. In this way, the node ensured higher availability of its latest certificate during future SAs establishments.

4.11.2. TCA Security Analysis

A TCA could not generate a certificate for another node with false credentials because it was validated by the node that requested the service. If that node believed the

certificate was invalid then it simply requested another one from a collocated TCA. The TCA certificates expired within a shorter period compared to a DCA certificate and, thus, there was no vulnerability related to the certificate renewal and revocation functions of the KMS. As previously mentioned, if DoS occurred by a TCA then the node could always get a TCA certificate from any other physically collocated TCA.

In the case of collusion between a TCA and a generic node, an invalid TCA certificate could not be detected by DCAs unless out-of-band authentication with DCAs was carried out. However, possession of a TCA certificate restricted a node to SAs with short lifetimes and lower trust levels forcing it to register with a DCA and obtain a DCA certificate. Once the node authenticated with a DCA, the DCA could detect the invalid TCA certificate of the node and isolate it from the network through revocation.

4.11.3. Generic Node Security Analysis

During behavior grading, a generic node could only accuse the peers with which it had SAs. Moreover, it could only complain once for each of its TPs and, in effect, it isolated itself in the network (see Section 4.4). As a result, it was discouraged from maliciously complaining about its peers. Positive reputation in the KMS was not recorded unless both nodes reported their SAs. In this way, nodes were forced to be less selfish and were motivated to report their trust to increase their trustworthiness within the network.

A collusion of a number of nodes could earn them higher positive reputation by reporting that they trusted one another. However, the number of nodes in a collusion would be relative small compared to the average number of TPs of a node in a network and could not significantly benefit the particular nodes. In addition, other schemes, such as IDSs, would enable the detection of any other type of malicious behavior of the nodes in the collusion and enable the rest of the nodes to complain to the DCAs.

A collusion of malicious nodes could also lead to the reporting of a number of complaints for a particular node. However, the impact of this attack would depend on the number of nodes in that collusion that had SAs with that node. As previously mentioned nodes could only complain about their TPs. Moreover, this reporting would in effect

isolate the group of malicious nodes from the rest of the network without necessarily leading to the revocation of the particular node, assuming the node had an overall "good" behavior.

Summarizing, the KMS behaved like an IDS but at a higher level of abstraction through the use of network-wide SAs. The KMS could detect malicious behavior of the nodes and DCAs and respond appropriately avoiding the final state where the system could be rendered inoperable. The KMS functioned by propagating information over a number of nodes and DCAs through the use of non-reputable transactions. The behavior grading scheme allowed the KMS to dynamically adjust itself and reconfigure its policies based on the level of malicious behavior in the network. At low levels of malicious activity the period of routine revocation was long and the frequency of security alerts and revocation was low and vice versa.

4.12. Conclusions

The proposed KMS was created by taking into account the limitations of previous work done by other researchers and by incorporating the requirements derived from the investigation of the state of technology. The KMS overcame the limitations of previous systems by allowing minimal initial key set up. The only pre-configuration requirement for a CA was that a node possessed an RCA certificate.

The KMS increased availability in a number of ways. The system offered multiple DCAs to generate, store, and distribute certificates to the nodes. If all the DCAs were unavailable a node obtained a peer's key from any of the TPs of that node. This functionality was achieved by having each node store the certificates of the nodes that it trusted. Furthermore, the system decreased the frequency of certificate revocation by relaxing time constraints. Certificates were revoked whenever a DCA requested. In addition, a node was motivated to reissue its certificate to reestablish its status as a trustworthy node.

The KMS introduced security into the network by using immediate and routine revocation and security alerts by DCAs. Additional security was enabled through the use of behavior grading, which relaxed the need of relying on strict identity verification and

allowed nodes to judge other nodes based on their trustworthiness. The transactions of the system were recorded in a non-reputable manner and were verified by more than two nodes providing balance of powers among nodes and DCAs.

The KMS was flexible enough to accommodate new nodes when the DCAs were unavailable. New nodes that joined the network and were pre-configured with an RCA certificate, could temporarily establish trust with other nodes. If they did not possess a certificate they could obtain a TCA certificate from any of the nodes or TCAs that were physically collocated by first authenticating using out-of-band methods. As a result, they could temporarily be accepted in the network, until they could register at a DCA. Moreover, this KMS did not necessarily require out-of-band authentication with a DCA. New nodes joining the network could simply register via peer nodes with the DCA, if they were unable to authenticate with out-of-band methods.

Interoperability with IPsec was ensured by incorporating the requirements derived from the preliminary work. The DNS role would need to be augmented to perform the required DCA functions. This system could be applied to secure other functions of a network such as routing. Some routing mechanisms specify the need for such a KMS whereas others imply its existence. Chapter 5 describes the analysis and results of the KMS' evaluation.

Chapter 5

Analysis and Results

This chapter describes the analysis of the KMS and presents the results of that analysis. Section 5.1 describes the methodology used to analyze the KMS. Section 5.2 covers the Monte Carlo simulation analysis of the KMS. Section 5.3 covers the overhead of the KMS. Section 5.4 presents the impact of mobility on the KMS' functionalities. Section 5.5 describes the implementation of the KMS and its interoperation with IPsec. Section 5.6 discusses the integration of the KMS with threshold cryptography schemes.

5.1. Analysis Methodology

The analysis of the KMS was carried out in a series of steps with the objective of addressing the research questions listed in Section 1.3. First, a Monte Carlo simulation was utilized to investigate the effectiveness of the KMS's functionalities. The functionalities were analyzed by investigating connectivity in randomly generated static networks. In the second step, the overhead of the KMS was analyzed with regards to behavior grading, revocation, and certificate reissuance. The third step involved the simulation of the KMS with the network simulation tool NS2 [16] to observe the effectiveness of the KMS in a mobile ad hoc environment. Simulation in NS2 enabled the evaluation of the KMS's functionality by quantifying the evaluation metrics in terms of time. Finally, the KMS was implemented and integrated with the existing IPsec implementation.

5.2. Monte Carlo Simulation

A Monte Carlo analysis of the system was utilized to assess the impact of network connectivity in MANETs on the service availability of the system. A Monte Carlo Analysis of a system or network is one without a time axis, i.e., multiple static network topologies were generated and the connectivity was analyzed based on the various network parameters, such as radio range and node density [100]. The KMS functionalities investigated with regards to connectivity were certificate issuance and reissuance, certificate acquisition for SA establishment, and certificate revocation. An analysis and comparison of service availability of this KMS as compared to threshold cryptography schemes was also provided. The comparison was based on certificate issuance.

Certificate issuance and reissuance required nodes to directly communicate with any of the DCAs available in the network. Nodes issued their certificates once they entered the network. Nodes reissued their certificates during routine revocation or whenever they needed to reestablish their trustworthiness level, as described in Section 4.5. Certificate acquisition was carried out during SAs establishment. Nodes could obtain the certificate of a peer either from a DCA or from any of the TPs of that particular peer. The purpose of certificate acquisition was to obtain a peer's certificate with the latest behavior grading information to better judge the trustworthiness of that peer. It is important to clarify that certificate acquisition differed from certificate issuance and reissuance. Certificate acquisition was dependent on the probability that any of the TPs of a node or DCAs were reachable, whereas certificate issuance was strictly dependent on the DCAs' reachability.

The certificate revocation effectiveness was affected by the ability of one or more DCAs to reach the TPs of a node within the network in order to inform them that their peer had malicious intentions and its certificate had been revoked. Revocation was investigated from the perspective of informing the TPs of a node *once*, i.e., each TP received one RN. In general, it was expected that a node might not dissolve its SA with its peer until after receiving more RNs. An extended scenario that investigates the

propagation of RNs and the effectiveness of informing the TPs of a node more than once is presented in Section 5.4.3.2.

Similar to certificate revocation in the KMS, certificate issuance with threshold cryptography schemes depended on the ability of a node to reach a group of peer nodes. However in the case of threshold cryptography schemes, those peer nodes were CAs instead of TPs. A node had to reach a percentage of the CAs within the network in order to obtain partial certificates. It then combined the partial certificates and built its certificate.

The Monte Carlo simulation was carried out with locally developed C code with the aim of investigating these specific functionalities. In the KMS model, any communication limitations arising from the lower protocol layers were ignored. It was assumed that there was no contention and transmission of data was error-free. All nodes had the same radio range and all links were bidirectional. Two nodes could communicate with each other if their radio range was equal or greater than the distance between them.

Guichal and Toh evaluated centralized and distributed service location protocols for pervasive wireless networks [101]. Parts of their analysis could be applied to the certificate issuance and certificate acquisition analysis of the KMS. They demonstrated that service availability increased with an increasing number of servers for specific path lengths. Their results were based on an average node degree of connectivity of 3.2. However, those results did not suffice for the purpose of this research. The analysis of this research built on Guichal's and Toh's work by using a variety of node degrees in the network and demonstrating how service availability varied with varying degrees of connectivity. In addition, availability in this research was measured regardless of the of the path length. The variation of the path length to a particular destination was shown at different levels of connectivity in the network.

5.2.1. Performance Metrics

A number of metrics were used for each of the previously mentioned functionalities that were investigated. The performance metrics were defined as follows.

The metrics of interest for certificate (re)issuance and certificate acquisition were:

Average Shortest Path Length for the Network (ASPLN): The average shortest path between any node in a network and its nearest DCA or TP. If a node was in the range of a number of DCAs or TPs, then the closest one was recorded. ASPLN demonstrated the path length to obtain a service. The shorter the path length, the higher was the probability that the node could obtain service in a MANET environment. (The term "shortest" in the ASPLx metric signified that the shortest paths between all sources and destinations were derived using Dijkstra's algorithm [102][103].)

Availability: The percentage of nodes out of the total nodes in a network that could contact any DCA/TP. Availability complemented the ASPLN results as it accounted for a partitioned network environment, where the path length equaled infinity. Once the network was connected and availability was 100%, ASPLN was used instead to provide information regarding the path length to any DCA/TP.

The metrics of interest for revocation were as follows.

Average Shortest Path Length for Revocation (ASPLR): This metric depicted the average shortest path length between a DCA and a number of randomly selected TPs of a particular node whose certificate had been revoked. The lower the average path length value was, the higher the probability that those TPs would be made aware of the revoked certificate of their peer.

Percentage of Reachable Nodes/CAs (PRN/C): In the case of revocation, PRN/C indicated the percentage of nodes, out of a group of nodes, which were reachable by a single DCA during revocation. Like *availability*, it provided an insight on node reach ability when the network was partitioned and complemented the results of ASPLR.

Iso (DCA is Isolated): Iso accounted for partitioning for the case where a randomly selected DCA was isolated and could not inform the existing TPs of a revoked malicious node.

The metrics of interest for certificate issuance for threshold cryptography schemes were the same as the metrics of revocation. However the semantics of the metrics were different. *PRN/C* indicated the percentage of CAs that was reachable by a single node during certificate issuance. *Iso* accounted for partitioning for the case where a randomly selected node was isolated from the network and could not communicate with any CAs.

5.2.2. Simulation Parameters

The simulation parameters selected for the Monte Carlo simulation are shown in Table 5.1. A more detailed description for the selection of these parameters follows.

Table 5.1. Parameters for Monte Carlo Simulation Analysis

Radio range (meters)	100,150,200,250 300
Nodes	40, 80, 120
Fixed area	1000 x 1000 m^2
DCAs	10-20 %
TPs	10-20%
Distribution	Uniform
Number of network graphs	700

5.2.2.1. Radio Range, Node Density, and Area

The service availability of the KMS was dependent on the connectivity among nodes in a MANET. The network parameters that affected connectivity were node density, radio range, and the area over which the nodes were distributed. To investigate connectivity in highly partitioned as well as connected MANETs, Bettstetter's research [104] was used to aid in the selection of appropriate values for these parameters. His research gave some practical guidance in selecting the radio ranges at different node densities that could provide different levels of connectivity. Bettstetter investigated the characteristics of minimum node degree and k-connectivity of a wireless multi-hop network with uniformly distributed nodes. Equation 5.1 [104] was used to calculate the approximate radio range within a network at which no node is isolated. The underlying assumptions of this equation are the same as the assumptions of the Monte Carlo simulation, i.e., uniform distribution, omnidirectional antennas, and same transmission range for all nodes

$$P(d_{\min} \geq n_o) = \left(1 - \sum_{N=0}^{n_o-1} \frac{(\pi \rho r^2)}{N!} e^{-\rho \pi r^2}\right)^n \qquad \text{EQN. 5.1}$$

P is the probability that each node in network has at least n_o neighbors. That is, the network has a node degree d_{min}, where $d_{min} \geq n_0$. The number of nodes in the network is *n*, and they have a radio range *r* (meters). The node density is ρ (nodes/meter^2). Figure 5.1 was generated from Equation 5.1. It shows the node degree variation based on the number of nodes and the radio range. The line "$d_{min} \geq$1- Pb 100%" indicates the radio range and node density with which the minimum node degree was greater than or equal to one with probability one and represents the boundary between a *connected* and a partitioned network. By definition, a *connected* network is represented by a connected graph, G, if it has a (u, v)-path whenever (u,v) $\in$V(G). The area below that line designated the radio range and node density combinations with which the network had greater than one partition. The area above the line indicated the radio range and node density combinations with which the network was connected and no node was isolated.

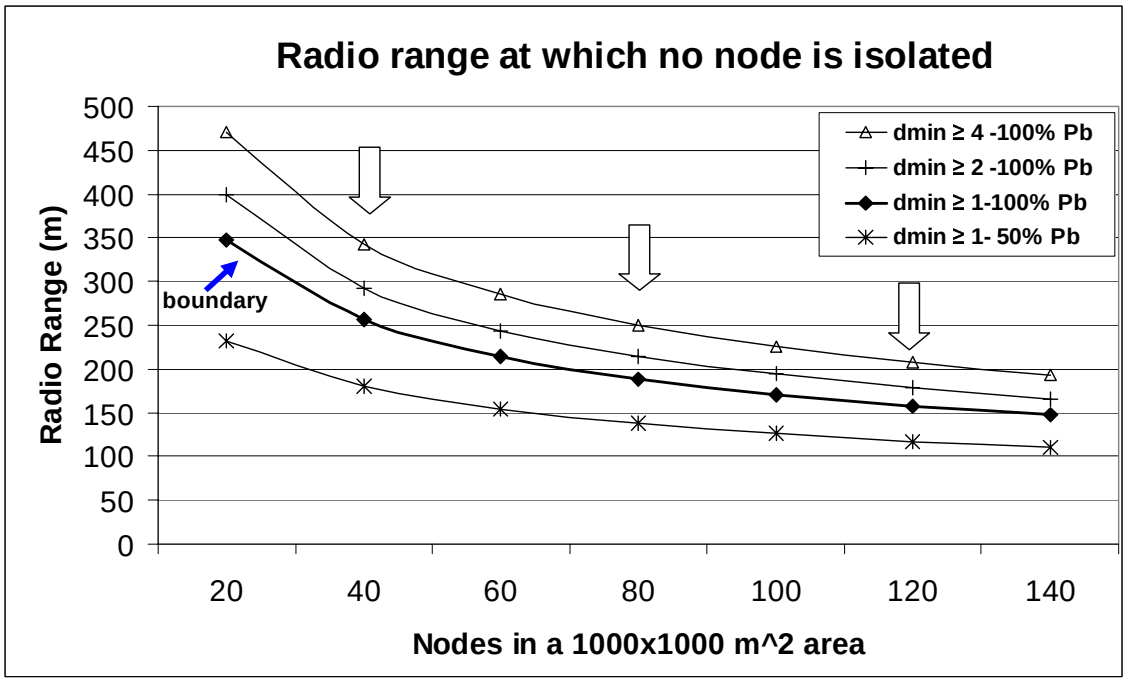


Figure 5.1. Connectivity based on radio range and node density.

Based on Figure 5.1, the number of nodes utilized in the simulation was 40, 80, and 120. The area was fixed to 1000x1000 m^2. According to Figure 5.1, radio ranges between 100 and 300 meters at the selected node densities facilitated data collection for both *partitioned* and *connected* networks.

In order to verify the validity of the selected radio ranges and node densities, 700 network graphs were generated for each radio range and node density combination and the number of partitions and node degree were recorded. (Seven hundred network graphs provided a 5% or smaller deviation from the mean with 95% confidence.) Figure 5.2 and Figure 5.3 demonstrate the number of partitions based on the radio range and node density and prove the validity of the selected parameters. Table 5.2 shows the average node degree of the generated network graphs.

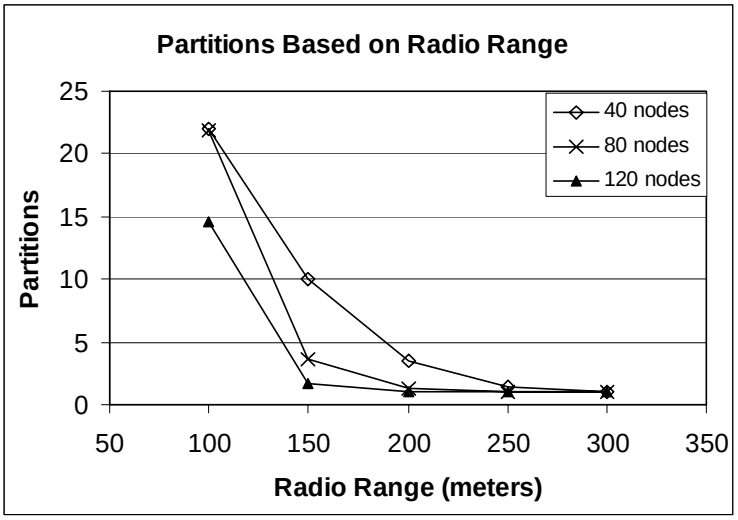


Figure 5.2. Partitions based on the radio range used.

Table 5.2. Average Node Degree for Scenarios Used

Nodes	**Radio Range (meters)**				
	100	**150**	**200**	**250**	**300**
40	1	2.5	4	6.1	8
80	2	5	8	12.3	16
120	3.5	7.3	12	18.6	25

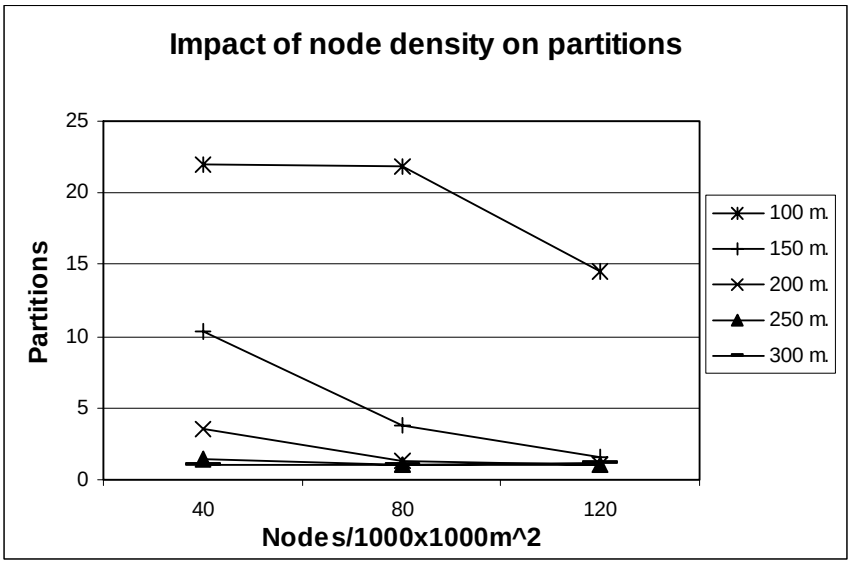


Figure 5.3. Partitions used in the network.

5.2.2.2. DCAs and TPs

The percentage of nodes selected to be DCAs were 10% and 20% of the total number of nodes, as was used in previous research [101][69]. In addition, a higher percentage of DCAs would decrease security of the network since it would provide a higher number of points of attacks and increase the communication overhead between the DCAs.

Similar to DCAs, the percentage of TPs of a particular node was considered to be 10% and 20% of the total number of nodes, as in Capkun, Hubaux, and Buttyan [63]. Those authors argued that it was not necessary for all nodes to trust each other to secure routing. An incomplete set of SAs in the network was sufficient to enable the establishment of secure routes. In a similar manner, these values were selected in this research in order to investigate the service availability effectiveness of the KMS with a small number of TPs (SAs).

Thus, in a network with 20% DCAs and 20% TPs, a node's peers could potentially acquire its certificate from a combined 40% of the nodes in the network, but could reissue its certificate only from the 20% DCAs.

5.2.2.3. Node Distribution

The nodes were uniformly distributed in a fixed area. The Mersenne Twister random number generator [105] was used to generate the random locations of all the nodes in each network graph. The Mersenne Twister pseudo-random number generator was selected because Pawlikowski, Jeong and Lee credit it as having a good virtual randomness in their work on credibility of simulation studies [106]. Seven hundred different network graphs were generated. This number of network graphs ensured a 5% or smaller deviation from the mean with 95% confidence interval.

5.2.3. Simulation Analysis

The following subsections present the analysis of the results of the Monte Carlo simulation. The KMS functionalities analyzed are certificate issuance and acquisition, certificate revocation, and certificate issuance in threshold cryptography schemes.

5.2.3.1. Certificate Issuance and Acquisition

Figure 5.4 and Figure 5.5 depict the availability of the KMS based on the number of nodes and radio range and, more specifically, its effectiveness in distributing certificates in a partitioned environment. (The corresponding partitions based on radio range are displayed in Figure 5.2.) As the number of DCAs increased from the centralized case (1 DCA) to 10% DCAs (of the total number of nodes), they distributed or issued certificates to nodes more effectively. In addition, the existence of 10% of TPs for a node (20% DCAs and TPs) could significantly increase SA establishment as a node could more easily obtain other nodes' certificates. If the node density was doubled from 40 to 80 nodes, as shown in Figure 5.4 and Figure 5.5, the availability converged to 100% at a radio range of 200 meters instead of 300 meters.

By fixing the radio range at 100 meters, Figure 5.6 illustrates availability in a highly partitioned network. The effectiveness of *certificate issuance* can be observed by the increase in availability as the DCAs increase from the centralized case on the left of the graph to a maximum of 20% DCAs. In the case of *certificate acquisition*, the whole

range (1 DCA (centralized case) to 40%) could be considered, since nodes could either obtain certificates from a maximum of 20% DCAs and 20% TPs. The relative increase in availability, for the 10 to 40% range of DCAs and TPs, was higher for the 40-node network (25 to 67%) as compared to the relative increase of the 80-node (51 to 85%) and 120-node (79 to 95%) networks, because the 40-node network was more partitioned (see Table 5.2).

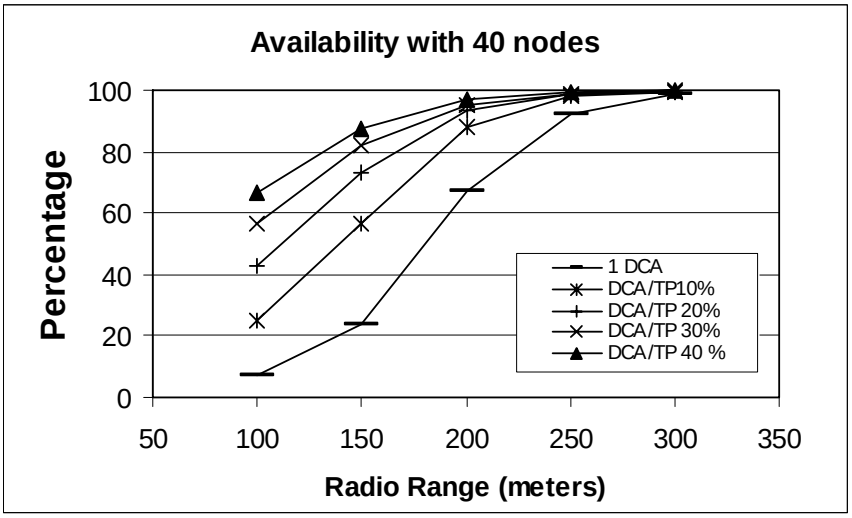


Figure 5.4. Number of nodes with service in the network.

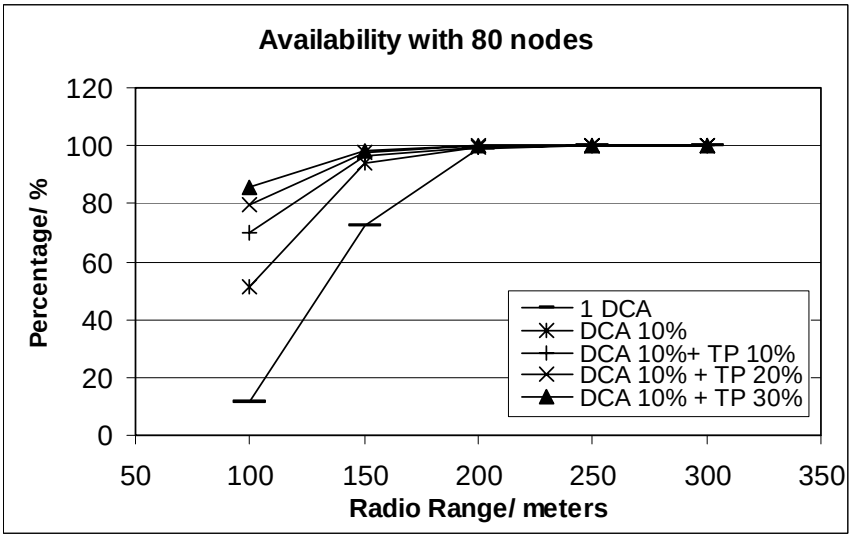


Figure 5.5. Number of nodes with service in the network.

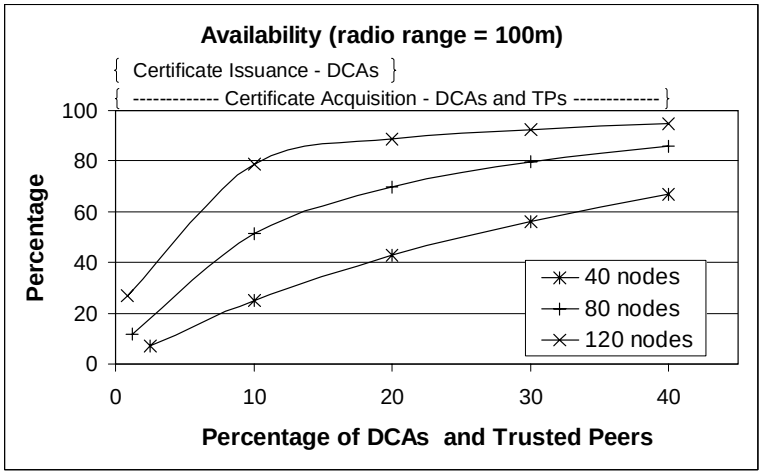


Figure 5.6. Number of nodes with service at a radio range of 100 meters.

When using a higher radio range, such as 250 m (see Figure 5.4), the network became connected and the entire network was encompassed in a single partition. At this radio range, the *Availability* increased to 100% and ASPLN was used to determine the effectiveness in the service availability of the KMS by illustrating the shortest path to a server. Figure 5.7 and Figure 5.8 demonstrate the variation in the path length depending on the number of DCAs/TPs, and radio range.

It is important to notice the "hump" in the path length in Figure 5.7 and Figure 5.8 for the 40-node network, which was caused by the change in connectivity. This variation in path length was non-intuitive because ASPLN did not decrease as the radio range increased, but fluctuated due to the existence of network partitions. The reason for this variation was that, initially, the short radio range partitioned the network into a number of small size partitions (see Figure 5.2). Therefore, if a DCA was available in a small partition, the path length to that DCA was short, since it was constrained by the size of the partition. As the connectivity increased, the partition size of the existing partitions increased since nodes joined to form bigger clusters. If a DCA was available in that cluster, then the path length to that DCA was on average longer. As the radio range and the degree of connectivity increased even more, the ASPLN reached a maximum value (e.g., 190 m for 40 nodes for Figure 5.7), which to some level represented the connectivity at which maximum cluster sizes existed. If nodes used an even higher radio range, clusters started to merge, enabling the nodes to find shorter paths to the DCAs and

TPs. As a result, the ASPLN decreased exponentially. Thus, as the network connectivity increased from a highly partitioned to a connected network, the ASPLN increased to a maximum value and then decreased exponentially.

Another notable observation was that the ASPLN variation for the 80-node network was not similar to the 120-node network (see Figure 5.7). The 120-node network had a higher degree of connectivity and was at different phases of the path length variation. More specifically, the trend of the path length for the 120-node network was similar to the final stage of the 80-node network (>200 m), where ASPLN decreased exponentially.

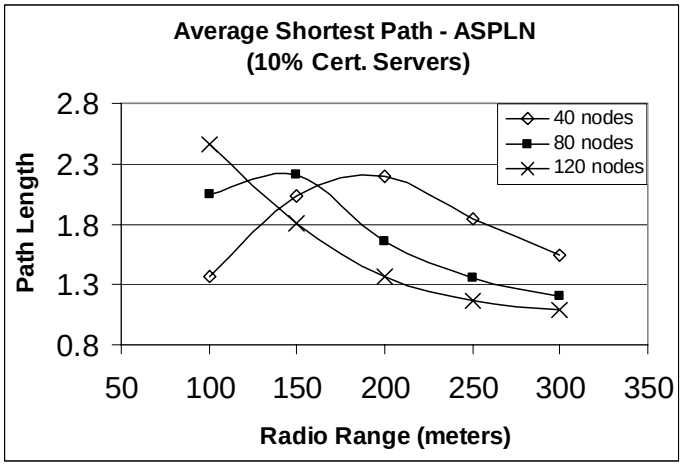


Figure 5.7. Shortest path to a server with 10% certificate servers.

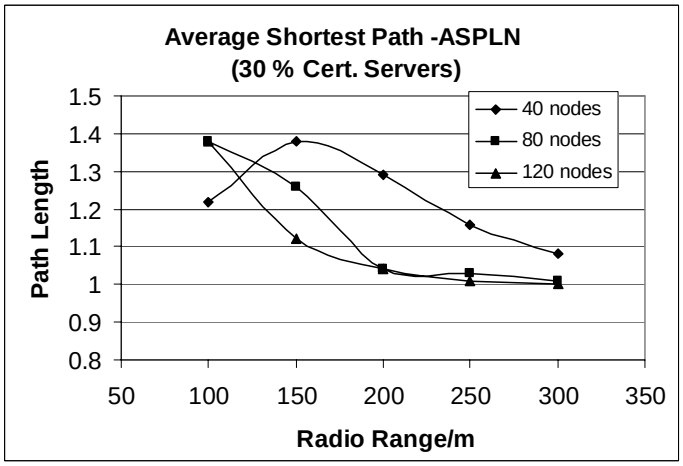


Figure 5.8. Shortest path to a server with 30% certificate servers.

As the number of DCAs and TPs increased, the path length decreased to a value of one. Overall, the path length varied between 1 and 3 hops, which was relatively low compared to a centralized system (see Figure 5.9 – "1 DCA"). In addition, within a 3-hop path length it was more likely that a node obtained service from a DCA or TP in a MANET environment. Figure 5.10 and Figure 5.11 present the variation of ASPLN for the 40-node and 120-node networks, respectively, with different radio ranges. These graphs demonstrate that as the number of DCAs/TPs increased the ASPLN graph shifted downwards towards the path length value of one.

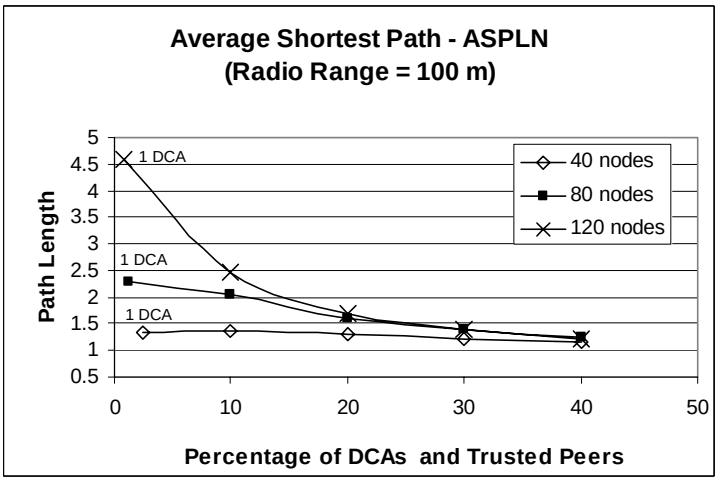


Figure 5.9. Shortest path with radio range of 100 meters.

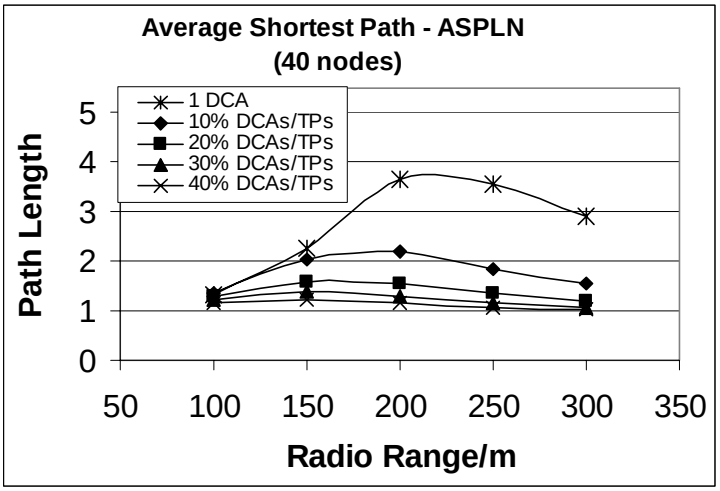


Figure 5.10. Shortest path to a server (40 nodes).

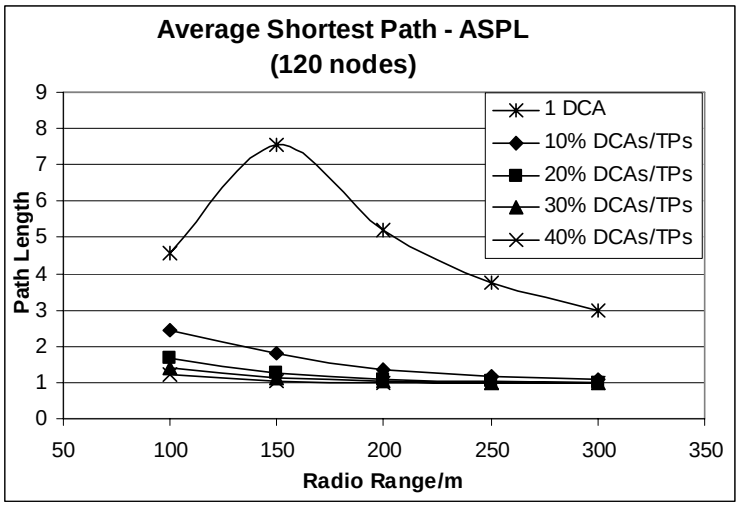


Figure 5.11. Shortest path to a server (120 nodes).

5.2.3.2. Certificate Revocation

Revocation dealt with the ability of a single DCA to contact a percentage of nodes, representing the TPs of a malicious node, to inform them of the revocation of their peer's certificate. Figure 5.12 demonstrates the PRN/C variation based on the radio range and node density. The percentage of reachable nodes was low for the highly partitioned network. The data for this graph were collected under the assumption that a DCA was not isolated, which is depicted in Figure 5.13. For example, in a network of 80 nodes and a radio range of 100 meters there was a 10% probability that the DCA would be isolated in the network. This value implied that there was a 90% probability (100%-10%) that the DCA would be able to communicate with 10% of the TPs of a malicious node, as shown in Figure 5.12.

Figure 5.14 shows the ASPLR from a single DCA to the TPs of a malicious node. The variation in the average path length was dependent on the cluster sizes and change in connectivity, as previously described. Even though connectivity was higher with 80 and 120 nodes, the average broadcast path length was also higher, which implied that it might be less likely to connect to all nodes from the first transmission. As connectivity increased to a radio range of 300 meters, the average broadcast path length for all node

densities converged to 3 hops, increasing the probability that a DCA would be able to contact all of the nodes involved.

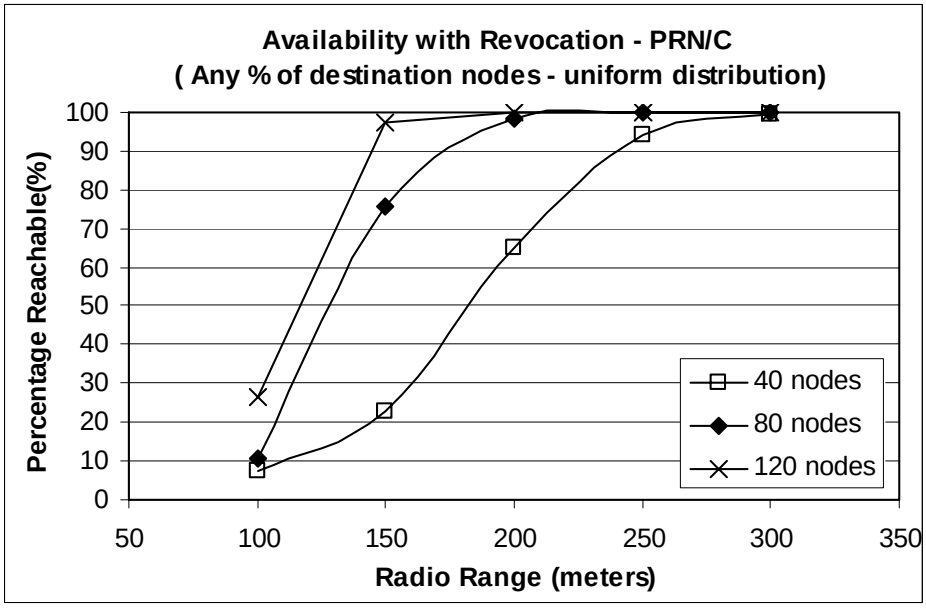


Figure 5.12. Percentage of nodes reached by one DCA.

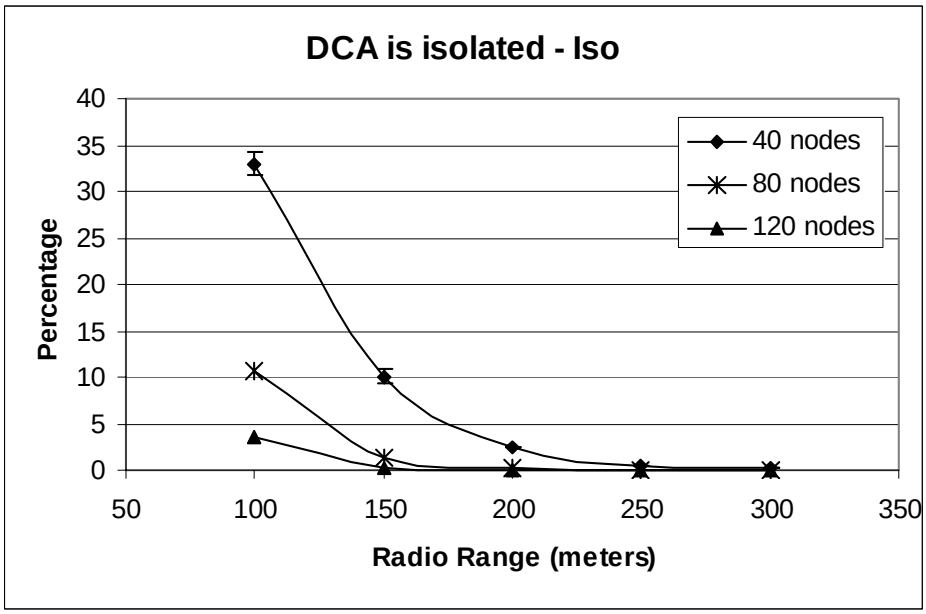


Figure 5.13. Probability that a DCA is isolated.

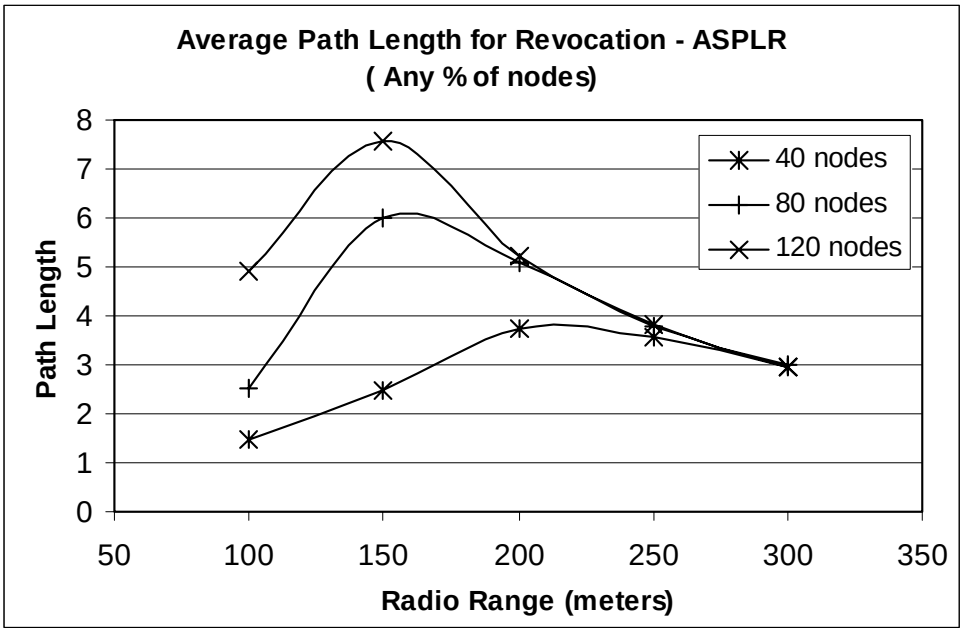


Figure 5.14. Path length from a DCA to reach a percentage of TPs.

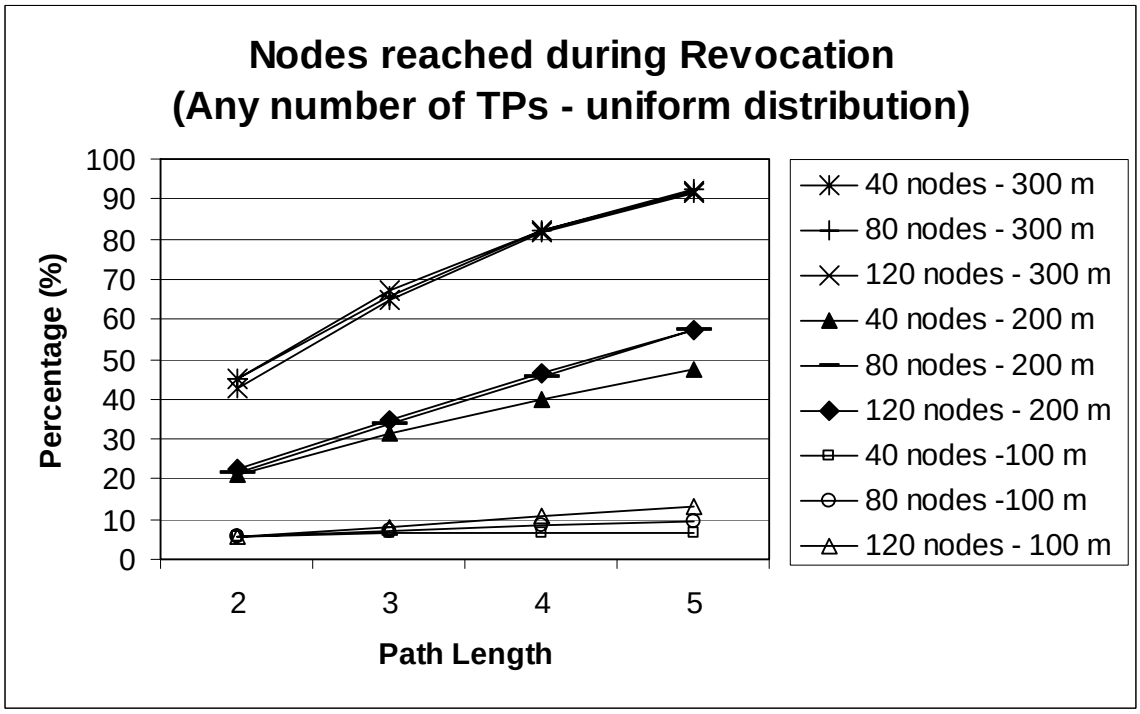


Figure 5.15. Percentage of nodes reached within specific path lengths.

Figure 5.15 provides an alternative picture with regards to the ability to connect to a group of TPs. It demonstrates the exact percentage of nodes that could be reached

during a revocation process based on the number of nodes and radio range within fixed path lengths. As can be observed, the radio range rather than the node density controlled the percentage of reachable nodes.

Even though these results illustrated the impact of connectivity on revocation, they did not capture the effectiveness of revocation at the particular instant that revocation was triggered. It would be more likely that at the particular instant that a node triggered revocation by complaining about its peer, the DCA would be in close proximity to both the malicious node and the peer that complained, as described in the negative reputation scenario. Thus, the DCA would be able to inform all the immediately impacted TPs of the malicious node in that partition. The TPs of the revoked node would then be aware of the malicious activity of their peer.

5.2.3.3. Threshold Cryptography Schemes

Threshold cryptography schemes lead to less accessibility to CAs during certificate issuance especially in rapidly-deployed partitioned environments. In these schemes, a single node had to contact a percentage of nodes acting as CAs and obtain partial certificates, before it could build its certificate. Figure 5.16 demonstrates the ability of a single node to reach a percentage of CAs, when that node was not isolated in the network (as shown in Figure 5.17). For example, for a 40-node network with 4 CAs (10% CAs) and a radio range of 150 meters, a node could only reach 22% of the number of CAs (see Figure 5.16) and it would be isolated 10% of the time (see Figure 5.17). If the node had to access 80% of the DCAs in the network to obtain partial certificates, then it would have to retransmit a number of times to be able to access a sufficient number of CAs. In our KMS, access to 22% DCAs guaranteed certificate issuance from the first connection attempt to a DCA.

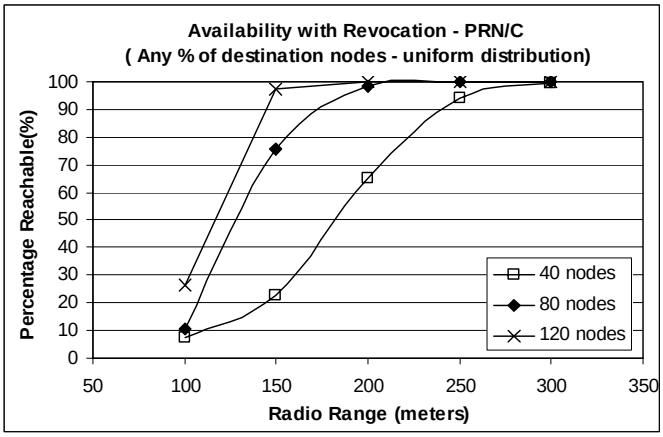


Figure 5.16. Percentage of CAs reached by one node.

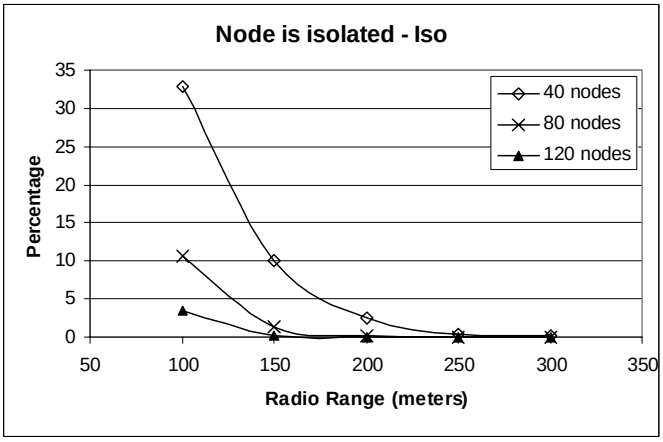


Figure 5.17. Probability that a node is isolated

5.2.4. Summary

The existence of DCAs increased the probability of certificate (re)issuance. In a partitioned environment with 100 meters radio range, *availability* with 10% of DCAs increased between 18% (40 nodes) to 51% (120 nodes) as compared to the centralized case. The higher the connectivity, the higher was the percentage increase in *availability* with a fixed number of DCAs in the network. In addition, nodes could more easily acquire the certificate of their peers during an SA establishment. In a partitioned network environment, with 100 meters radio range, *availability* with 10% of DCAs and 10% TPs increased between 36% (40 nodes) to 61% (120 nodes), as compared to the centralized case. In both partitioned and connected networks, the average path length between the

nodes and the server was less than 3 hops, whereas in the centralized case the path length was less than 8 hops.

This simulation analysis has also quantified the impact of network partitioning on revocation and, more specifically, the ability of a single DCA to reach a node's TPs. Certificate revocation was particularly challenging in a partitioned network because the DCA was isolated from the network for 4% (120 nodes) to 33% (40 nodes) of the time. Furthermore, when the DCA was not isolated it could only reach a small percentage of the TPs of a malicious node (8% for 40 nodes to 28% for 120 nodes). In addition, when the network was more connected (150 meters radio range) the average path length to the TPs was up to 8 hops, which could diminish the probability of successful communications. As the connectivity increased the average path to all TPs converged to 3 hops offering higher guarantees for revocation. The results from this analysis signified the importance of using security alerts as a means of increasing the security of the system.

In threshold cryptography schemes a node could only reach a small percentage of DCAs in a partitioned network. This limitation implied that multiple retries had to be sent to be able to reissue a certificate. On the contrary, this KMS provided higher availability for certificate issuance.

5.3. Overhead of the KMS

This section analyses the overhead of the KMS with regards to behavior grading, revocation, and reissuance. A discussion of certificate and RN sizes is also provided.

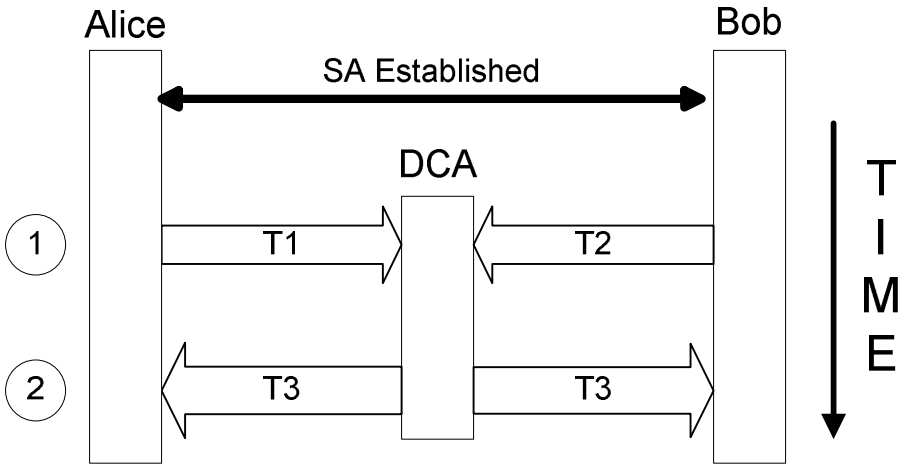
5.3.1. Behavior Grading

The overhead of the behavior grading mechanism was categorized according to its functionalities, positive reputation recording (as shown in Figure 5.18) and negative reputation recording (as shown in Figure 5.19). In the positive reputation overhead analysis, the worst case that would require the most overhead is considered. In this case, the KMS recorded all trusted pairs in a mesh connection, which is shown in Equation 5.2.

N represents the number of nodes in the network. The process involved two steps as shown in Figure 5.18: (1) nodes informed the DCA, and (2) the DCA confirmed the information and replied to the nodes. The summation of each step simplified to Equation 5.3.

$$SAs\_mesh, \omega = \frac{N(N-1)}{2} \quad \text{EQN. 5.2}$$

$$\begin{aligned} Load\_mesh &= Registration + DCA\_reply \\ &= (2\omega * |T_1| / |T_2|) + (2\omega * |T_3|) \\ &= (2\omega(|T_1| + |T_3|)) \\ &= (N(N-1)(|T_1| + |T_3|)) \end{aligned} \quad \text{EQN. 5.3}$$



$T1=\{Trust\ Bob, Cert_B\}k_A$
$T2=\{Trust\ Alice, Cert_A\}k_B$
$T3=\{Trust\ Bob\}pk_{DCA1}+\{Trust\ Alice\}pk_{DCA1}$ and/or $Cert_A$ and/or $Cert_B$

Figure 5.18. Positive reputation.

Negative reputation overhead was investigated from the perspective of a single node. We assumed the worst case where a node had SAs with the rest of the network and complained about all of them. The process involved two steps, as shown in Figure 5.19: (1) reporting the malicious node, and (2) obtaining an acknowledgement with updated certificates from a DCA. The summation of each step simplified to Equation 5.4.

$$\text{Report\_DCA} = ((N-1)|C_1|)$$
$$\text{DCA\_response} = ((N-1)|C_1|)+(2|C_2|(N-1))$$
$$\text{Load\_node} = \text{Report\_DCA} + \text{DCA\_response}$$
$$= ((2|C_1|+|C_2|)(N-1)) \qquad \text{EQN. 5.4}$$

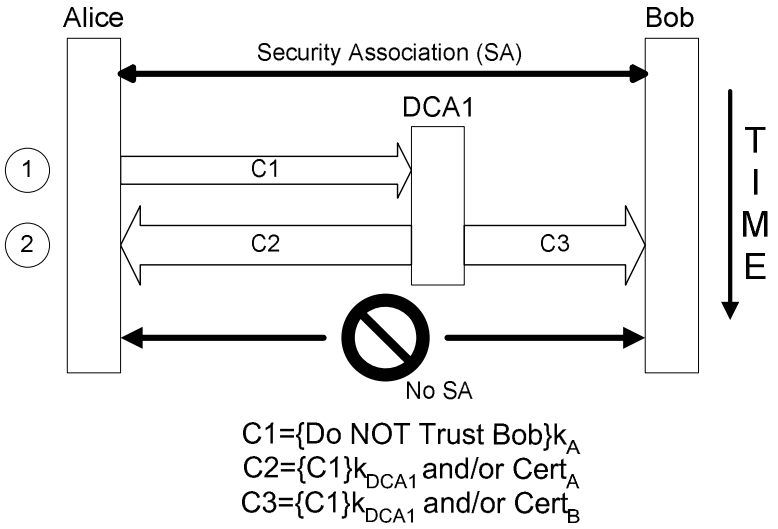


Figure 5.19. Negative reputation

Nodes elected whether they wanted to receive their updated certificates with every SA established or with every complaint. These options were also set by the security policies of the KMS, and were controlled with parameters T3 and C2 in Equations 5.3 and 5.4, as shown in Figure 5.18 and Figure 5.19, respectively. Thus, these parameters introduced flexibility in the system to dynamically adjust its overhead requirements based on the overhead constraints within the network. Stricter security policies imposed by the DCAs/nodes resulted in more frequent certificate updates requested by or sent to nodes, while more relaxed security policies implied less impact on the network for both processing and network overhead.

The transactions among TPs could be carried out over existing secure tunnels, such as IPsec tunnels. An analysis of the IPsec overhead is presented in our previous work [44].

5.3.2. Revocation

Revocation overhead could be directly mapped onto the network. The overhead of the extended scenario can be separated into three parts as shown in Figure 5.20: (1) DCA1 informs the node's TPs (i.e., A and B), (2) DCA1 informs a subset of other DCAs, the SDCAs, (e.g., DCA2 and DCA3), and (3) the SDCAs inform the node's TPs. Even though the second part of revocation contributes to synchronization among a subset of DCAs, information synchronization among DCAs is beyond the scope of this research. An examination of synchronization protocols for mobile devices has been offered in [92].

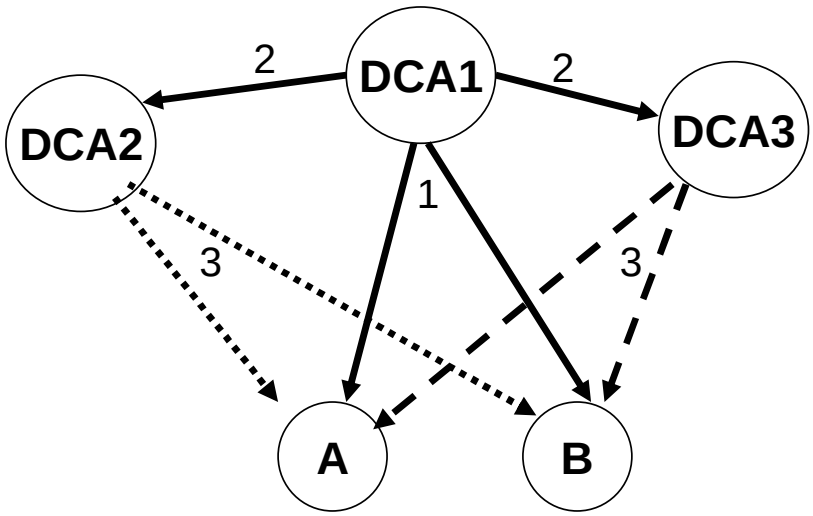


Figure 5.20. Extended revocation scheme.

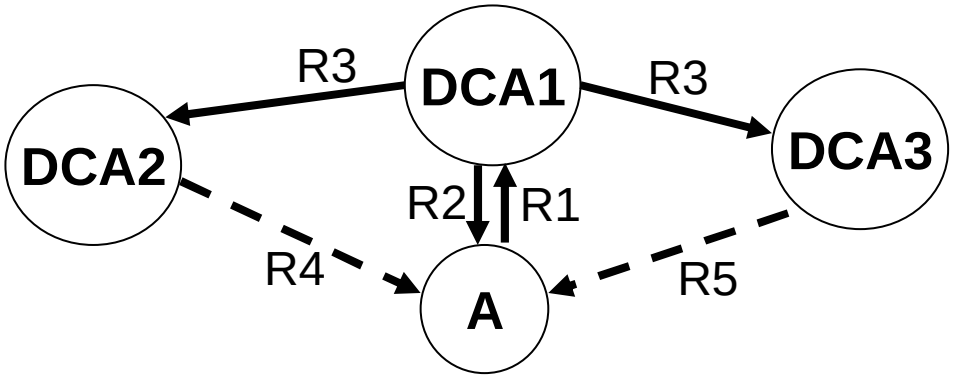
Equation 5.5 summarizes the overhead associated with this revocation procedure.

$$\begin{aligned} \text{Revocation} &= Initiating\_DCA + SDCA\_synch + SDCAs\_overhead \\ \text{Revocation} &= (|RN| * TPs) + (|RN| * SDCAs) + (SDCAs * |RN| * TPs) \\ \text{Revocation} &= |RN|(TPs + SDCAs + SDCAs * TPs) \end{aligned} \qquad \text{EQN. 5.5}$$

5.3.3. Reissuance

Figure 5.21 shows the overhead of the extended reissuance scenario, that similar to revocation, involves more than one DCA. The scenario is extended because it takes

into account the malicious activity detection aspects of the KMS, as previously described. The reissuance process takes place is three steps: (1) node A sends its information to DCA1 and obtains its certificate, (2) DCA1 communicates the information to a subset of DCAs (the SDCAs), and (3) DCA2 and DCA3 send an acknowledgement to node A. Similar to revocation, this extended scenario accounts for partial synchronization.



$R1=\{Request\}pk_{A}$
$R2=\{CertA\}pk_{DCA1}$
$R3=\{|R1| \text{ and/or } |R2|\}pk_{DCA1}$
$R4=\{|R3|\}pk_{DCA2}$
$R5=\{|R3|\}pk_{DCA3}$

Figure 5.21. Reissuance and partial synchronization transactions.

The overhead associated with the extended scenario is shown in Equation 5.6. In messages R4 and R5 a cryptographic hash of R3 could be used instead of the entire message in order to lower the size of the messages.

$$\begin{aligned} \text{Reissuance} &= Initial\_DCA\_Commun. + SDCA\_synch. + SDCAs\_confirmation \\ &= (|R1| + |R2|) + (Percent\_DCA * |R3|) + (Percent\_DCA * |R4|) \end{aligned} \quad \text{EQN. 5.6}$$

5.3.4. Certificate and Revocation Notice (RN) Size

The RN sent to nodes was based on the X.509 v3 [57][58][59]. Table 5.3 shows an example of the field sizes of the RN. The total size of the RN would be 90 bytes.

Table 5.3. Revocation Notice (RN) Size

Certificate Fields	**Size/bytes**
1) Issuer name	40
2) Current Date	15
3) Revoked certificate (e.g., serial number)	15
6) Signature algorithm identifier e.g., MD5	4
7) Signature	16 (MD5 - 128 bits)
Sum	**90**

Table 5.4. Certificate Size

Certificate Fields	**Size/bytes**
X.509 Certificate	
1) Version	1
2) Certificate serial number	1
3) Issuer name	40
4) Period of validity	15
5) Subject name	40
6) Signature algorithm identifier e.g., MD5	4
7) Signature	16 (MD5 - 128 bits)
8) Public key	128 (RSA -1024 bits)
Extensions for KMS	
1) Certificate possession	2
2) Authentication method	1
3) TCA certificate validity period	2
4) KMS role	1
5) Behavior grading	3
Sum	**304**

The DCA certificate format was similar to the X.509 v3 [57][58][59] with some extensions as shown in Table 5.4. Examples of some field sizes that could be utilized in

a MANET are shown in the second column of the table. The field sizes could vary based on the requirements of the network environment in which the KMS would be deployed. The KMS extensions did not significantly increase the size of the certificate. A certificate would be approximately 300 bytes. Based on the example of the DCA certificate size, a TCA certificate issued to new nodes would be approximately 600 bytes as it would include the DCA certificate of the certificate issuer in addition to the credentials of the new node.

5.4. Mobility Simulation

The purpose of this simulation was to investigate the impact of mobility on the service availability of the KMS. The simulation was carried out using the NS2 [16]. Similar to the Monte Carlo Simulation, the functionalities of the KMS analyzed were certificate issuance, certificate requisition and certificate revocation. A comparison of this KMS with certificate issuance in threshold cryptography schemes was also provided.

As discussed in Section 5.2, this analysis investigated an extended revocation scenario. In the Monte Carlo analysis, revocation was investigated with regards to the ability of a single DCA to reach the TPs of a malicious node and inform them *once*, i.e., each TP received one RN. It was expected that a node might not dissolve its SA with its peer until after receiving more RNs. The extended scenario of the Mobility Simulation investigated the propagation of RNs and the effectiveness of informing the TPs of a node more than once.

The mobility simulation was carried out in NS2 by integrating code developed in C++ and TCL with the existing NS2 modules. The code aimed at reproducing the behavior of the KMS to investigate the different functionalities. The data collected were aggregated and averaged by a series of Perl scripts. The NS2 simulation evaluated a wireless ad hoc environment and, thus, took into account communication limitations arising from the lower protocol layers, such as contention. All nodes had the same radio range and two nodes could communicate with each other if their radio range was equal or greater than the distance between them.

5.4.1. Performance Metrics

Similar to the Monte Carlo analysis, a number of metrics were used to investigate certificate issuance, certificate acquisition and revocation. The performance metrics were defined as follows.

The metrics of interest for certificate (re)issuance and certificate acquisition were as follows.

Success Ratio: The average number of successful attempts out of the total number of attempts to communicate with a DCA/TP. The success ratio was independent of the frequency of the attempts and could provide an indication of the average time to reach a server by using Equation 5.7

$$Average\ Time\_DCA/TP = \left(\frac{Period\_between\_Attempts}{Success\_Ratio} \right) \qquad \text{EQN. 5.7}$$

Average Period Between Non-Consecutive Successes (APBNCS):. This metric was utilized to take into account network partitions. In a highly partitioned network environment there would be a burst of successful consecutive attempts when the node was located in a partition that contained one or more DCA/TPs. However, when the node moved away from that partition, there was a burst of failed attempts until the node was able to join another partition and communicate with other DCA/TPs. The APBNCS metric disregarded consecutive successful attempts and measured the period between bursts of successful attempts. Thus, it more accurately reflected the impact of partitioning on service availability.

The metric of interest for revocation was as follows.

Average Period for Revocation (APR): This metric indicated the average period taken to execute the revocation procedure. As previously described, a DCA would initiate revocation after receiving a negative reputation report. The DCA would contact the TPs of a node and a number of its peer DCAs, the SDCAs. Those SDCAs would, in turn, inform the TPs of that node. The APR metric recorded the time take for this procedure to take place. For example, in Figure 5.22, DCA1 would initiate revocation and contact

node A, node B, DCA2 and DCA3. Node A and B are the TPs of Node X (not shown). DCA2 and DCA3 would then contact node A and B. In this particular example DCA1 informed two SDCAs and nodes A and B were informed three times. The number of SDCAs to revoke and inform the TPs could vary based on the policies of the KMS. In addition, Nodes A and B could elect to distrust Node X after receiving different numbers of RNs based on their individual security policies. This analysis investigated the impact on APR when varying both the KMS and individual security policies (i.e., utilized different number of SDCAs and RNs).

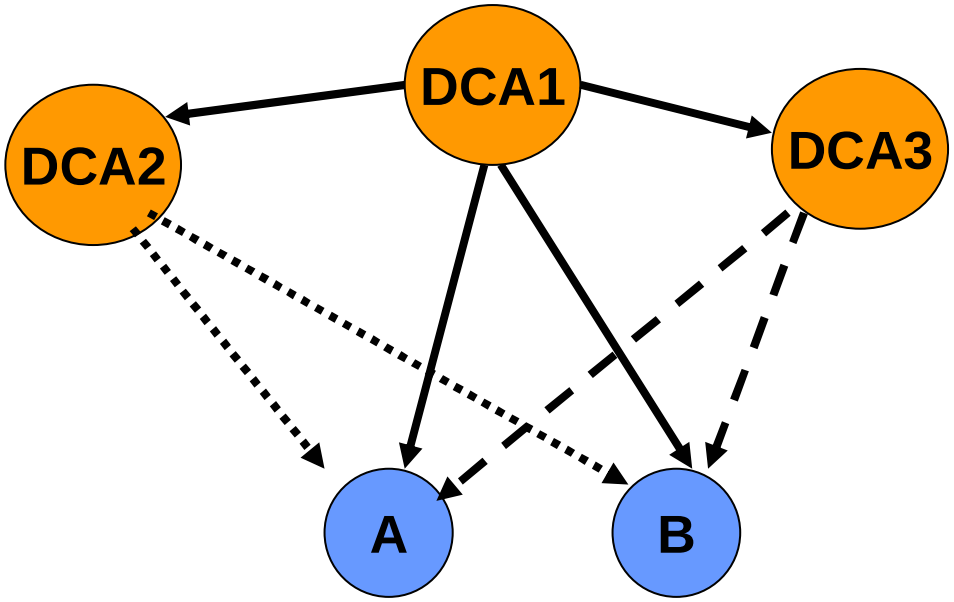


Figure 5.22. Revocation procedure.

The metric of interest for certificate issuance using threshold cryptography was the *Average Period to Reach CAs (APRC)*. This metric indicated the total average period required to reach a group of CAs.

5.4.2. Simulation Parameters

The simulation parameters and factors selected for the NS2 simulation are shown in Table 5.5. A more detailed description for the selection of these parameters follows.

Table 5.5. Parameters and Factors for NS2 Simulation Analysis

Radio range (meter)	100, 150, 200, 300
Nodes	40, 80, 120
Fixed area	1000 x 1000 m^2
DCAs	10-20 %
TPs	10-20%
Communication interval (second)	15
Mobility model	Random Waypoint Model
Node speed (m/s)	3, 5, 10, 15
Simulation time (second)	20000
Warm-up period (second)	1000
Repetitions	20
Routing protocols	AODV, OLSR

5.4.2.1. Radio Range, Node Density, and Area

The service availability of the KMS was dependent on the connectivity among nodes in a MANET. The network parameters that affected connectivity were the number of nodes, the radio range, the area over which the nodes were distributed, and the mobility model utilized. The BonnMotion tool [107] was utilized to select the parameters that would allow investigation of connectivity in highly partitioned as well as connected networks. The BonnMotion tool could generate mobility scenarios and statistically analyze them to provide information such as the average degree of connectivity and the average number of partitions. To verify the network connectivity, 100 mobility scenarios were generated for each radio range and node density combination and each scenario was statistically analyzed. The scenarios were based on the random waypoint mobility model [108].

Figure 5.23 and Figure 5.24 demonstrate the number of partitions based on the radio range and node density and prove the validity of the selected parameters. Table 5.6

shows the average node degree of the generated network graphs. It is important to note that the connectivity was approximately the same with different node speeds.

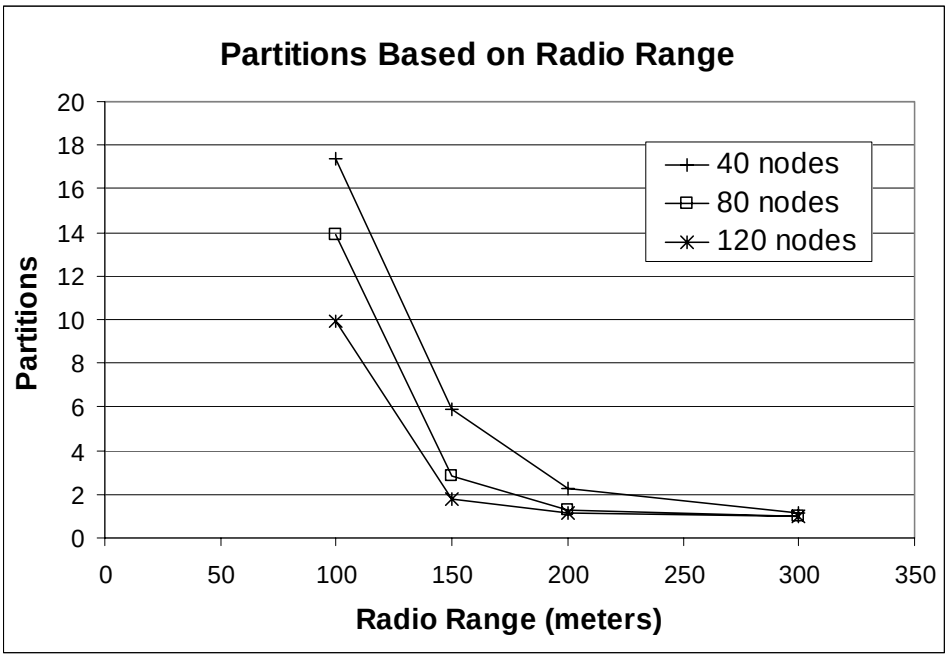


Figure 5.23. Partitions based on the radio range used.

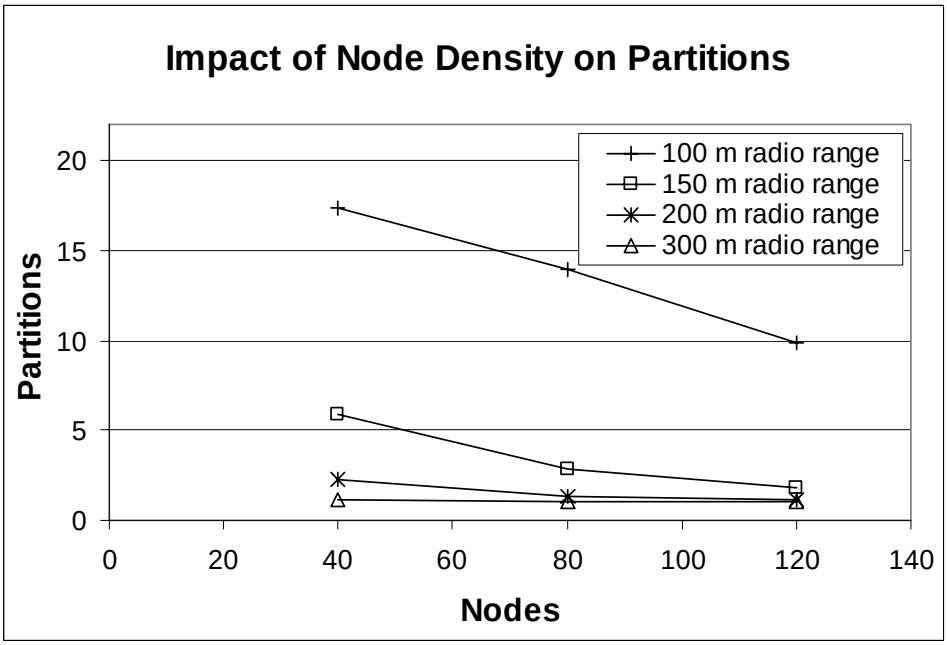


Figure 5.24. Partitions used in the network.

Table 5.6. Average Node Degree for Scenarios Used

Nodes	**Radio Range (meters)**			
	100	**150**	**200**	**300**
40	1.6	3.55	6	12.3
80	3.3	7.45	12.3	25
120	4.95	11.2	18.5	38

5.4.2.2. DCAs and TPs

The percentage of nodes selected to be DCAs was 10% and 20% as was used in the Monte Carlo simulation analysis. Similarly, between 10% and 20% of the total number of nodes were selected to be TPs.

5.4.2.3. Radio Propagation Model

Radio propagation used the two-ray ground model [16][109]. Like the free-space model, the two-ray ground model predicts the received power as a deterministic function of distance. It represents the communication range as an ideal circle meaning that two nodes are connected if the distance between them is greater or equal to their radio range. Unlike the free-space model, the two-ray ground considers both the direct path and the ground reflection path and does not assume the ideal propagation condition of the free-space model, i.e., there is a seldom a single clear line-of-sight path between two communicating entities. Results of this research could be extended to consider other radio propagation models.

5.4.2.4. Mobility Model

The random waypoint model utilized in this research was the only one offered in NS2 and is one of the most commonly used mobility models for simulations [110][111][112][113][114]. The random waypoint trace files were generated based on four user-defined parameters: (i) radio range, (ii) speed, (iii) area, and (iv) pause time.

In a random waypoint model, every node is initially uniformly distributed within a two-dimensional space. Each node then moves to a random uniformly distributed selected destination at a certain speed. When a node reaches its destination, it pauses for a certain time before it selects another destination and starts moving again towards that destination. The radio range of the nodes in the analysis was 100, 150, 200, and 300 meters. The speed was selected to be 3, 5, 10, 15 m/s in an attempt to map situations where nodes may move slower, such as in the case of people walking, or in the case where nodes move faster, e.g., in cars. The area was fixed to 1000 m x1000 m. The pause time was set to 5 seconds and the simulation time was 20,000 seconds. The results ensured a 5% or smaller deviation from the mean with a 95% confidence interval.

Unlike the distribution of the Monte Carlo analysis, the node distribution in a random waypoint model was not uniform and nodes tended to concentrate at the center of the two-dimensional space. Bettstettter analyzed the statistical properties of the random waypoint model in more depth in [14][15].

The initial random distribution of mobile nodes in the random waypoint model does not represent the manner in which nodes distribute themselves when moving [115][108]. A warm-up period of 1000 seconds was set up to attain a steady state behavior as was suggested in [115]. To validate this warm-up period, simulation runs with warm up-periods of 100,000 seconds were carried out, which indicated that a longer warm-up period did not significantly affect the results obtained.

5.4.2.5. Routing Protocols

The routing protocols utilized in the simulation were the Ad Hoc On-Demand Distance Vector (AODV) protocol [116] and the Optimized Link State Routing (OLSR) protocol [117]. The selection of these two protocols was based on their availability and proper functionality in NS2.

AODV is a reactive routing protocol for MANETs [116], whereas OLSR is a proactive MANET routing protocol [117]. With a reactive protocol the routing paths are built on demand whenever a node needs to send packets to a peer and it does not have a known route to that peer. In contrast, with a proactive protocol a node maintains routing

paths to its peers by periodically updating its routing table through the broadcast of control messages.

The potential advantage of AODV is that it discovers routes on demand and, therefore, it is not required to send periodic routing packets. The AODV algorithm avoids the Bellman-Ford "count to infinity" problem and, therefore, quickly converges when connectivity changes [116]. The disadvantage of AODV is that it may require a high control overhead in environments with high mobility and heavy traffic loads. In addition, it relies on blind broadcasts for route discovery because it is unable to determine whether a destination was unreachable or whether the route request message was lost [118].

With proactive protocols, the periodic broadcasts increase the overhead in the network. In an effort to reduce this control overhead, OLSR uses selected routers that broadcast messages as opposed to flooding mechanisms [117][119]. Clausen, et al. [117] stated that OLSR is particularly suitable for large and dense networks that tend to be more connected. OLSR is not very suitable for highly partitioned networks and is less responsive to changes in connectivity since the route discovery is not carried out on demand. Another disadvantage is that the path discovered by OLSR may not be the shortest.

The deployment of reactive versus proactive protocols has been investigated by Lin [71] and recommendations were made with regards to selecting a routing protocol that is suitable for a particular MANET environment. The objective of this research was not to compare these protocols but to assess the effectiveness of the KMS with each protocol.

5.4.2.6. Communication Interval

The communication interval was the frequency of repeating a particular scenario. In the case of certificate issuance and acquisition, the communication interval set the frequency with which a node attempted to communicate with a DCA/TP. In the case of revocation, the communication interval set the frequency with which each of the DCAs attempted to communicate with the TPs of a node.

The communication interval was set to 15 seconds for a number of reasons. First, a short interval could more accurately reflect the performance of the KMS during the dynamic changes of connectivity in the network for metrics that recorded time of execution. More specifically, the metric *APBNCS* recorded the period between non-consecutive successful attempts. Since successful attempts depended on connectivity, a shorter communication interval would more likely detect the point of break of connectivity and more accurately predict the time interval between communications when the network partitioned. Second, in the case of *APR*, the average period of revocation, the short time interval reflected the necessity to communicate the revocation information to the nodes in a timely manner and, thus, gave the minimum time to carry out revocation. In addition, in the case of *APRC*, the average period to reach a group of CAs, the short period reflected the urgency to conduct enough CAs to issue or reissue a certificate. Therefore, *APRC* indicated the shortest possible time to obtain a certificate.

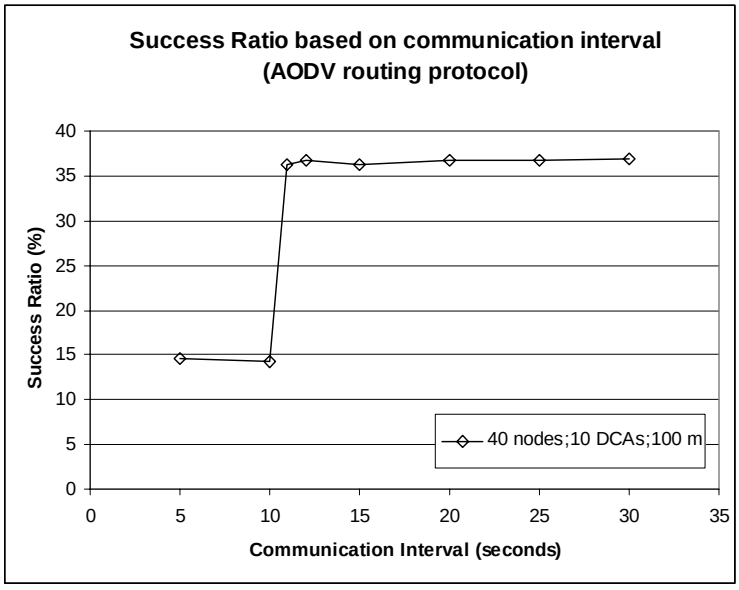


Figure 5.25. Impact of communication interval with AODV.

The lower limit on the selection of the communication interval value was imposed by the AODV routing protocol (see Figure 5.25). The interval was set to be greater than or equal to the MAX_RREQ_Time-Out (maximum route request time-out) parameter value of AODV. The MAX_RREQ_Time-Out of 10 seconds was the timeout period

after a series of failed network-wide searches, i.e., if a series of route requests for a particular destination failed then the AODV routing protocol did not attempt to discover the same route for a period of 10 seconds. As a result, the routing entries remained the same for 10 seconds even though connectivity changed. Extra attempts during that interval simply failed, yielding a lower percentage of successful connections. The *Success Ratio* increased from 14% to 36% for communication intervals greater than 10 seconds. It is important to note that the *Success Ratio* metric was not impacted by the communication interval as the 15, 20, 25, and 30 seconds intervals gave approximately the same value for the Success Ratio.

5.4.3. Simulation Analysis

The following subsections present the analysis of the results of the NS2 simulation. The KMS functionalities analyzed were certificate issuance and acquisition, certificate revocation, and certificate issuance in threshold cryptography schemes.

5.4.3.1. Certificate Issuance and Acquisition

Figure 5.26 and Figure 5.27 depict the Success Ratio of the KMS based on the number of DCAs and TPs and, more specifically, its effectiveness in distributing certificates in a partitioned environment. (The corresponding partitions based on radio range are displayed in Figure 5.23.) Figure 5.26 was generated while utilizing the AODV routing protocol whereas Figure 5.27 was generated while utilizing the OLSR routing protocol. Even though the objective of this research was not to compare the routing protocols, Figure 5.26 and Figure 5.27 demonstrate that AODV provided higher Success Ratio compared to OLSR.

As previously mentioned, the effectiveness with respect to certificate issuance could be observed by considering the centralized case to 20% DCAs and TPs shown on the x-axis of the graph. For the case of certificate acquisition, the whole range of DCAs and TPs (1 DCA (centralized case)-40%) could be considered, since nodes could either obtain certificates from DCAs or TPs. As expected, the Success Ratio for a centralized system, demonstrated by the data points on the far left of the graphs, was lower compared

to the scenarios that involved more than one DCA or TP. The existence of 10% DCAs (of the total number of nodes), improved the ability to issue/reissue a certificate. In addition, the existence of 10% of TPs for a node (20% combined DCAs and TPs) could more easily facilitate the establishment of SAs.

Figure 5.26 shows that the relative increase in the Success Ratio as the number of DCAs and TPs increased was higher for the 40-node network as compared to the 80-node and 120-node network, because the 40-node network was more partitioned (as shown in Figure 5.23). As a result, the 80-node and 120-node lines tended to be horizontal or have a smaller gradient when utilizing greater than 10% DCAs or TPs. The OLSR protocol was less sensitive to changes in connectivity due to its proactive nature and, therefore, a higher node density yielded an overall lower success ratio as compared to the AODV protocol (see Figure 5.27). In addition, since OLSR did not reactively build routing tables, the Success Ratio to obtain a certificate was more dependent on the existence of DCAs/TPs as compared to AODV. Thus, utilizing a higher number of DCAs and TPs did not quickly yield an almost horizontal line as in the case of using greater than 10% DCAs with AODV (see Figure 5.26), but rather a more inclined line showing a higher relative increase in the Success Ratio with the usage of DCAs/TPs.

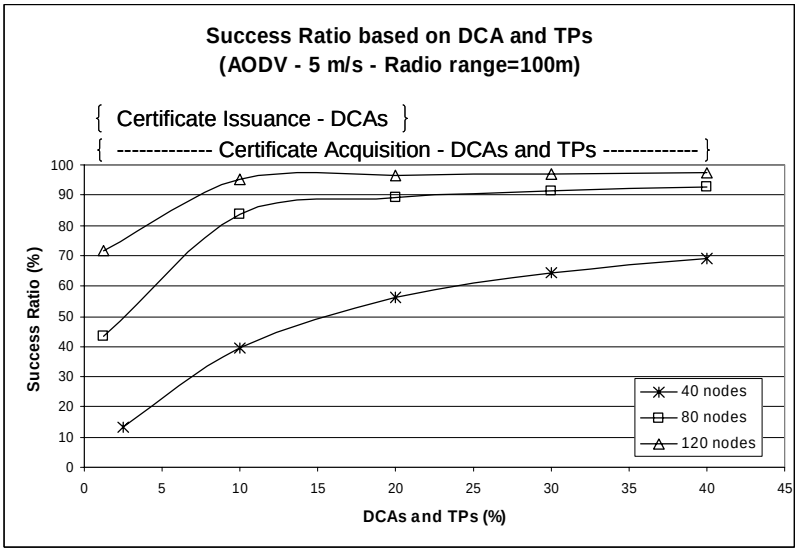


Figure 5.26. Success Ratio with the AODV protocol.

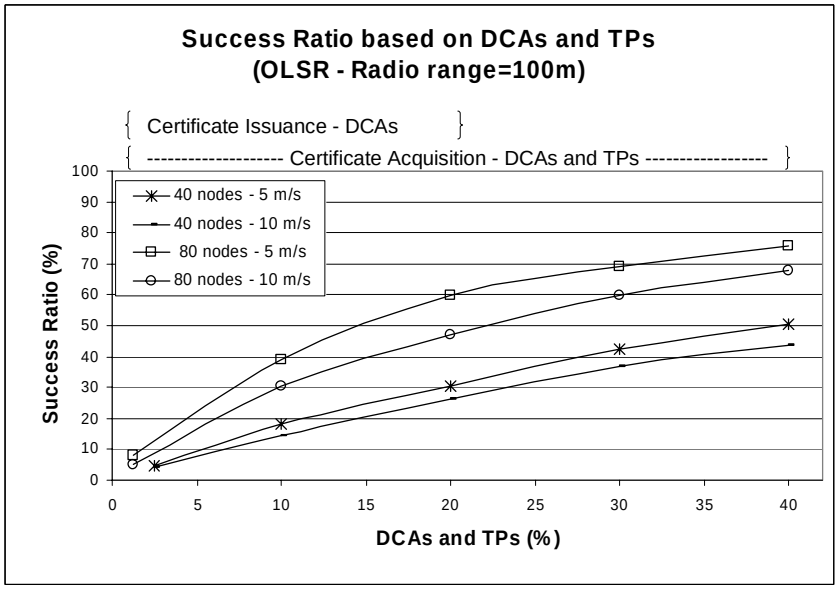


Figure 5.27. Success Ratio with the OLSR protocol.

Figure 5.28 and Figure 5.29 indicate the impact of node speed on the Success Ratio with the AODV and OLSR protocols, respectively. These graphs could act as a guide to derive the possible deviation of the Success Ratio with regards to the node speed. With the AODV protocol the Success Ratio increased with the increasing node speed due to the reactive nature of the AODV protocol. In contrast, the Success Ratio with the OLSR protocol decreased due to its proactive nature.

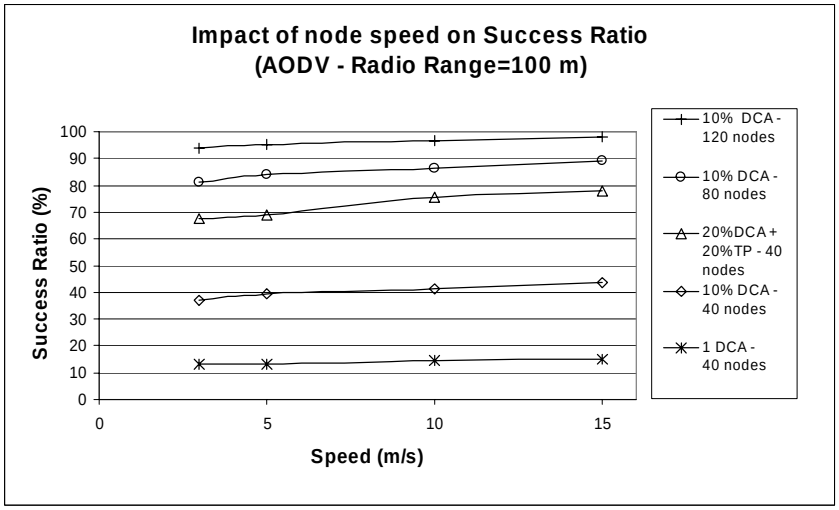


Figure 5.28. Success Ratio deviation based on the speed of nodes (AODV).

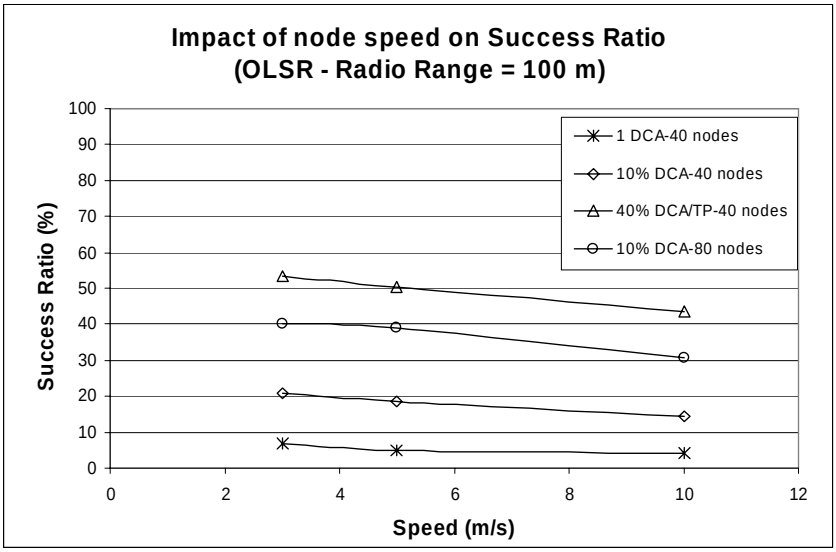


Figure 5.29. Success Ratio deviation based on the speed of nodes (OLSR).

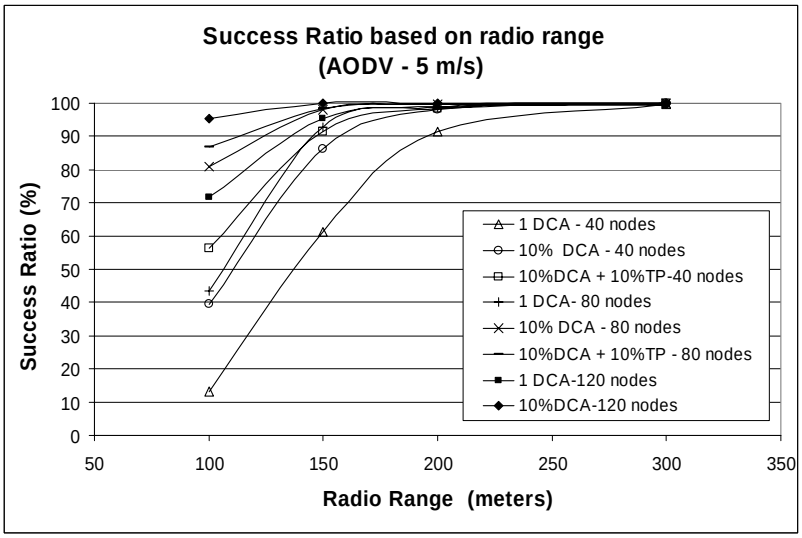


Figure 5.30. Impact of radio range on the Success Ratio (AODV).

Figure 5.30 and Figure 5.31 indicate the variation of the Success Ratio with regards to the radio range of the network nodes. As the radio range increased, connectivity increased causing the Success Ratio to converge to 100%. The usage of DCAs and TPs pushed the Success Ratio to 100% at a shorter radio range compared to the centralized case. For example, in Figure 5.30, the Success Ratio with the centralized case in a 40-node network converged to approximately 100% at 300 meters radio range instead of 200 meters when using 10% DCAs and 10% TPs. As previously mentioned,

the OLSR protocol was less responsive to connectivity and thus the radio range required to converge to 100% Success Ratio was longer.

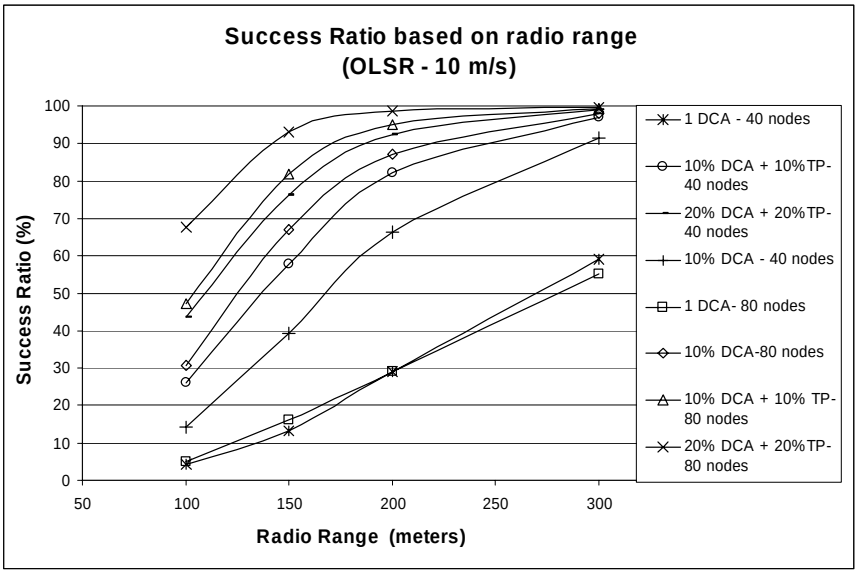


Figure 5.31. Impact of radio range on the Success Ratio (OLSR).

Figure 5.32, Figure 5.33, and Figure 5.34 demonstrate the average period between non-consecutive successes and, more specifically, the impact of network partitioning for highly partitioned network environments (with radio range of 100m). In Figure 5.32, with the centralized case, a node had to wait for an average of approximately 510 seconds (8.5 min) for the OLSR protocol and 320 seconds (5.3 min) for the AODV protocol to reissue its certificate or obtain a certificate of its peers to establish an SA. However, by utilizing 10% DCAs or higher the APBNCS was kept below 100 seconds (1.5 minutes) in most of the scenarios. As the number of DCAs and TPs in the network increased, the APBNCS converged to 50 seconds and was less impacted by network partitioning. Figure 5.33 depicts the impact of radio range on the APBNCS metric. As the radio range increased, connectivity increased and APBNCS converged to 30 seconds. With the OLSR protocol a node required a higher average period compared to the AODV protocol. Figure 5.34 presents a different perspective of the APBNCS. As the speed increased the impact of partitions on the APBNCS was smaller. It is important to note that in the centralized case the average period decreased from 280 seconds to 130 seconds, which was a higher rate

compared to the other scenarios. However, the other scenarios already possessed a relatively lower APBNS compared to a centralized system.

These APBNCS values used to reissue a certificate could provide practical guidance in selecting the expiration date for the *window of time* field that is needed for the execution of routine revocation. For example, if there were 10% DCAs in a highly partitioned network of 40 nodes with the OLSR protocol and routine revocation had been triggered and communicated to the nodes, then the period required by the nodes to reissue the certificates might be greater than 150 seconds (10% DCAs) and smaller than 510 seconds (1 DCA), as shown in Figure 5.32. The expiration date had to, therefore, consider these APBNCS values to alleviate the impact of partitioning and allow nodes sufficient time to obtain new certificates. (This example assumes that routine revocation had already been communicated via broadcasting. The study of the efficiency of broadcasting methods is beyond the scope of this research.)

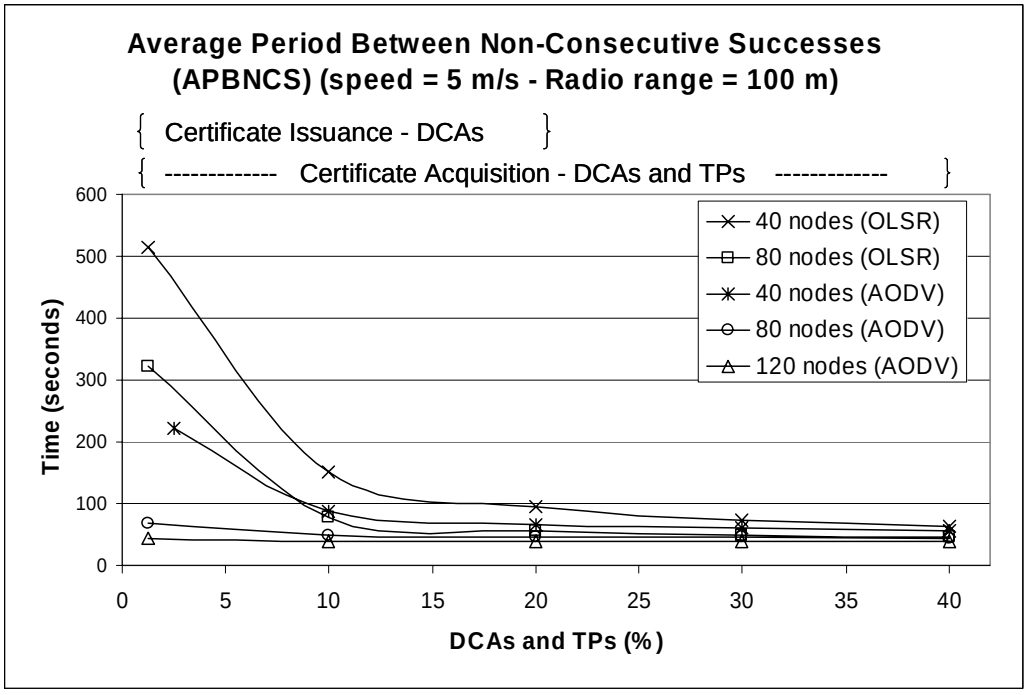


Figure 5.32. Average period to obtain service.

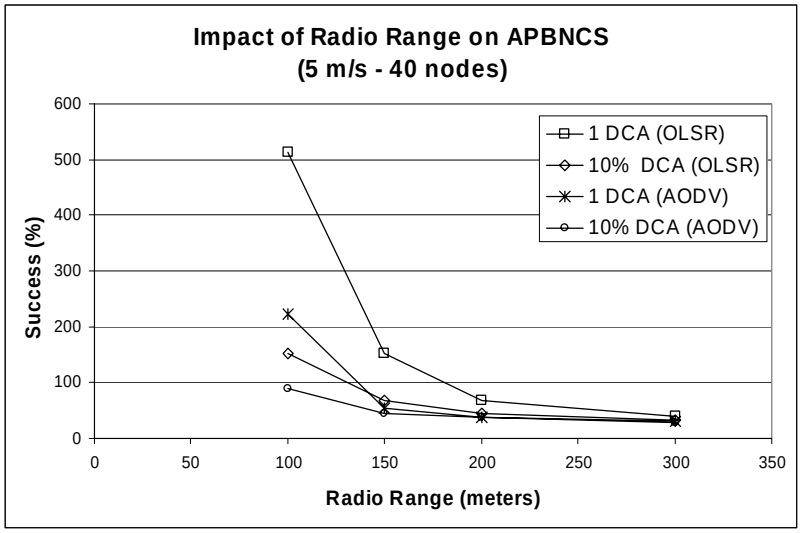


Figure 5.33. Impact of radio range on APBNCS.

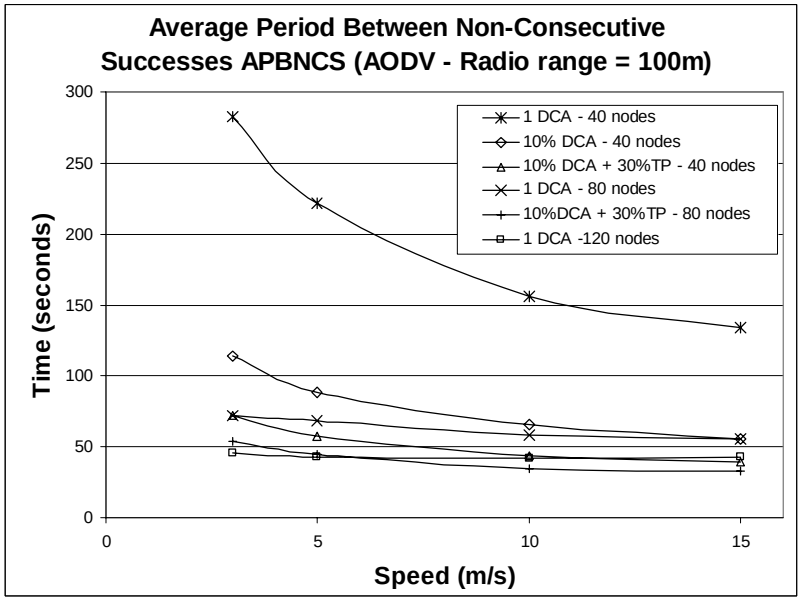


Figure 5.34. Impact of node speed on APBNCS.

5.4.3.2. Certificate Revocation

In this simulation, an extended scenario of revocation was investigated. Revocation was the ability of a group of DCAs to contact the TPs of a malicious node and inform them of the revocation of their peer's certificate. The revocation was initiated by a single DCA and was triggered when the bad behavior grading of a particular node

met a security threshold. The single DCA would communicate its revocation decision to a group of SDCAs, which in turn would approve the action and inform the TPs of the malicious node. The effectiveness of this scheme was investigated using 1 to 5 SDCAs out of the total number of DCAs in the network. Data were collected to reflect the time required for the node's TPs to receive 1 to 3 RNs. It was assumed that once a node received one RN it would be more conscious with its interaction with the revoked node. Once it received more than one RN it could dissolve its SA with the malicious node.

Figure 5.35 demonstrates the average time required to contact all the TPs of a node based on the radio range, speed, and node density. All scenarios were based on three SDCAs in the network and each of the TPs received two RNs. The APR decreased exponentially as the radio range of the nodes increased. More specifically, the time required to inform the TPs of a node was less than 100 seconds at a radio range greater than 150 meters for most of the scenarios. The two exceptions to this observation were the higher node density network (80 nodes) with node speeds of 10 m/s and 15 m/s. At that speed the APR decreased at a relatively lower rate as the radio range increased when compared to the other scenarios because connectivity in the network changed at a faster rate and it was more difficult to communicate with the TPs of a node.

Figure 5.36 presents a different perspective of the revocation effectiveness by fixing the radio range at 100 meters with the AODV protocol, which represents a highly partitioned network. It depicts the total time required to contact a percentage of a node's TPs and inform them with two RNs. For example, if a node in a network of 40 nodes had 10% TPs (of the total number of nodes) and was moving at 3 m/s it would take 420 seconds (7 min) for 100% of its TPs to receive two RNs and 250 seconds (4.2 min) for 75% of its TPs to receive two RNs. As expected a higher number of TPs increased the total APR and a higher node speed decreased the total APR. The 80-node network provided a lower APR compared to the 40-node network. In addition, with a denser network, the APR was less impacted by the node speed (see Figure 5.36; 80-node scenarios with 3 and 15 m/s).

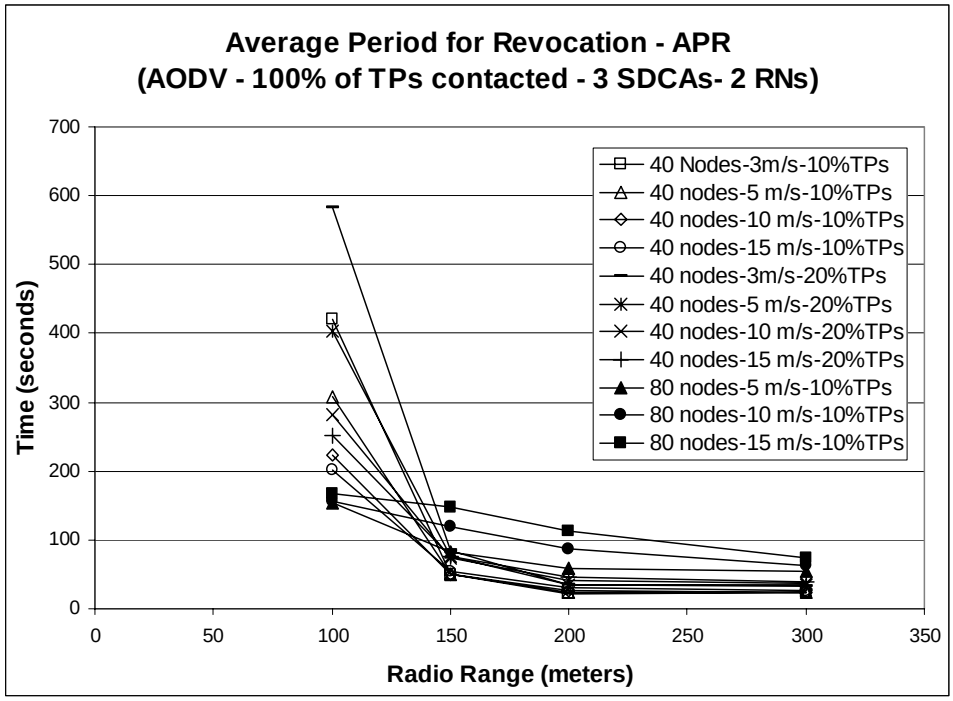


Figure 5.35. Impact of radio range, speed, and node density on revocation (AODV).

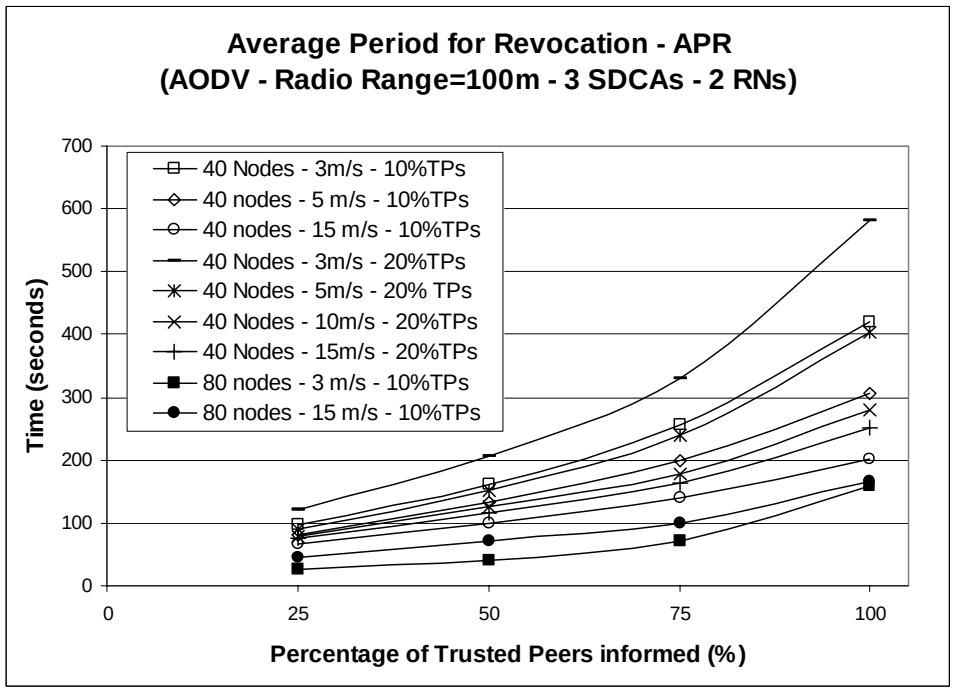


Figure 5.36. Percentage of TPs reached by DCAs (AODV).

Figure 5.37 illustrates the impact of the number of SDCAs and RNs on revocation in a highly partitioned environment. The node speed was 3 m/s, which yielded the maximum APR values. As the number of SDCAs increased, the average period to contact the node's TPs decreased and as the number of RNs required increased the APR increased. For example, with the 40-node network and 3 SDCAs, the APR to contact 100% of the node's TPs increased by approximately 300 seconds when the number of RNs increased from 2 to 3. In addition, Figure 5.37 shows that the relative increase in the APR was lower with the usage of multiple SDCAs as the percentage of a node's TPs to be contacted increased from 25% to 100%. As a result, the APR curves had a lower slope as the SDCAs increased assuming the number of RNs remained the same. Furthermore, the relative impact of the SDCA number on the APR for the 80-node network was lower compared to the 40-node network because the network was more connected. Figure 5.38 demonstrates the impact of SDCAs based on the radio range. All the scenarios in the graph assumed that the TPs of a node received three RNs. As the number of SDCAs increased the lines shifted lower due to a lower APR value.

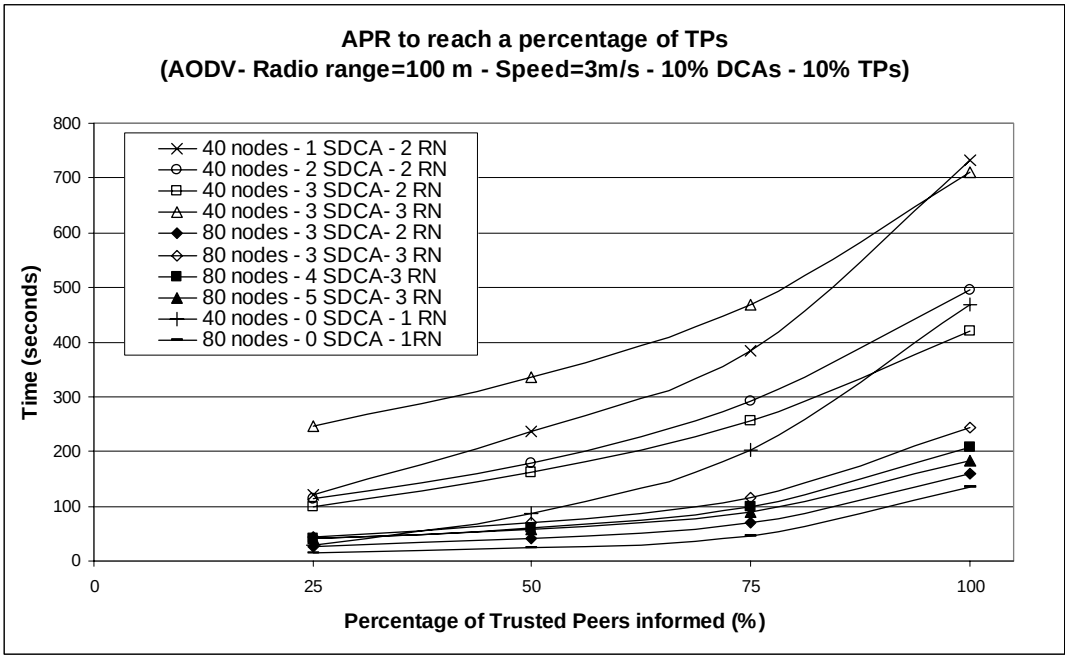


Figure 5.37. Impact of SDCAs and RNs on APR (AODV).

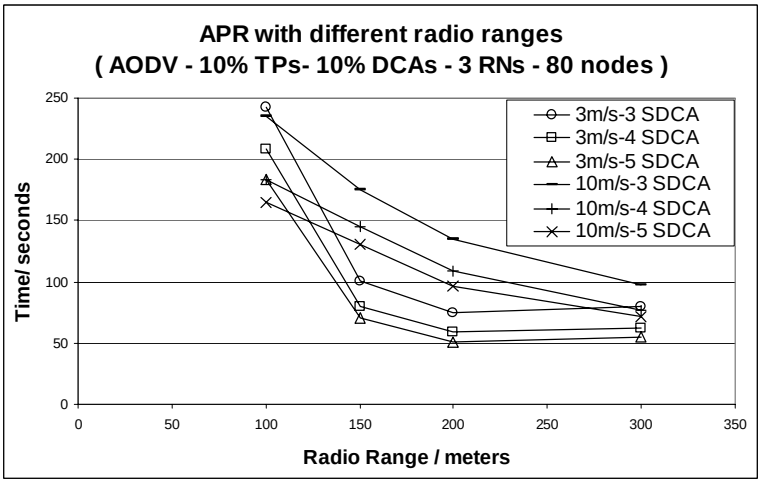


Figure 5.38. Effectiveness of SDCAs during revocation (AODV).

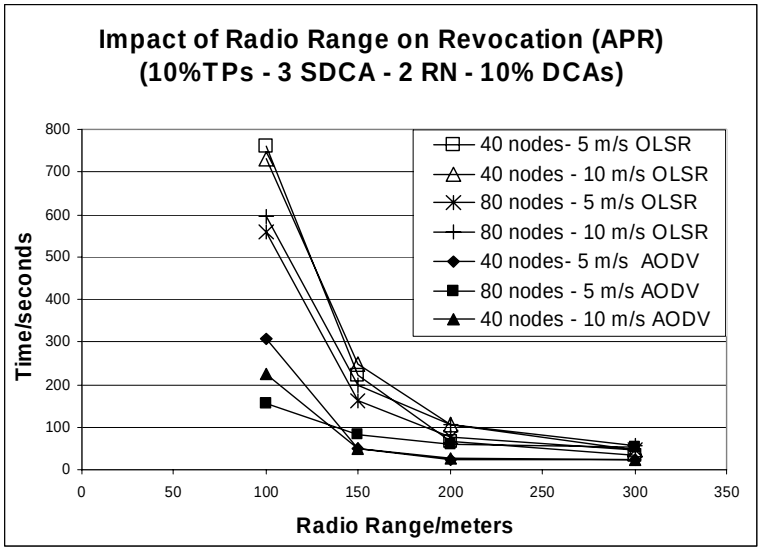


Figure 5.39. Impact of radio range on APR with AODV and OLSR.

Figure 5.39 and Figure 5.40 present the revocation results when using the OLSR and the AODV protocols. The APR with the OLSR protocol was a lot higher compared to the APR with the AODV protocol. For example, with the 40-node network and 5 m/s node speed with OLSR, the APR at 100 meters is 760 seconds compared to 300 seconds with AODV (see Figure 5.39). However, as shown in Figure 5.38, 75% of the node's TPs would be informed in 560 seconds, i.e., 200 seconds earlier. In addition, this type of low availability signifies the importance of revocation alerts utilized in the KMS. The

DCAs would utilize stricter behavior threshold and inform the TPs of nodes more frequently via security alerts. As a result, the nodes would be more aware of any malicious behavior of their TPs without being as dependent on revocation.

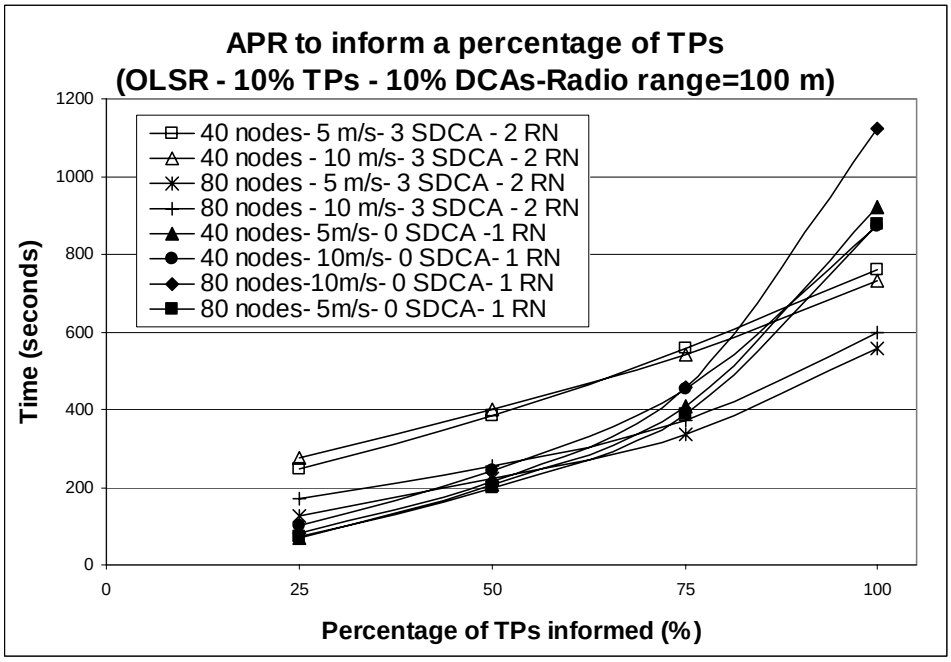


Figure 5.40. Percentage of TPs reached by DCAs with OLSR.

5.4.3.3. Threshold Cryptography Schemes

Threshold cryptography schemes lead to less accessibility to CAs during certificate issuance especially in partitioned environments. Figure 5.41 shows the time taken for a node to contact a percentage of CAs. For example, the time taken to communicate with 100% of the 10% of CAs (4 CAs for 40-node network) and obtain partial certificates ranged between 900 and 1100 seconds for the OLSR protocol and 100 and 500 seconds for the AODV protocol. In our KMS, a node could obtain a certificate within a period of 100 seconds since it only needed to contact one DCA. Figure 5.42 demonstrates how the average time to contact 100% of the CAs varied based on the radio range. As expected, with the OLSR protocol a node required a longer time to contact a group of CAs.

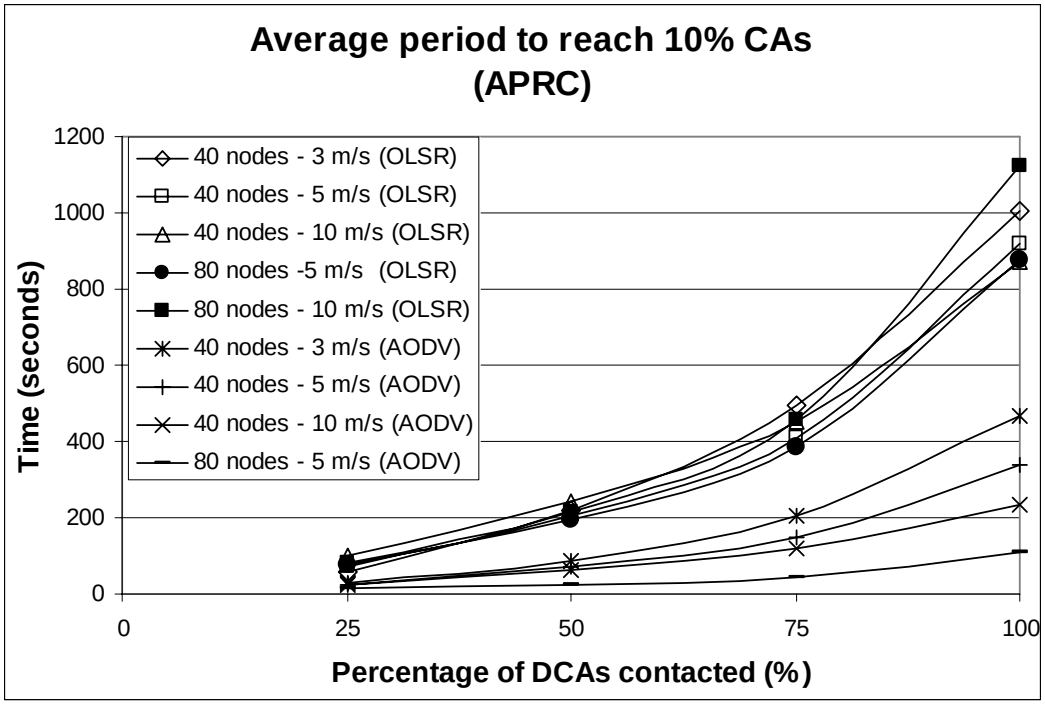


Figure 5.41. Average period to issue a certificate.

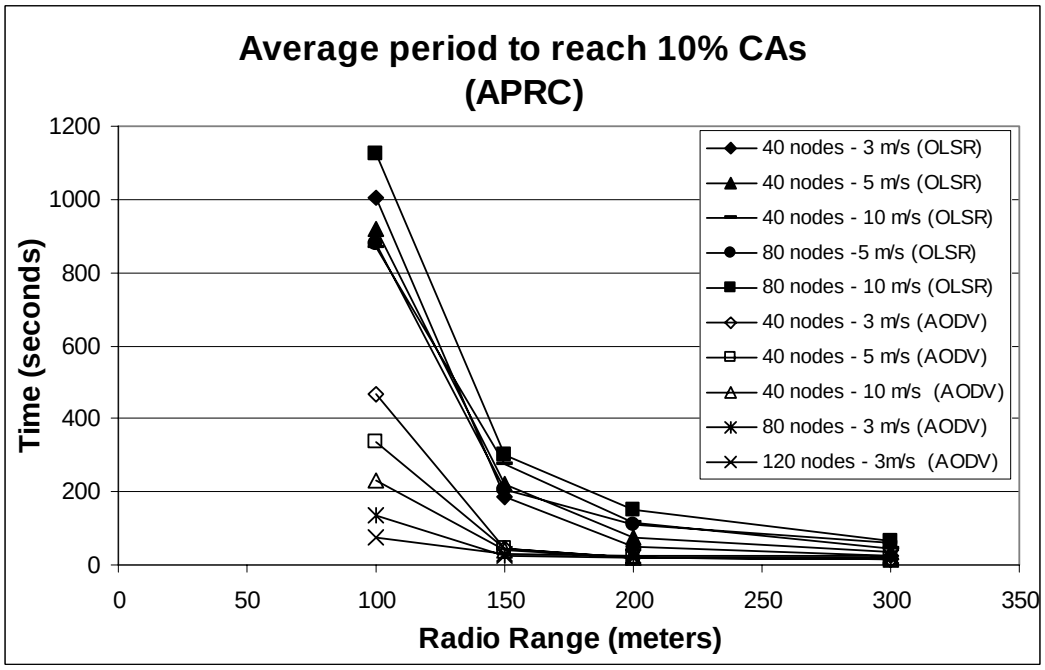


Figure 5.42. Impact of radio range on certificate issuance.

5.4.3.4. Out-of-band TCA and DCA Authentication

To assess the need as well as the effectiveness of the existence of the DCA and TCA control plane, an analysis was carried out to investigate the time that it would take for a node to come to close proximity to a DCA or TCA and obtain a certificate via out-of-band methods. The mobility model used was the random waypoint model. Figure 5.43 depicts the time required for a node to come within 5 meters of a DCA or TCA when moving at 1 m/s, 3 m/s, and 5 m/s. Peers in a MANET would physically encounter one another to perform out-of-band authentication and exchange their ID information. In [8], the authors used a secure communication range of 5 meters and we adopted the same range in our analysis as well.

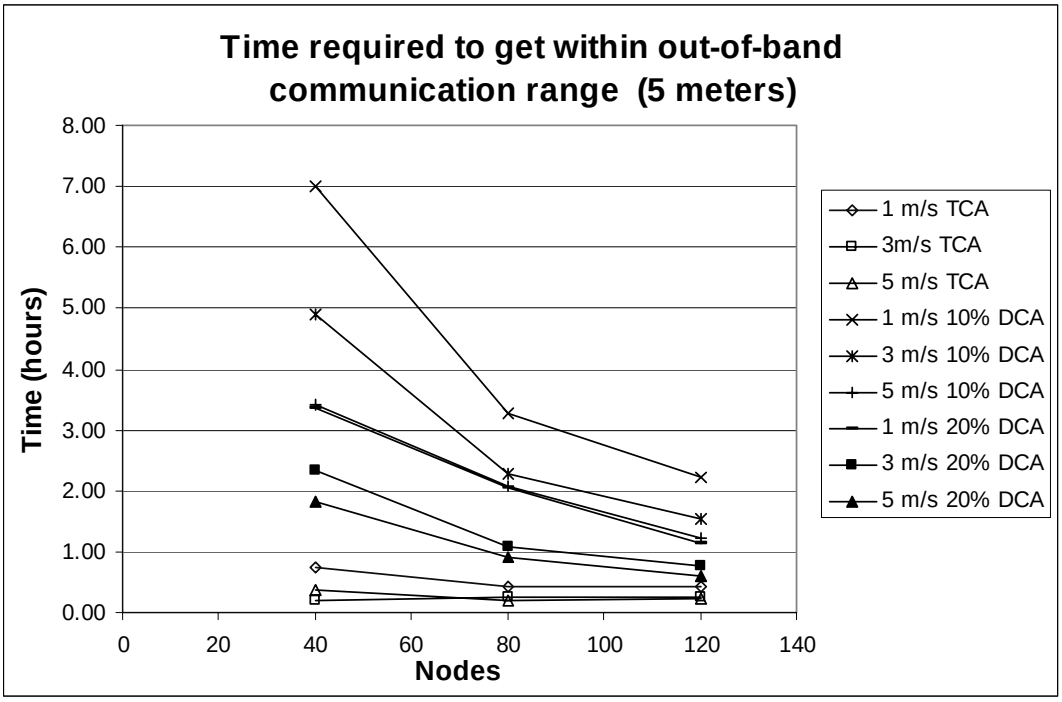


Figure 5.43. Balancing availability with DCAs and TCAs.

Figure 5.43 showed that the time taken to come to close proximity to a DCA is relatively longer compared to a TCA. For example, a new node moving at 5 m/s in a 40-node network took approximately 0.4 hours (24 minutes) to come close to a TCA and between 1.8 hours (20% DCAs) to 3.4 hours (10% DCAs) to come close to a DCA.

Thus, the existence of the TCA functionality provided increased service availability to a new node until it could register with a DCA. Even though, these results could vary with a different mobility model they still provided some level of validation of the importance of the existence of multiple DCAs and TCAs.

5.4.4. Summary of Mobility Analysis

The data collected provided some practical guidance on the selection of the KMS parameters such as security policies and number of DCAs based on the network connectivity. The effectiveness of the KMS is shown in Table 5.7. For example, in a partitioned network, certificate issuance with 10% DCAs for the 40-node network increased by 14% compared to the centralized case for both OLSR and AODV. Certificate acquisition for the 40-node network with 10% DCAs and 10% TPs increased by 25% for OLSR and 43% for AODV, as compared to the centralized case.

The dependence of a group of CAs in threshold cryptography schemes increased the response time to obtain a certificate. This KMS allowed nodes to obtain certificates within a much shorter period of time (see Table 5.8). This analysis has also validated the existence of the control plane of DCAs and TCAs. Nodes took a lot longer to communicate with a DCA compared to a TCA

Table 5.7. Percentage Increase in Service Availability Compared to the Centralized Case

Routing Protocols	Network Size	Certificate Issuance - 10 % DCAs (%)	Certificate Acquisition - 10% DCAs & 10% TPs (%)
OLSR	40	14	25
	120	35	52
AODV	40	14	43
	120	23	24

Radio range = 100 meters

Table 5.8. APR with Threshold Cryptography Schemes and Our KMS

Routing Protocols	Network Size	APR / seconds (with 5 m/s)	APR / seconds (with 10 m/s)
OLSR	40	921	871
	80	1125	877
AODV	40	338	233
	80	111	97
This KMS			
AODV OLSR	40	14 – 82 seconds for all scenarios	
	80		

Radio range = 100 meters; 40 nodes; 10% CAs

Finally, Table 5.9, Table 5.10, and Table 5.11 summarize the results for revocation. Table 5.9 shows the average period taken to distributed two RNs to 75% and 100% of a node's TPs in a highly partitioned network. Table 5.10 demonstrates the advantage of utilizing a higher number of SDCAs. For example, in a 40-node network where the nodes moved at 3 m/s, increasing the number of SDCAs from 1 to 3 dropped the period of revocation from 732 to 431 seconds. Table 5.11 depicts the impact of requiring the KMS to distribute a different number of RNs. The revocation analysis quantified the period taken to revoke nodes and justified the need for revocation alerts as part of a KMS.

Table 5.9. Revocation with 3 SDCAs and 2 RNs

Routing Protocols	Node speed / m/s	75 % of TPs	100% of TPs
AODV	3	256	421
	15	151	202
OLSR	5	540	730
	10	557	761

Radio range = 100 meters; 40 nodes; 10% DCAs; 10% TPs

Table 5.10. Impact of SDCAs on Revocation

Nodes	RN	SDCAs	APR / seconds (with 3 m/s)	APR / seconds (with 10 m/s)
40	2	1	732	381
		2	495	254
		3	421	202
80	3	3	243	235
		4	208	184
		5	184	165

Radio range = 100 meters; 10% DCAs; 10% TPs; AODV

Table 5.11. Impact of Received RNs on Revocation

Nodes	RNs	75 % of TPs	100% of TPs
40	2	256	421
	3	410	710
80	2	71	160
	3	117	243

Radio range = 100 meters; 10% DCAs; 10% TPs;3SDCAs;AODV

5.5. Implementation of the KMS

The KMS was implemented in the testbed described in Figure 3.2 and described in [74]. The objective of this implementation was to provide a proof of concept of the effectiveness of the KMS in distributing certificates and provide proof of its interoperability with the existing FreeS/WAN IPsec implementation. The KMS functionalities that were implemented were: (1) DCA and TP modules, and (2) positive behavior reputation with DCA synchronization.

The existing IPsec implementation offered limited service availability due to strict dependence on the DNSs. Nodes authenticated each other by obtaining the public key of

their peers from the DNS (as shown in Figure 5.44). If any of the DNSs were not available, which is a common challenge in a MANET due to network partitioning, an IPsec SA negotiation failed. To address certificate availability of the KMS, the existing IPsec implementation was made aware of the various sources that could provide the required authentication information needed to establish an IPsec SA. The increased service availability was provided through DCAs and TPs.

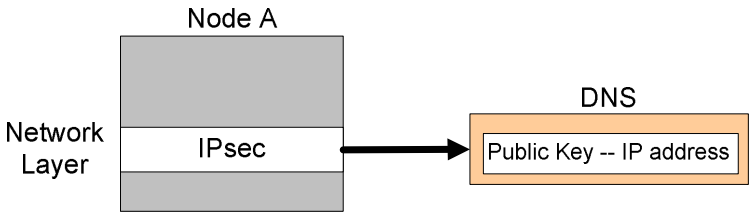


Figure 5.44. Existing IPsec architecture.

The existing IPsec implementation was modified so that IPsec on each Gateway communicated with a KMS client, instead of a DNS, whenever it had to establish an IPsec tunnel with a peer Gateway (see Figure 5.45). The KMS client was deployed on each Gateway in the MANET and was responsible for collecting the required information on behalf of IPsec. DCAs were deployed on a few of the subnet nodes to be protected by the subnet Gateways. The DCA functionality was implemented as a stand-alone application instead of modifying the DNS implementation. Modifying the DNS did not offer any additional functionality to the KMS.

During an SA establishment, the KMS client on that Gateway would first query any of the available DCAs. If none of the DCAs were available, the KMS client on that Gateway dynamically switched to the TP configuration to obtain the certificate from the TPs of its peer. The process of TP set up was automated and required no extra transactions. Nodes that acquired their peers' certificates before establishing SAs, stored them and distributed them on behalf of their peers. The certificates on the DCAs complied with X.509 v 3 certificates and were extended to provide any other information, such as behavior grading (see Section 4.4). Once peers successfully authenticated each other and an SA was established, the IPsec mechanism reported the trust to the KMS client, which in turn reported the information to the DCA. In addition, each node reported

its SA to the TopoView application (described in Section 3.3.1). The TopoView application was modified to receive this information and dynamically display the IPsec tunnels in the network (see Figure 5.46).

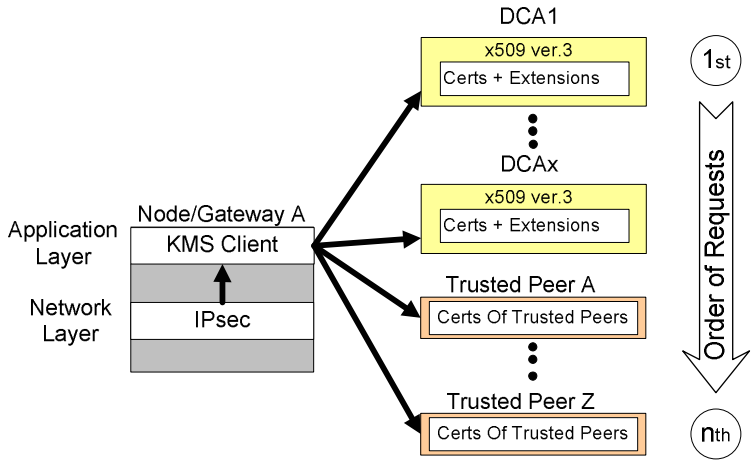


Figure 5.45. Modified IPsec implementation.

DCAs distributed the positive reputation information of the nodes in the network. Each DCA periodically synchronized its individual updates with the rest of the DCAs. Each positive reputation update and event in the KMS was assigned a serial number and each DCA kept a record of the latest serial number of its peer DCAs. The synchronization was achieved in two steps. First, a DCA advertised the latest serial number that corresponded to its latest update. Second, the peer DCAs requested information about the events that they had not received. The objective of the two phase synchronization was to introduce some level of flexibility in obtaining the desired updates. DCAs could have been unable to obtain information at a particular instant based on the network characteristics, such as partitioning.

Summarizing, the KMS was implemented and integrated with the existing IPsec implementation. The combination of DCAs and TPs provided higher functionality and facilitated the establishment of IPsec SAs between nodes. Integration with TopoView provided a dynamic visual representation of the SAs established in the network as shown in Figure 5.46.

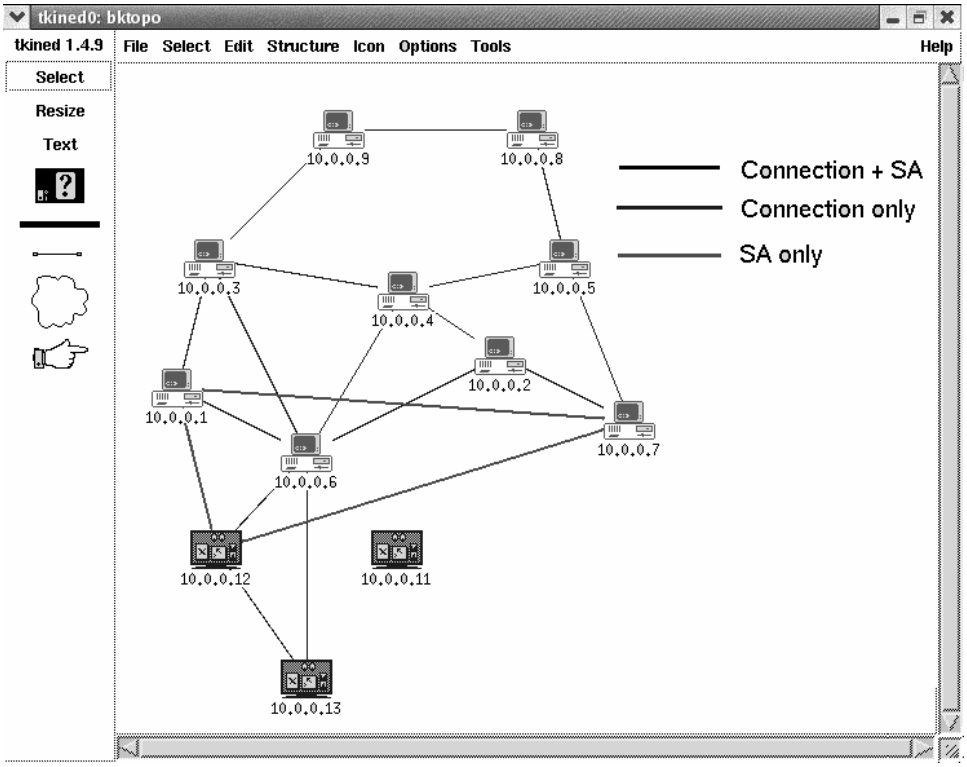


Figure 5.46. Dynamic picture of connectivity and SAs in the backbone topology.

5.6. Integrating Threshold Cryptography

The data collected in the Monte Carlo simulation analysis and the mobility simulation proved that threshold cryptography is unsuitable for highly partitioned MANETs because threshold cryptography schemes were dependent on the availability of multiple CAs. In addition, this analysis did not account for the possibility of malicious CAs, which could further decrease the availability of the system.

In more connected environments where high availability is not required, the proposed KMS could be integrated with threshold cryptography schemes. The resulting KMS would complement the revocation authorization and response weakness (see Section 2.9) of threshold cryptography schemes through behavior grading. The behavior grading scheme of this KMS could provide the control plane of CAs with a justification to initiate revocation for a malicious node. In addition, this KMS could enable information propagation and verification across multiple CAs during the process of node

registration. Through the certificate issuance procedure described in Section 4.2.2, a node could more easily register into the network and obtain a certificate. Once it registered out-of-band with one CA, the information would propagate to the other CAs, which could provide it with more partial certificates. Of course, this integration would require that the combiner node that combines the partial certificates is a CA, as presented by Zhou and Hass [2] so that CAs would be able to grade one another. Overall, the system would be more resistant to security attacks by preventing the existence of invalid certificates and isolating malicious nodes and CAs.

5.7. Conclusion

This chapter presented results and analysis of the proposed research problem. The effectiveness of the KMS to provide service to the nodes was demonstrated via both Monte Carlo and NS2 simulation. The functionalities investigated were certificate issuance, certificate acquisition, and revocation. The control plane of DCAS and TPs increased availability in a highly partitioned network.

The Monte Carlo simulation investigated the functionalities of the KMS by uniformly distributing nodes in a fixed area. Availability during certificate issuance with 10% DCAs and 100 meters radio range increased by at least 18% in the least connected environment (e.g., 40 nodes) and by a maximum of 51% for the most connected environment (e.g., 120 nodes) as compared to a centralized system. In addition to the presence of DCAs, the existence of 10% TPs of a node increased the availability for certificate acquisition to 36% (40 nodes) and 61% (120 nodes). The existence of 20% TPs further increased the availability to 49% (40 nodes) and 66% (120 nodes). The path length to communicate with certificate servers while utilizing at least 10% certificate servers stayed below 3 hops as compared to 8 hops in a centralized system. During revocation, a DCA reached less than 26% of a node's TPs in a highly partitioned environment justifying the need for security alerts for this KMS.

The threshold cryptography analysis showed that in a partitioned environment consisting of 40 nodes and a radio range of 100 meters, nodes only reached between 7% and 26% of the number of CAs in the network and were isolated with a probability

ranging between 4% and 33%. Therefore, nodes were dependent on multiple retries to be able to build their certificate. Our KMS in this environment offered four times higher probability, which guaranteed higher service availability to nodes (assuming 10% DCAs).

Unlike the Monte Carlo analysis, NS2 investigated the effectiveness of the KMS with regards to mobility. The results were based on the random waypoint mobility model in which nodes tended to concentrate at the center of the network area.

In a highly partitioned network, certificate issuance with 10% DCAs increased by 14% to 35% for the various scenarios when compared to a centralized system. With the aid of 10% TPs (20% DCAs and TPs), the probability of certificate acquisition ranged between 24% and 55% when compared to the centralize case. A further increase of TPs to 20% pushed the probability of certificate availability between 25% and 61%.

The revocation analysis quantified the effectiveness of revocation in a MANET. The analysis was performed by utilizing a different number of SDCAs and sending a different number of RNs to the TPs of nodes. In a highly partitioned network with 40 nodes, 100 meters radio range, and three SDCAs, the time to inform all of the TPs of a node (10% TPs) with two RNs was between 200 and 420 seconds using the AODV protocol. However, with the OLSR protocol the period to inform the node's TPs was between 730 and 760 seconds. Similar to the Monte Carlo simulation, this analysis emphasized the importance of security alerts for this KMS.

The significance of utilizing a control plane of TCAs and DCAs was also demonstrated. In a 40-node network with 10% DCAs, a node required approximately 24 minutes to come to close proximity to a TCA and perform out-of-band authentication. However, it took between 3.4 and 7.0 hours to get close to a DCA and register into a network with out-of-band methods. Thus, through a TCA a new node could temporarily establish SAs with existing nodes until physically encountering a DCA.

A comparison of this KMS with threshold cryptography schemes proved that threshold schemes provided lower availability for certificate issuance. The threshold cryptography analysis has revealed that nodes required between 880 and 1130 seconds with the OLSR protocol and between 100 and 340 seconds with the AODV protocol to build a certificate. In this KMS, between 10 to 80 seconds were sufficient to obtain a certificate.

The KMS was implemented and interoperated with IPsec and TopoView, a network topology monitoring tool. An overhead analysis was also provided with regards to the behavior grading of the KMS, certificate sizes, and certificate issuance and revocation. The behavior grading overhead could fluctuate based on the security policies of the KMS, which controlled the frequency of sending updated certificates. In addition, certificate issuance and revocation overhead was dependent on the number of SDCAs, which was again controlled by the KMS. Unlike the parameters just mentioned, the revocation overhead was also impacted by the number of TPs of a node that was independent of the KMS' functions. Chapter 6 summarizes the contributions of this research.

Chapter 6

Conclusions and Future Work

A mobile ad hoc network is a collection of independent mobile routers. MANET links are wireless, which results in communications that are less dependable than wired links and have capacity constraints. A MANET is vulnerable to eavesdropping and the nodes in this network often have little physical protection. To counteract some of these threats, a MANET can use mechanisms, such as IPsec, to secure transmitted data. However, prior to IPsec deployment, nodes need to establish SAs. During the establishment of an SA, two nodes authenticate one another using certificates, which are a primary form of identity verification. A KMS creates, distributes, and manages these certificates. Thus, the KMS is at the heart of the network's defenses.

This research investigated key management in a MANET and developed a KMS that was suitable for a highly partitioned MANET environment. The KMS overcame the limitations of previous systems. It increased service availability in a highly partitioned network environment, minimized pre-configuration of KMS's nodes, and accommodated new nodes joining the network. In addition, it provided a means of obtaining feedback based on the network activity in order to dynamically adjust its configuration. This chapter describes the work performed during this dissertation and highlights significant achievements and extensions to the existing state of the art.

6.1. Significant Results

This research provided a framework for a distributed KMS that increased service availability in highly partitioned networks. Our system integrated a number of components in a unique way to overcome the limitations of previous KMSs. The system

utilized a modified hierarchical PKI model consisting of a control plane of RCAs, DCAs, and TCAs.

The KMS increased availability through the use of the following functionalities.

1) ***Multiple DCAs with two methods of authentication***. Multiple DCAs could sign certificates to nodes after authenticating via out-of-band methods or via peer connectivity. Peer connectivity promoted flexibility in the network because the nodes did not have to be physically collocated with the DCAs. As compared to threshold cryptography, certificate issuance was not dependent on a group of CAs.
2) ***TCAs***. Any node in the network could sign certificates for new nodes joining the network to establish temporary SAs. Based on our analysis, a new node in a highly partitioned network authenticated out-of-band with a TCA within a relatively shorter time compared to authenticating with a DCA.
3) ***TPs***. TPs stored the certificates of the nodes with which they shared an SA increasing the certificate availability. TPs spread the repository overhead and were dynamically assigned based on trust distribution in the network.
4) ***Routine revocation.*** The certificates of the nodes did not expire and revocation was initiated according to the KMS policies. Different policies allowed a dynamic balance between availability, security, as well as network overhead. The expiration date recorded in the *window of time* field allowed nodes sufficient time to obtain an updated certificate.

Based on the functionalities just mentioned, the system introduced more flexibility since *new* nodes could establish SAs after obtaining an RCA, or a TCA, or a DCA certificate. In addition, *existing* nodes in the network established SAs after obtaining their peer's certificates from either a TP or a DCA.

New nodes could use their RCA certificates to register in the network and serve as DCAs. Registration was carried out dynamically and a node was authorized by sending a request with its RCA certificate to the existing control plane of DCAs. This system promoted minimal pre-configuration of the DCAs at the initial steps of KMS's deployment and facilitated dynamic deployment of additional DCAs during the network lifetime.

The behavior grading scheme of the KMS provided a means of obtaining feedback based on the network activity. The behavior grading scheme relaxed the need of relying on strict identity verification and allowed nodes to judge other nodes according to their trustworthiness. The scheme was independent of the type of IDSs carried out at the node level. Moreover, it allowed nodes to utilize the feedback from more than one IDS in making their trust decisions. Nodes aggregated trust into a binary form (i.e., trust or distrust) and reported it to the KMS. The behavior grading information, the methods of authentication in the network, and possession of TCA or DCA certificate were recorded on a DCA certificate, thus providing nodes more information to aid them in their trust decision making. In addition, this information allowed the DCAs to dynamically adjust their revocation, reissuance, and security alert policies. Moreover, in the case of TCA certificates, it provided some indication of the trustworthiness of the certificate issuer, since the TCAs attached their DCA certificates to the certificates that they issued.

Unlike other schemes, this KMS utilized security alerts to complement the revocation methods by increasing their effectiveness in informing nodes of malicious behavior. The security alerts introduced more checks and balances into the network since nodes were made aware of the levels of malicious activity throughout the network lifetime.

The performance analysis of the system was investigated using a Monte Carlo and an NS2 simulation. The functionalities investigated were certificate issuance, certificate acquisition and revocation.

In the Monte Carlo analysis, the probability of certificate issuance with 10% DCAs increased by 18% to 51% (40 to 120 nodes; 100 meters radio range) as compared to a centralized system. In the NS2 simulation, certificate issuance with 10% DCAs increased by 14% to 35% for the various scenarios when compared to a centralized system.

In the Monte Carlo analysis, the existence of 10% TPs of a node (in addition to 10% DCAs) increased the availability for certificate acquisition between 36% to 61% (40 to 120 nodes) compared to a centralized system. In addition, the existence of 20% TPs increased availability between 49% to 66% (40 to 120 nodes). In the NS2 simulation, 10% TPs (20% DCAs and TPs), increased the probability of certificate acquisition

between 24% to 55%, and 20% TPs increased the probability of certificate availability between 25% to 61% compared to a centralized system.

In both simulation methods, the revocation analysis quantified the impact of network partitioning on revocation and emphasized the importance of security alerts in this KMS. In the Monte Carlo simulation, a DCA reached less than 26% of a node's TPs in a highly partitioned environment (40 nodes; 100 meters radio range). In the same partitioned environment in NS2, if revocation was using three SDCAs, the time to inform the TPs of a node with two RNs was between 200 to 420 seconds using the AODV protocol and 730 to 760 seconds with OLSR.

A comparison of this KMS with threshold cryptography schemes proved that threshold schemes provided lower availability for certificate issuance. In the Monte Carlo simulation, the threshold cryptography analysis showed that in a partitioned environment consisting of 40 nodes and a radio range of 100 meters, nodes only reached between 7% to 26% of the required number of CAs to obtain a certificate (assuming 4 CAs). With this KMS, access to 25% DCAs guaranteed service to the nodes. In the NS2 simulation, nodes required between 880 and 1130 seconds with the OLSR protocol and between 100 to 340 seconds with the AODV protocol to build a certificate (assuming four CAs). In this KMS, 10 to 80 seconds were sufficient to obtain a certificate.

This research has also provided an overhead analysis of this KMS with regards to the behavior grading of the KMS, certificate sizes, and certificate issuance and revocation. The behavior grading overhead fluctuated based on the security policies of the KMS and on the trust deviation between nodes in the network (e.g., number of TPs).

6.2. Recommendations for Future Work

The contribution of this research is the KMS framework and, more specifically, the unique way the various components that comprise the KMS interoperate to provide the desired functionalities. It is recommended that the following areas be investigated to further expand the state of the art.

- Investigate ways that input from two or more IDSs that utilize both host-based and network-based intrusion detection could be automatically aggregated into

a reputation index at the node level. The reputation index would be utilized by nodes to make their trust decisions, prior to reporting them to the KMS.

- Investigate integration of QoS and security policies of a KMS so that these two components interoperate seamlessly in a fashion that is beneficial for the security of the network. Due to the complexity of each system, current research has focused on the each of these topics separately. The integrated KMS-QoS mechanism should control the traffic flow within a network based on the behavior grading of the nodes and on the network characteristics, thus preventing degradation of performance of the network and minimizing damage.
- Develop a demonstration program that provides visual information about the activity of the nodes with regards to behavior grading. This mechanism could aid both the nodes and the KMS in making their security decisions.

6.3. Conclusions

This chapter has presented conclusions based upon the research results, and recommended areas of future research. The goal of this research was to provide a framework for key management that provides redundancy and robustness for SA establishment between pairs of nodes in MANETs. The KMS strikes a balance between the two extremes of centralized and anarchy PKI models by using a hierarchical trust model in which nodes can dynamically assume management roles. The system ensures high service availability for the network members in a highly partitioned environment through a number of schemes. A new behavior grading mechanism provides security criteria for the network nodes and aids the management functions of the KMS to revoke or reissue certificates for nodes.

The Monte Carlo and NS2 simulations demonstrated that the control plane of DCAs and TCAs with the utilization TPs significantly increased availability and aided SA establishment in a highly partitioned network environment. This KMS provided higher guarantees for issuing certificates to nodes compared to threshold cryptography schemes.

The results of this dissertation have demonstrated that the research goal was achieved. This work has been published in two papers [120][74]. In addition, it has been accepted at the International Journal of Information Technology for its special issue on Computer Security (invited).

Bibliography

[1] "IRTF RRG Ad hoc Network Scaling Research Subgroup," http://www.flarion.com/ans-research/, available July 4, 2003.

[2] L. Zhou and Z. Haas, "Securing Ad Hoc Networks," *IEEE Network*, Vol. 13, No. 6, pp. 24–30, Nov./Dec. 1999.

[3] S. Yi, R. Kravets, "MOCA: Mobile Certificate Authority for Wireless Ad Hoc Networks," *2nd Annual PKI Research Workshop Program (PKI 03)*, Apr. 2003

[4] M Bechler, H.-J. Hof, D. Kraft, F. Pahlke, and L. Wolf, "A Cluster-Based Security Architecture for Ad Hoc Networks," *IEEE INFOCOM*, Vol. 4, Mar. 2004, pp. 2393-2403.

[5] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad Hoc Networks," in *Proceedings of the 9th International Conference on Network Protocols (ICNP)*, Nov. 2001, pp. 251-260.

[6] S. Capkun, L. Buttyan, and J. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, Vol. 2, No. 1, pp. 52-64, Jan.-Mar. 2003.

[7] J.-P. Hubaux, Th. Gross, J.-Y. Le Boudec, and M. Vetterli, "Toward Self-Organized Mobile Ad Hoc Networks: The Terminodes Project," *IEEE Communications Magazine*, Vol. 39, No. 1, pp. 118-124, Jan. 2001.

[8] J.-P. Hubaux, L. Buttyan, and S. Capkun, "The Quest for Security in Mobile Ad Hoc Networks," in *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2001, pp. 146-155.

[9] S. Capkun, L. Buttyan, and J.-P. Hubaux, "Small Worlds in Security Systems: an Analysis of the PGP Certificate Graph," *ACM New Security Paradigm Workshop (NSPW)*, 2002, pp. 28-35.

[10] L. Buttyan and J.-P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, Vol. 8, No. 5, pp. 579-592, Oct. 2003.

[11] G. Montenegro and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses," in *Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS)*, Feb. 2002.

[12] G. O'Shea and M. Roe, "Child-proof Authentication for MIPv6 (CAM)," *ACM Computer Communications Review*, Vol. 31, No. 2, pp. 4-8, Apr. 2001.

[13] "Linux FreeS/WAN," http://www.FreeSWAN.org, available July 30, 2003.

[14] C. Bettstetter, "Mobility modeling in wireless networks: categorization, smooth movement, and border effects," *ACM Mobile Computing and Communications Review*, Vol. 5, No. 3, pp. 55-67, July 2001.

[15] C. Bettstetter, G. Resta, and P. Santi, "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, Vol. 2, No. 3, pp. 257-269, July-Sept. 2003.

[16] "The Network Simulator - ns-2," http://www.isi.edu/nsnam/ns/, available Feb. 21, 2005.

[17] "Wikipedia: The Free Encyclopedia," http://en.wikipedia.org/wiki/, available July 10, 2005.

[18] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*, 2nd ed. Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[19] C. Madson and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm with Explicit IV," *IETF RFC 2405*, Nov. 1998.

[20] "Advanced Encryption Standard," Federal Information Processing Standards (FIPS) 197, Nov. 2001. Available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[21] S. Frankel, R. Glenn, and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec" *IETF RFC 3602*, Sept. 2003.

[22] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, pp. 120-126, Feb. 1978.

[23] W. Diffie, M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp.644-654, Nov. 1976.

[24] S. Bellovin, J. Schiller, and C. Kaufman, "Security Mechanisms for the Internet," *IETF RFC 3631*, Dec. 2003.

[25] D. Atkins,W. Stallings, and P. Zimmermann, "PGP Message Exchange Formats," *IETF RFC 1991*, Aug. 1996.

[26] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures," *IETF RFC 1421*, Feb. 1993.

[27] T. Dierks and C. Allen, "The TLS Protocol," *IETF RFC2246*, Jan. 1999.

[28] Alan O. Freier, Philip Karlton,Paul C. Kocher, "The SSL Protocol, Version 3.0," *IETF draft*, draft-freier-ssl-version3-02.txt, Nov. 18, 1996.

[29] T. Ylonen, "The SSH (Secure Shell) Remote Login Protocol," *IETF draft*, draft-ylonen-ssh-protocol-00.txt, Nov. 15, 1995.

[30] C. Lonvick, "SSH Protocol Architecture," *IETF draft*, draft-ietf-secsh-architecture-22.txt, Mar. 2005.

[31] C. Lonvick, "SSH Authentication Protocol," *IETF draft*, draft-ietf-secsh-userauth-27.txt, Mar. 2005.

[32] C. Lonvick, "SSH Connection Protocol," *IETF draft*, draft-ietf-secsh-connect-25.txt, Mar. 2005.

[33] T. Ylonen and C. Lonvick, "SSH Transport Layer Protocol," *IETF draft*, draft-ietf-secsh-transport-24.txt, Mar. 14, 2005

[34] N. Doraswamy and D. Harkins, *IPsec The New Security Standard for the Internet, Intranets, and Virtual Private Networks*, Prentice Hall PTR, Upper Saddle River, NJ, 1999.

[35] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol," *IETF RFC 2401*, Nov. 1998.

[36] R. Rivest, "The MD5 Message-Digest Algorithm," *IETF RFC 1321*, *RSA Data Security, Inc.*, Apr. 1992.

[37] "What are MD2, MD4, and MD5?," http://www.rsasecurity.com/rsalabs/faq/3-6-6.html, available July 11, 2003.

[38] D. Piper, "The Internet IP Security Domain of Interpretation for ISAKMP," *IETF RFC 2407*, Nov. 1998.

[39] R. Perlman and C. Kaufman, "Key exchange in IPsec: analysis of IKE," *IEEE Internet Computing*, Vol. 4, No. 6, pp. 50-56, Nov.–Dec. 2000.

[40] S. Kent, R. Atkinson, "IP Encapsulating Security Payload (ESP)," *IETF RFC 2406*, Nov. 1998.

[41] S. Kent, R. Atkinson, "IP Authentication Header," *IETF RFC 2402*, Nov. 1998.

[42] S. M. Bellovin, "Problem areas for the IP Security Protocols," in *Proceedings of the Sixth Usenix Unix Security Symposium*, July 1996.

[43] N. Ferguson and B. Schneier, "A Cryptographic Evaluation of IPsec," July 11, 2003. Available at http://www.schneier.com/paper-ipsec.html

[44] G. C. Hadjichristofi, N. J. Davis, and S. F. Midkiff, "IPsec Overhead in Wireline and Wireless Networks for Web and Email Applications," in *Proceedings of the IEEE Performance, Computing, and Communications Conference (IPCCC)*, Apr. 2003, pp. 543-547.

[45] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," *IETF RFC 2104*, Feb. 1997.

[46] "What are SHA and SHA-1?," http://www.rsasecurity.com/rsalabs/faq/3-6-5.html, available July 11, 2003.

[47] H. Dobbertin, "The Status of MD5 After Recent Attack," *RSA Labs' CryptoBytes*, Vol. 2, No. 2, Summer 1996.

[48] C. Madson, R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH," *IETF RFC 2403*, Nov. 1998.

[49] "Has DES been broken?," http://www.rsasecurity.com/rsalabs/faq/3-2-2.html, available July 11, 2003.

[50] "What is triple-DES?," http://www.rsasecurity.com/rsalabs/faq/3-2-6.html, available July 11, 2003.

[51] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," *IETF RFC 2409*, Nov. 1998.

[52] P. Karn, “The Photuris Key Management Protocol,” *Internet draft*, (draft-karn-photuris-00.txt), Dec. 1994 (expired June 1995).

[53] G. Caronni, H. Lubich, A. Aziz, T. Markson, and R. Skrenta, “SKIP—Securing the Internet,” in *Proceedings of Fifth Workshop on Enabling Technologies* (WET-ICE), June 1996, pp. 62-67.

[54] D. Maughan, M. Schertler, M. Schneider, and J. Turner, “Internet Security Association and Key Management Protocol (ISAKMP),” *IETF RFC 2408*, Nov. 1998. Available at http://www.ietf.org/rfc/rfc2408.txt

[55] H. Orman, “The OAKLEY Key Determination Protocol,” *IETF RFC 2412*, Nov. 1998. Available at http://www.ietf.org/rfc/rfc2412.txt

[56] H. Krawczyk, "SKEME: a versatile secure key exchange mechanism for Internet," in *Proceedings of the Symposium on Network and Distributed System Security*, Feb. 1996, pp. 114-127.

[57] S. Kent, “Privacy Enhancement for Internet Electronic Mail:Part II: Certificate-Based Key Management,” *IETF RFC 1422*, Feb. 1993.

[58] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu, “Internet X.509 Public Key Infrastructure:Certificate Policy and Certification Practices Framework,” *IETF RFC 3647*, Nov. 2003.

[59] R. Housley, W. Polk, W. Ford, and D. Solo, “Internet X.509 Public Key Infrastructure:Certificate and Certificate Revocation List (CRL) Profile,” *IETF RFC 3280*, Apr. 2002.

[60] R. Perlman, "An overview of PKI trust models," *IEEE Network*, Vol. 13, No. 6, pp. 38-43, Nov.-Dec.1999.

[61] L. Zhou, F. B. Schneider, and R. V. Renesse, "COCA: A secure distributed online certification authority," *ACM Transactions on Computer Systems*, Vol. 20, No. 4, pp. 329-368, Nov. 2002.

[62] J. Douceur, “The Sybil Attack,” in Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS), 2002, pp. 251-260.

[63] S. Capkun, J.-P. Hubaux, and L. Buttyan, "Mobility Helps Security in Ad Hoc Networks," *in Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, June 2003, pp. 46-56.

[64] J. Myers, "Simple Authentication and Security Layer (SASL)," *IETF RFC 2222, Network Working Group,* Oct. 1997.

[65] W Sommerfeld, "Requirements for an IPsec API," *IETF draft*, draft-ietf-ipsp-ipsec-apireq-00.txt, June 17, 2003.

[66] T. Lin, S. F. Midkiff, and Jahng S. Park, "A Dynamic Topology Switch for the Emulation of Wireless Ad Hoc Networks Using a Wired Network," in *Proceedings of IEEE Local Computer Network (LCN) (Wireless Local Network Workshop)*, Nov. 2002, pp. 791-798.

[67] "ISC BIND," http://www.isc.org/products/BIND/, available July 20, 2003.

[68] K. Phanse and L. A. DaSilva, "Addressing the requirements of QoS management for wireless ad hoc networks," *Computer Communications,* Vol. 26, No. 12, pp. 1263-1273, July 2003.

[69] K. Phanse and L. A. DaSilva, "Protocol Support for Policy-Based Management of Mobile Ad Hoc Networks," *IEEE/IFIP Network Operations and Management Symposium (NOMS 2004)*, Apr. 2004, pp. 3-16.

[70] K. Phanse, L. A. DaSilva, and S. F. Midkiff, "Design and Demonstration of Policy-based Management in a Multi-hop Ad Hoc Network," *Journal of Ad Hoc Networks*, Vol. 3, No. 3, pp. 389-401, May 2005.

[71] T. Lin, "Mobile Ad-hoc Network Routing Protocols: Methodologies and Applications," Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Apr. 2004.

[72] T. Lin, S. F. Midkiff, and J. S. Park, "Approximation Algorithms for Minimal Connected Dominating Sets and Application with Routing protocol in Wireless Ad Hoc Network, " in *Proceedings of the IEEE International Performance Computing and Communications Conference (IPCCC)*, Apr. 2003, pp. 157-164.

[73] T. Lin, S. F. Midkiff, and J. S. Park, "A Framework for Mobile Ad Hoc Routing Protocols," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Vol. 2, Mar. 2003, pp. 1162-1167.

[74] L. A. DaSilva, S. F. Midkiff, J. S. Park, G. C. Hadjichristofi, K. Phanse, T. Lin, and N. J. Davis, "Network Mobility and Protocol Interoperability in Ad Hoc

Networks," *IEEE Communications Magazine*, Vol. 42, No.11, pp. 88-96, Nov. 2004.

[75] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements," *IETF draft*, draft-ietf-dnsext-dnssec-intro-05.txt, Oct. 2004.

[76] P. Hoffman, "IPsec Monitoring MIB," *IETF draft*, draft-ietf-ipsec-monitor-mib-06.txt, Apr. 2003.

[77] P Hoffman, "Internet Key Exchange (IKE) Monitoring MIB," *IETF draft*, draft-ietf-ipsec-ike-monitor-mib-04.txt, Apr. 2003.

[78] Y.-S. Wang and J. Touch, "Application deployment in virtual networks using the X-Bone," in *Proceedings DARPA Active NEtworks Conference and Exposition*, May 2002, pp. 484-491.

[79] J. Touch, "Dynamic Internet Overlay Deployment and Management Using the X-Bone," *Computer Networks*, July 2001, pp. 117-135.

[80] J. Touch, L. Eggert, and Y. Wang, "Use of IPsec Transport, Mode for Dynamic Routing", *IETF RFC 3884*, Sept. 2004.

[81] K. Seo and S. Kent, "Security Architecture for Internet Protocol," *IETF draft*, draft-ietf-ipsec-rfc2401bis-05.txt, Dec. 2004.

[82] M. Blaze, A. Keromytis and M. Richardson, and L. Sanchez, "IP Security Policy (IPSP) Requirements," *IETF RFC 3586*, Aug. 2003.

[83] M. Li, D. Arneson, A. Doria, J. Jason, C. Wang, and M. Stenberg, "IPsec Policy Information Base," *IETF draft*, draft-ietf-ipsp-ipsecpib-10.txt, Apr. 2004.

[84] J. Jason, L. Rafalow, and E. Vyncke, "IPsec Configuration Policy Information Model," *IETF RFC3585*, Aug. 2003.

[85] K. Channakeshava, K. S. Phanse, L. A. DaSilva, B. Ravindran, S. F. Midkiff, and E. D. Jensen, "IP Quality of Service for Soft Real-Time Applications," *Workshop on Real-Time Networks (RTN)*, July 5, 2005, to appear.

[86] "VeriSign," http://www.verisign.com, available May 10, 2005.

[87] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks," in *Proceedings Symposium on Network and Distributed System Security (NDSS)*, Feb. 2002.

[88] Y. Zhang and W. Lee, “Intrusion Detection in Wireless Ad Hoc Networks,” in *Proceedings of the Sixth International Conference on Mobile Computing and Networking (MobiCom 2000)*, Aug. 2000, pp. 275-283.

[89] S. Buchegger and J.-Y. Le Boudec, "Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks," in *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, 2002, pp. 403-410.

[90] P. Michiardi and R. Molva, “CORE: A Collaborative Reputation Mechanism to enforce cooperation in Mobile Ad-hoc networks,” in *Proceedings of the Sixth Joint Working Conference on Communications and Multimedia Security*, Sept. 2002. pp 107 –121.

[91] W. J. Adams, G. C Hadjichristofi, and N. J. Davis, “Calculating a node’s reputation in a mobile ad hoc network,” *in Proceedings of the IEEE Performance, Computing, and Communications Conference (IPCCC)*, Apr. 2005, pp. 303-307.

[92] S. Agarwal, D. Starobinski, and A. Trachtenberg, “On the Scalability of Data Synchronization Protocols for PDAs and Mobile Devices,” *IEEE Network*, Vol. 16, No. 4, pp. 22-28, July-Aug. 2002.

[93] B. Smith, S. Murthy, and J. Garcia-Luna-Aceves, “Securing Distance Vector Routing Protocols,” *Symposium on Network and Distributed Systems Security (NDSS ’97)*, Feb. 1997, pp. 85-92.

[94] S. Murphy and M. Badger, “Digital signature protection of the OSPF routing protocol,” in *Proceedings of the Symposium on Network and Distributed System Security (SNDSS ’96)*, Feb. 1996, pp. 93-102.

[95] S. Cheung, “An Efficient Message Authentication Scheme for Link State Routing,” *13th Annual Computer Security Applications Conference*, Dec. 1997, pp 90-98.

[96] M. G. Zapata, “Secure Ad hoc On-Demand Distance Vector (SAODV) Routing,” *IETF MANET Mailing List*, Message-ID:3BC17B40.BBF52E09@nokia.com, Available at ftp://manet.itd.nrl.navy.mil/pub/manet/2001-10.mail, Oct. 8, 2001.

[97] B. Dahill, B. Levine, E. Royer, and C. Shields, *A Secure Routing Protocol for Ad Hoc Networks*, Technical Report UM-CS-2001-037, Electrical Engineering and Computer Science, University of Michigan, Aug. 2001.

[98] S. Yi, P. Naldurg, and R. Kravets, "Security-Aware Ad hoc Routing for Wireless Networks," *Technical Report UIUCDCS-R-2001-2241*, Department of Computer Science, University of Illinois at Urbana-Champaign, Aug. 2001.

[99] P. Papadimitratos and Z. Haas, "Secure Routing for Mobile Ad Hoc Networks," SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), Jan. 2002.

[100] R. Jain, The Art of Computer Systems Performance Analysis, John Wiley and Sons Inc., NY, 1991.

[101] G. Guichal and C.-K. Toh, "An evaluation of centralized and distributed service location protocols for pervasive wireless networks,"" *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2001*, Vol. 2, Sept.-Oct. 2001, pp. E-55-E-61.

[102] "Dijkstra's Algorithm," http://mathworld.wolfram.com/DijkstrasAlgorithm.html, available July 6, 2005.

[103] "Algorithms for Computing the K Shortest Paths," http://terra.act.uji.es/REA/, available July 7, 2005

[104] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, 2002*, pp. 80-91.

[105] "Mersenne Twister Homepage," http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html, available Nov. 20, 2003.

[106] K. Pawlikowski, H.-D. J. Jeong, and J.-S. R. Lee, "On credibility of simulation studies of telecommunication networks," *IEEE Communications Magazine*, Vol. 40, No. 1, pp.132-139, Jan. 2002.

[107] "BonnMotion - A mobility scenario generation and analysis tool," http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/, available May 5, 2005.

[108] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, Vol. 2, No. 5, pp. 483-502, 2002.

[109] T. S. Rappaport, Wireless communications, principles and practice. Prentice Hall, 1996.

[110] R. V. Boppana and S. P. Konduru. "An adaptive distance vector routing algorithm mobile, ad hoc networks," *IEEE INFOCOM*, Vol. 3, Apr. 2001, pp. 1753-1762.

[111] G. Pei, M. Gerla, and X. Hong, "LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility," in *Proceedings of IEEE/ACM MobiHOC*, pp. 11-18, 2000.

[112] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-tree routing in wireless networks," in *Proceedings of IEEE International Conference on Network Protocols*, Oct.-Nov. 1999, pp. 273-282.

[113] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. "Scenario based performance analysis of routing protocols for mobile ad-hoc networks," in *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking*, Aug.1999, pp. 195-206.

[114] E. M. Royer and C. E. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking*, Aug. 1999, pp. 207-218.

[115] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," *IEEE INFOCOM*, Vol.2, Mar.-Apr. 2003, pp.1312-1321.

[116] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," *IETF draft*, Nov. 2002. Available at http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-12.txt.

[117] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot,"Optimized Link State Routing Protocol," *IETF draft*, Mar. 2002. Available at http://www.ietf.org/internet-drafts/draftietf-manet-olsr-06.txt.

[118] I. D. Aron and S. K. S. Gupta, “On the scalability of on-demand routing protocols for mobile ad hoc networks: an analytical study,” *Journal of Interconnection Networks*, Vol. 2, No. 1, pp. 5-29, 2001.

[119] A. Qayyum, L. Viennot, and A. Laouiti, “Multipoint relaying: An efficient technique for flooding in mobile wireless networks,” Research Report-3898, INRIA, France, 2000.

[120] G. C. Hadjichristofi, W. J. Adams, and N. J. Davis, "A Framework for Key Management in Mobile Ad Hoc Networks," in *Proceedings of the International Conference on Information Technology Coding and Computing*, Vol. 2, Apr. 2005, pp. 568-573.

[121] G. C. Hadjichristofi, W. J. Adams, and N. J. Davis, “A Framework for Key Management in Mobile Ad Hoc Networks,” accepted at the *International Journal of Information Technology*.

Vita

George Hadjichristofi was born on November 14, 1974 in Nicosia, Cyprus. After graduating from the American Academy High School in Larnaca, he served in the army for two years. He was then offered a Fulbright scholarship and moved to Blacksburg, Virginia to attend Virginia Tech. George Hadjichristofi received his B.S. and MS degree in computer engineering from Virginia Tech in 1999 and 2001, respectively. He joined the Ph.D. program in the spring of 2002. During his graduate studies, he served as a Graduate Teaching Assistant for one year and a Graduate Research Assistant for five years. His current research interests include wireless networks, ubiquitous computing, information assurance, and network performance evaluation and simulation.

George Hadjichristofi's publications include:

G. C. Hadjichristofi, N. J. Davis, and S. F. Midkiff, "IPsec Overhead in Wireline and Wireless Networks for Web and Email Applications," in *Proceedings of the 22nd IEEE Performance, Computing, and Communications Conference (IPCCC)*, Apr. 2003, pp. 543-547.

L. A. DaSilva, S. F. Midkiff, J. S. Park, G. C. Hadjichristofi, K. Phanse, T. Lin, and N. J. Davis, "Network Mobility and Protocol Interoperability in Ad Hoc Networks," IEEE Communications Magazine, Vol. 42, No. 11, pp. 88-96, Nov. 2004.

W. J. Adams, G. C Hadjichristofi, and N. J. Davis, "Calculating a Node's Reputation in a Mobile Ad Hoc Network," in *Proceedings of the 24th IEEE Performance, Computing, and Communications Conference (IPCCC)*, Apr. 2005, pp. 303-307.

G. C. Hadjichristofi, W. J. Adams, and N. J. Davis, "A Framework for Key Management in Mobile Ad Hoc Networks," in *Proceedings of the International Conference on Information Technology Coding and Computing*, Vol. 2, Apr. 2005, pp. 568-573.

G. C Hadjichristofi, W. J. Adams, and N. J. Davis, “A Framework for Key Management in Mobile Ad Hoc Networks,” accepted at the International Journal of Information Technology (IJIT).