

**Enhancing Self-Security in Wireless Adhoc
Networks using Multi-hop Authentication**

By

JAGADEESWARAN BHUVANESVARAN

Bachelor of Engineering

University of Madras

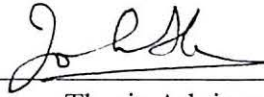
Chennai, India

2001

**Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
In partial fulfillment of
The requirements for
The Degree of
MASTER OF SCIENCE
May 2004**

Enhancing Self-Security in Wireless Adhoc
Networks using Multi-hop Authentication

Thesis Approved:



Thesis Advisor



Dean of the Graduate College

Preface

The Internet world is rapidly progressing from a wired era to a wireless era. New security schemes are designed for this transition. Fully Distributed Certification Authority (FDCA) is one such scheme that enables self-security in wireless Ad-hoc networks. FDCA distributes the responsibility of issuing certificates to a needy to all the nodes in the network thus improving availability. But this scheme demands several assumptions which are real cons to this scheme. One such assumption is that the node needing certificate should at least have a predefined constant number of nodes at any time, which is really not assured and not possible at all times. To enhance the FDCA scheme, a multi-hop authentication scheme has been proposed and analyzed. This scheme ensures availability and uses the bandwidth effectively.

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to Dr. Johnson Thomas for his assistance and able guidance through out my thesis in Oklahoma State University. I would also like to thank my committee members, Dr. Ajith Abraham and Dr. Venkatesh Sarangan, for serving in my committee. I am also very thankful to Dr. Debao Chen for his timely help.

I dedicate my work to my father Mr. Bhuvaneshvaran for his ever-lasting encouragement and emotional support throughout the years. I also thank my mother Mrs. Shanthi Bhuvaneshvaran and my sister Ms. Samundeeswari for their endless love and affection.

Finally I would like to thank my friend Mr. Ravichandran Raja for making it possible for me to finish my thesis work with his encouragement, help and support.

Table of Contents

Chapter		Page
1	Introduction	
1.1	Wireless Networks.....	1
1.2	Adhoc Networks.....	2
1.2.1	Applications of Adhoc Networks.....	3
1.3	Adhoc Network Security.....	4
1.3.1	Security Goals.....	4
1.3.2	Wireless security Threats, Challenges and Attacks.....	5
1.3.3	Security Attacks.....	7
1.3.4	Key Management.....	9
2	Background: Key Management	
2.1	Definition Overview.....	11
2.2	Asymmetric Encryption.....	12
2.3	Digital Signature.....	13
2.4	Digital Certificate.....	14
2.5	Certificate Authority.....	15
2.6	Public Key Infrastructure (PKI).....	16
2.6.1	How PKI works?.....	16
3	Key Management Schemes in Adhoc Networks	
3.1	Overview	19
3.2	Partially Distributed Certificate authority.....	21
3.2.1	System Description.....	22
3.2.2	System Maintenance.....	23
	Initialization.....	23
	Share Update.....	24
3.2.3	Certificate Issuing.....	26
3.2.4	Certificate Renewal.....	26
3.3	Fully Distributed Certificate Authority.....	28
3.3.1	System Description.....	28

3.3.2 System Maintenance.....	29
Initialization by the dealer.....	29
Self Initialization.....	27
Share Update.....	30
3.3.3 Certificate Retrieval/Renewal.....	30
3.3.4 Certificate Revocation and CRL.....	32

Chapter 4 Problems in Adhoc Key Management

4.1 Problem description.....	35
------------------------------	----

Chapter 5 Proposed Solution

5.1 Basic Scheme.....	37
5.2 System and Adversary Model.....	37
5.3 The Proposed Architecture.....	39
5.3.1 Localize Trust Model.....	39
5.3.2 Overview.....	41
5.4 Localized Certificate Services.....	42
5.4.1 Certificate Revocation List.....	48

Chapter 6 Simulation and Results

6.1 Performance Measure.....	51
6.2 Steps Involved in Simulation.....	51
6.3 Charts.....	52
6.4 Results.....	59

Chapter 7 Conclusions

7.1 Conclusions.....	60
7.2 Future Work.....	60

References.....	61
Appendix.....	63
1. Source Code.....	63

Chapter 1

Introduction

1.1 Wireless Networks

The success of wired computer networks has urged researchers to converge their interest towards the next level of technology research, Wireless networks. The history of wireless networks started in the 1970s and the interest has been growing ever since. During the last two decades, and especially lately, the interest has almost exploded probably because of the fast growing Internet. Today we have two kinds of wireless networks. The first kind and most used today is a wireless network built on-top of a "wired" network and thus creates a *reliable infrastructure wireless network*.

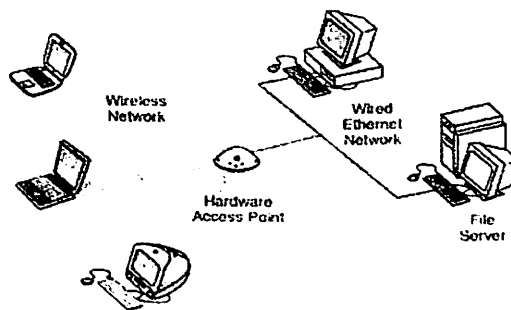


Figure 1 Wireless LAN.

As seen in the figure 1 the wireless nodes also connected to the wired network and able to act as bridges in a network of this kind are called base-stations. An example of this as seen in figure.1, there is one access point for three computers, two of them are laptops.

The person using the laptop can move around connected only within in the range of the wireless access point. Once the laptops are out of the range of the access point, they get disconnected from rest of the computers. Other more recent networks of this kind are cell phones. Once the cell phone gets out of the range of a tower, they hang-up and try to connect to the next tower nearby. These networks are also called *Wireless Local Area Networks* (WLAN).

The other kind of wireless network is one where there is no infrastructure at all except the participating mobile nodes. This is called an *infrastructures network* or more commonly an ad hoc network. The word "ad hoc" can be translated as "improvised" or "not organized". Hence the name Adhoc networks

1.2 Adhoc Networks

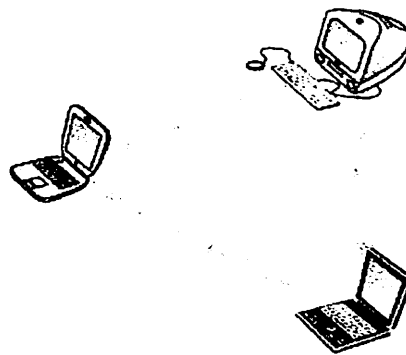


Figure 2 Adhoc Network.

Unlike traditional networks adhoc networks do not have fixed infrastructure or mobile switching centers. They are dynamic, peer-to-peer networks in which hosts rely upon each other to keep the network connected through wireless links and routing performed by mobile and fixed nodes.

As seen in the figure 2 an Adhoc network consists of a number of computers each equipped with a wireless networking interface card. Each computer can communicate directly with all of the other wireless enabled computers. They can share files and printers this way, but may not be able to access wired LAN resources, unless one of the computers acts as a bridge to the wired LAN using special software.

1.2.1 Applications of Adhoc Networks

Adhoc networks are suitable for certain applications. These include military engagements, disaster recovery operations and conferences. For military engagements, many military units are in constant motion and these units are vulnerable to destruction by enemies. Therefore military units would want to communicate using networks which do not have a fixed infrastructure and are very much mobile. Adhoc networks therefore are the ideal choice for this application.

In disaster recovery operations there is a high chance that an existing base station might be destroyed by natural calamities. Since communication is very vital in those situations, a baseless network with high mobility would be desirable.

Meetings and conferences may require data transfer between members independent of their locality. If a wired conventional network is used, a lengthy and troublesome setup process would be required. Adhoc networks can be of great use in such situations.

1.3 Adhoc Network Security

The growing commercial and military deployment of these networks has made security design increasingly important. But providing security in adhoc networks is challenging due to the fact that the mobile nodes are dynamic and largely autonomous.

1.3.1 Security goals

In trying to realize and incorporate more safety features into mobile networks, the following issues have to be addressed:-

- 1) **Availability:** Availability ensures that the network survives despite Denial of Service (DOS) attacks. On physical and media access control layer, an attacker can use jamming techniques to interfere with the communication on the physical channel. An attack on the network layer can disrupt the routing Protocol. Attacks on higher layers could bring down high level services such as the key management service.
- 2) **Confidentiality:** Confidentiality means no unnecessary information should be leaked out of the network and no unauthorized agents should be allowed within the network snooping around for information.
- 3) **Integrity:** For integrity to be maintained, the information is sent/received between mobile nodes has to maintain the same state as before it was sent/received, and is not corrupted.

- 4) **Authentication:** Authentication enables a node to ensure the identity of the peer node it is communicating with. Without authentication, an attacker would impersonate a node thus gaining unauthorized access to resources and sensitive information and interfering with the operation of other nodes
- 5) **Non-repudiation:** Non-repudiation ensures that the originator of a message cannot deny having sent the message. This would aid in isolating and locating of compromised nodes.
- 6) **Accountability:** This means that the actions of an entity must be traceable uniquely to that entity.

1.3.2 Wireless security Threats, Challenges and Attacks

Due to the fact that mobile nodes are largely autonomous, they are easy prey to being captured, or hijacked by possible viruses and worms. A case in example would be the Internet worm, Code Red, which spread rapidly to infect many of the Windows-based server machines. Many intra-networks run by companies rely on firewalls to fight off the worm. However, there were multiple incidents where the Code Red worm was caught from within the intra-network, and this was largely due to the fact that these networks also employed mobile computing, i.e. mobile nodes moving in and out of the intra-network topology, made them susceptible to the outside attacks.

Hence, mobile wireless networks are more prone to physical security threats than fixed line connections. These threats include eavesdropping, spoofing, and denial-of service (DoS) attacks. Nodes may also be hijacked by other malicious nodes. There are also cases of security breaches where the offending mobile node is unaware that it is sending out malicious codes to other nodes. Of course, other reasons for threatening the security of a network (mobile or fixed) could be due to employee sabotage (intra-corporation) or industrial espionage (inter-corporation). All of these represent potential threats in wireless networks.

These threats, if executed smoothly and successfully, can bring down an entire organization's system, and more importantly, place its data at risk. Sensitive information could be breached and disclosed to unintended organizations.

To summarize, wireless networks face the entire existing security problems faced by wired networks, as well as additional security threats that are unique to wireless networks. The following is a condensed list (but not limited to) of the salient threats and vulnerabilities of wireless networks:

- 1) An attacker would impersonate a node thus gaining unauthorized access to resources and sensitive information and interfering with the operation of other nodes

- 2) When a node is stolen or compromised it may cause Denial-of-service attacks by not forwarding packets to other nodes or not sharing neighboring node information.
- 3) It is easier to steal wireless devices like PDAs and hand phones. This is therefore a much greater chance of being stolen and personal information being revealed.
- 4) Malicious entities may launch attacks on other organizations through some wireless networks, thereby masking its identity.
- 5) Introduction of viruses and malicious codes into a wireless network.
- 6) Extraction of data without detection.
- 7) Interception of sensitive information through wireless networks that is not encrypted.

Some of the above listed threats and vulnerabilities are also in wired but these threats are much more difficult to deal with in wireless networks due to the changing topology and also due to availability of many attack points due to wireless nature.

1.3.3 Security Attacks

Attacks on networks can be classified under two large categories, namely active attacks & passive attacks. A passive attack would be one in which an authorized party simply gains access to an asset and does not modify its content, e.g.

eavesdropping. Passive attacks can be either simply eavesdropping or traffic analysis.

- 1) **Eavesdropping** – The attacker simply monitors transmissions for message content. An example of this attack is a person listening into the transmission on a LAN between two workstations or tuning into transmissions between a wireless handset and a base station.
- 2) **Traffic analysis** – The attacker gains intelligence by monitoring the transmissions for patterns of communication. A considerable amount of information is contained in the flow of messages between communicating parties.

An active attack is one whereby an unauthorized party makes modifications to a message, data stream or a file. Without a doubt, active attacks deal double the damage than their passive counterparts. Active attacks may take the form of one of four types (or a combination): masquerading, replay, message modification and DoS.

- 1) **Masquerading** – The attacker impersonates an authorized user and thereby gains access to some authorized privileges.
- 2) **Replay** – The attack monitors transmissions through traffic analysis and retransmits the message originally sent by as the legitimate user.
- 3) **Message modification** – The attacker alters a legitimate message by deleting, adding to, changing or reordering it.

4) **Denial of Service** – The attacker prevents or prohibits the normal use or management of resources such as communication facilities.

All risks against the IEEE Wireless 802.11 protocol are the result of one or more of these attacks. The consequences of these attacks include loss of proprietary information, legal and recovery costs, tarnished images, and loss of network service.

1.3.4 Key Management

Cryptographic schemes such as digital signatures are often employed to protect both routes and as well as data. . Digital signature is the electronic equivalent of a handwritten signature. It assures the recipient, that the data has not been altered or substituted. Digital signatures are explained in detail in the chapter 2. Every node in the network owns a public key and a private key pair. Public key is well known to all the nodes in the network and the private key is kept secret by the owner. Data encrypted using a public key can be decrypted only using the owner of the respective private key pair. This scheme is called as public key infrastructure. The public key infrastructure is studied in detail in next chapter.

A key management service or otherwise called as certification authority (CA) is a server node which issues the public-private key pair and the certificate, to other nodes in the network. A certificate is a piece of data which says that the node owning the private key is legitimate and trustworthy for a fixed time. A key management service managed by a centralized server is not a good idea for Adhoc networks because, the sever (certification providing authority - CA) may not be available to issue the private-public key pair to other nodes to establish a secure

connection. Moreover if the CA is compromised, the attacker can sign any erroneous certificates with the private key. Naive replication of CA to increase the availability of CAs can make the network more vulnerable because compromising of a single CA can cause the whole system to fail. One approach to solving the single CA problem is to distribute the trust to a set of nodes by letting these nodes share the key management responsibility. This distribution is achieved using a technique called threshold cryptography which is detailed in chapter 2. However, threshold cryptography may result in an unacceptable number of nodes not obtaining a certificate. The objective of our work is to increase the success rate of obtaining certificates by proposing a novel multi-hop threshold certificate scheme. To increase the success rate, we introduce the multi-hop authentication technique, whereby we not only request the neighboring nodes for partial keys, but also nodes request nodes at multiple hops.

The organization of rest of this thesis is as follows. In Chapter 2, background information on Key Management is provided. Chapter 3 describes the two important key management schemes in Adhoc networks. The typical problems in existing Key management scheme in adhoc network are discussed in Chapter 4. In Chapter 5, the description of the proposed enhancement to an existing key management scheme is given. Chapter 6 evaluates the performance of the proposed approach, and in the last Chapter the thesis is concluded with pointers for future research.

Chapter 2

Background: Key Management

2.1 Definition and Overview

The purpose of Key establishment is to create a common key for a group of two or more participants to be used for encryption and authentication of their communications. Key Management can be defined as “*The set of technique and procedures supporting the establishment and maintenance of keying relationship between authorized parties*” [1]. In the above definition the *keying relationship* can be defined as a state where communicating entities share common data to facilitate cryptographic techniques.

One aspect of key establishment protocol is key distribution, which means that the key is generated by one party and distributed to other participants. Key management contains techniques and procedures supporting the following,

- Initialization of system users within domain
- Generation, distribution and installation of keys.
- Controlling the use of keys.
- Update, revocation and destruction of keys.
- Storage, backup/recovery and archival of keys

The most popular key management scheme presently used is Public key infrastructure (PKI). Before delving in detail of PKI, the cryptography techniques used in PKI are described below.

2.2 Asymmetric Encryption

An asymmetric encryption system is a system involving two related transformations – one defined by a public key (the public transformation) based on RSA cryptography developed at MIT in 1970, and another defined by a private key (the private transformation). The essential property of asymmetric encryption is that it is computationally infeasible to determine the private key from the public key. Public key are used for encrypting data and Private Key is used to decrypt it and vice versa. Essentially, with a public key pk_{user} and private key sk_{user} , the plain text message (m) can be encrypted to obtain the cipher text where E stands for encryption and D stands for decryption.

$$c = E_{pk_{user}}(m) \text{ and } m = D_{sk_{user}}(c).$$

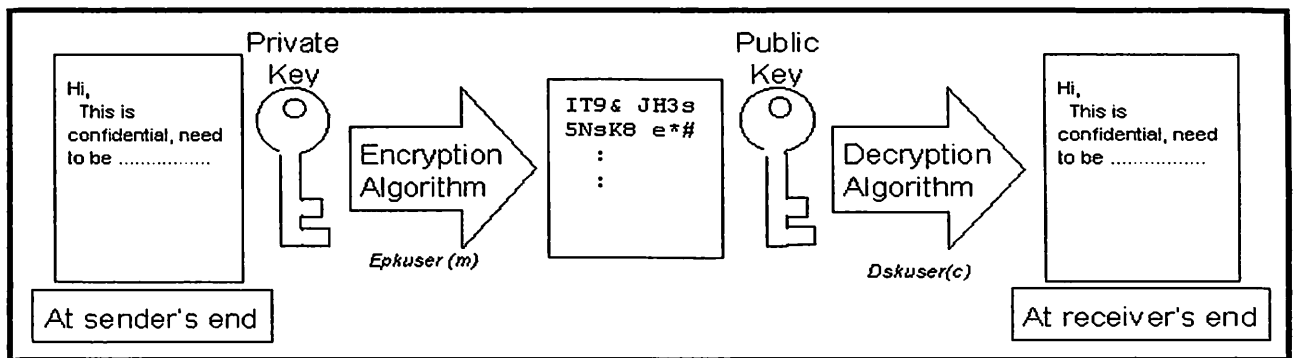


Figure 3 Asymmetric encryption

In the figure 3 $E_{pk_{user}}()$ and $D_{sk_{user}}()$ are the encryption and decryption functions. Some of the popular public key algorithms are Diffie-Hellman and RSA. The purpose of developing this technique was to address the problems of using one

key for both encryption and decryption, as it is done in the more traditional symmetric encryption.

2.3 Digital Signature

Digital signatures are very important part of digital certificates. A digital certificate is a statement issued by a trusted party that verifies the ownership of a public key to that of a user. The trusted party digitally signs this statement and therefore anyone with the authentic public key of that trusted party can verify the certificate. Digital signature is the electronic equivalent of a handwritten signature. It assures that the recipient that the data has not been altered or substituted. Digital signatures use public key cryptography: in its simplest form the sender encrypts the message with the sender's private key. The receiver decrypts the message with the sender's public key. The receiver is therefore confident that the sender is who he claims to be, since only that particular sender has the private key to encrypt the message. Since the private key is used for encryption, the signature will be different for each user, and the message cannot be forged.

In a typical scheme, there are two steps that must take place in order to obtain a digital signature:

1. A *hash* is created from the data (for example, a message) being sent. It is a sequence of 0's and 1's and commonly has a length of 128 bits. This hash cannot be reverse-engineered; that is, it cannot be used to determine what the original data was.

2. The hash is then encrypted using the author's private key. The result is the *digital signature*.

The recipient decrypts the digital signature using the specific author's public key and checks to verify that the message creates the same hash. A different hash indicates that the message has been tampered with.

2.4 Digital Certificates

A digital certificate is a statement issued by a trusted party that verifies the owner of a public key is that user. The trusted party digitally signs this statement and therefore anyone with the authentic public key of that trusted party can verify the certificate and therefore, use the public key therein and be sufficiently sure that it belongs to the owner mentioned in the certificate. A certificate always contains at least three pieces of information: the name of the owner, the public key, and a digital signature computed over those other two.

Certificates can use any digital signature mechanism, although most commercial system uses the RSA algorithm. Without the signature to check, an attacker could substitute one public key for another and masquerade as one person or another in a data exchange using public keys. Any end entity must have a copy of CA's public key in order to check a certificate's digital signature.

Digital certificates also carry additional information, which are optionally included for extra security checkup like publishing date, data of revocation, user certificate serial number etc. These help to detect forgery and prevent active attacks.

The purpose of the digital certificate is to state that the information on the certificate has been attested by another entity. This attestation helps to establish the accuracy and authenticity of the public key, therefore by implication the authentication and confidentiality of any message that can be verified with this public key.

An important thing to remember about digital certificates is that they verify the identity of the person you are dealing with, in the other words, the certificate confirms that the person is who he claims to be. But it does not provide any assurance about the integrity of the person or that the transaction that you make with someone with a digital certificate will be honored.

There are four components to a generic certificate [2]:

- 1) The owner's identity.
- 2) The owner's public key.
- 3) Certification Authority (CA) information: the certification authority, date of issue, expiry, etc.
- 4) The digital signature, obtained by hashing the previous three components and the CA's private key.

The certificate can be verified by using the CA's public key.

2.5 Certificate Authorities (CA)

To simplify the idea of digital certificates, one can think of them as passports. Like passports, they identify a person, as well as encode other personal information.

Additionally, there are security features to aid in the prevention of forgery. Consequently, *Certificate Authorities* are comparable to passport offices since they are the organizations that verify the identity of an individual and provide them with a unique certificate.

2.6 Public Key Infrastructure(PKI)

PKI is the combination of various technologies, infrastructure and practices needed to enable the use of public key encryption and digital signatures in distributed applications on a significant scale. The main purpose of PKI is to distribute keys accurately and reliably from the certificate authority to those who need the public-private key pair. PKI also covers certificate renewal, revocation, status checking and private key backup and recovery.

PKI performs the following three basic functions:

1. **Authentication:** It validates the identity of the parties in communication and transactions
2. **Confidentiality:** It ensures that information, even if intercepted, cannot be unused by unintended recipients.
3. **Non-repudiation:** It ensures that transaction once committed is binding and irrevocable.

2.6.1 How PKI works?

When two parties want to communicate over internet or network, using PKI ensure the confidentiality and security of their transaction. Both should have digital

certificate. To obtain a certificate, both parties have to request to the certification authority. The certification Authority issues a digital certificate to both parties using the public key infrastructure. The certification contains the details such as the public key in the certificate's expiration date and CA's digital signature. Both parties also get the private key corresponding to their public key from the CA.

Now both parties can send information to each other very securely. When one party wants to send a message to the other, the message is encrypted using the other party's public key and uses the private key of sender to create digital signature. When the receiver receives the message, it is decrypted using the receiver's private key and verified using the digital signature. The following table illustrates how PKI works,

Intension	Action
Sender want to send a secure message over internet	Sender creates Digital signature using this private key.
Sender want to make sure no other than intended receiver should read to message	Sender encrypts the message using receivers public key
Receiver want to read the message	Decrypts using his private key and reads
Receiver wants to verify that the message was not modified during transmission.	Uses the public key of sender to verify the sender's digital signature.

In the next chapter we explain the Key management schemes presently proposed for Adhoc networks.

Chapter 3

Key Management Schemes in Adhoc Networks

3.1 Overview

Popular network authentication architectures using a centralized Certificate Authority (CA) work fine in a wired networks, but are not suitable for ad hoc networks. Nodes in an adhoc network cannot rely on any infrastructure, either in terms of routing, servers or organizational support. Therefore ad hoc networks cannot rely on the availability of one particular central server. If the CA becomes unavailable, nodes cannot retrieve the public keys of other nodes in order to establish secure communications and the system is compromised.

Several protocols have already been proposed to implement key mechanisms in ad hoc networks. Some consider distributing the functionality of the centralized CA server among a group of servers, so that availability could be improved by avoiding a single point of failure whereas other prefer avoiding the use of a centralized authority by letting the nodes organize themselves

When distributing the CA authority to a group of servers, a simple replication of the central server would make the system even more vulnerable as an attacker would only need to compromise one of these servers. To cope with this issue, security functions' sharing has been a very active research area in cryptography.

Threshold cryptography sharing [4] serves as a basic primitive because it allows securely sharing the system private key. The concept of proactive secret sharing is also used to further improve the robustness of the threshold secret sharing by periodically updating the secret shares.

Some other protocols prefer, on the other hand, establishing a secure key exchange without using either any certificates or authority. The idea is therefore to let the users organize themselves and derive a key session from a password or some other ways to ensure authentication. We do not discuss these schemes and protocols below.

We will discuss in detail about two of the most important certificate authority schemes related to this thesis below. These schemes efficiently distribute the Certificate Authority function to mobile nodes in the network. Under the techniques used in these schemes, the encryption key (private key of the Certificate authority) is divided into several parts and distributed to nodes of the network. If K such shares of the encryption key can be acquired, the original key can be reconstructed. This concept of dividing the key among the mobile nodes is known as '*Threshold Cryptography*'. But when a node constructs the private key using k share of such acquired shares, there is a high risk that the node may be compromised, which breaks the whole network's security.

A simple solution to this problem is to not reconstruct the private key by acquiring k shares, instead whenever a certificate is needed to be generated each secret key holder generates a partial certificate over the data. By collecting k such partial signatures or certificate the node that needs to be authenticated generates a full signature

or certificate. This partial signature generated by combining k shares is known as '*Threshold Digital signature*'.

Applying Threshold digital signature to the public key infrastructure (PKI) in adhoc networks presents a solution to the key management problem by dividing the certificate authority's functionality among n mobile nodes. Each of the nodes carries a partial share of the CA's private key. Whenever a client requires obtaining a certificate from the CA, it must contact all the nodes in its neighborhood. If at least K of these neighboring nodes reply with the partial certificates, then the partial certificates are combined to form a full certificate.

There are several Certificate authority schemes proposed for adhoc network. The most important of them which are related to this thesis are

- 1) Partially Distributed Certificate Authority scheme.
- 2) Fully Distributed Certificate Authority scheme.

3.2 PARTIALLY DISTRIBUTED CERTIFICATION AUTHORITY

This solution has been proposed by Zhou and Haas [5] and then by Yi and Kravets in two papers [6] [7]. Their solution is to distribute the services of a certificate authority to a set of nodes by letting them share the key management responsibility through the use of threshold cryptography.

3.2.1 System Description

In this solution, the system as shown in the figure 4 contains four types of nodes: dealer, clients, servers and combiners, each of them having a public/private key pair.

1. *Dealer*: The administrative authority responsible for initializing the network.
2. *Client*: A normal node in the ad hoc network, which queries server nodes for certificates.
3. *Server*: CA nodes, which store certificates in a directory structure and generate partial certificates in response to client requests.
4. *Combiner*: These are CA nodes too, responsible for combining the partial certificates into valid ones.

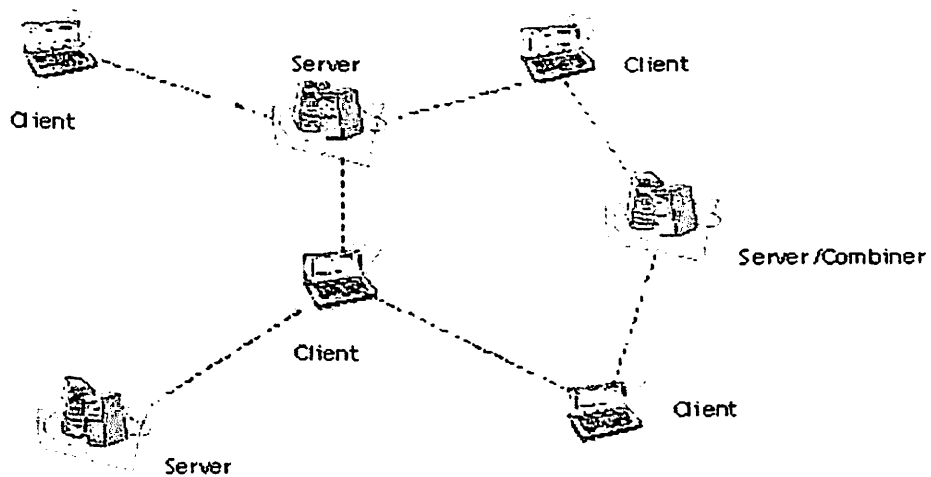


Figure 4 Partially Distributed Certificate Authority

The certificate authority service also has a public/private key pair. All nodes in the system know the public key of the service and trust any certificate signed by the corresponding private key. This corresponding private key, which is the certificate signing key sk_{CA} , is divided into n shares ($s_1, s_2 \dots s_n$), using Shamir's secret sharing scheme, that are then distributed to a number of particular nodes called servers (or MOCA standing for MOBILE Certificate Authority). These servers know the public keys of all nodes in the network and store their corresponding certificates. They are also responsible for generating partial signatures for these certificates using their share of the signing key.

Finally, the combiner nodes are responsible for generating a valid certificate using the different partial certificates generated. (Note: combiners are one of the solutions proposed by Zhou and Haas. In the other implementation, clients combine themselves the partial certificates).

3.2.2 System Maintenance

Initialization

Distribution of trust in this solution is accomplished by the use of a (n, k) threshold scheme, which means that the certificate signing key sk_{CA} is divided into n shares and that only k nodes (with $k < n$) are required to jointly perform the signing operation. k can be chosen between 1 and n . Setting k to a higher value has the effect of making the system more secure against possible adversaries since k is the number of MOCA's an adversary needs to compromise to break the system. At the same time, a high

value of k can cause more communication overhead for clients as they need to contact at least k servers to get certification services.

One of the other major challenges is to choose which nodes will be part of the distributed CA. In general, this set of nodes could be randomly chosen from the nodes in the network, but with such an approach, it is possible that the chosen nodes would not be able to perform correctly their tasks as nodes generally have heterogeneous capabilities. It is then better that the nodes chosen to be part of the CA should have a high level of security, jointly with high processor or memory resources.

The system has an administrative authority that plays the role of a dealer. It is the only entity that has the knowledge of the complete certificate signing key and is in charge of distributing their shares to the n MOCA during the bootstrapping of the network. It also provides offline the nodes with their own certificates and the CA's certificate (public key). Then, the nodes are independent, in particular for the share update that we are going to cover in the following part.

Share Update

At periodic intervals the servers update their shares of the CA's private key. A proactive threshold cryptography scheme uses share refreshing, which enables servers to compute new shares from old ones without disclosing the service private key to any server. After refreshing, the servers remove the old shares and use the new ones to generate partial signatures. Because old and new shares are independent, an attacker has to compromise k servers within a short amount of time to break the whole system.

Given n servers, let (s_1, s_2, \dots, s_n) be an (n, k) threshold scheme sharing the private key sk_{CA} of the system, each server i having his share s_i . The share refreshing proceeds as follows: first, each server randomly generates a (n, k) sharing of 0 $(s_{i1}, s_{i2}, \dots, s_{in})$ and distributes every sub share s_{ij} to server j through a secure link. When a server j gets the sub shares $(s_{1j}, s_{2j}, \dots, s_{nj})$, it can compute a new share from these subshares and his old share $(s'_j = s_j + \sum_{i=1}^n s_{ij})$.

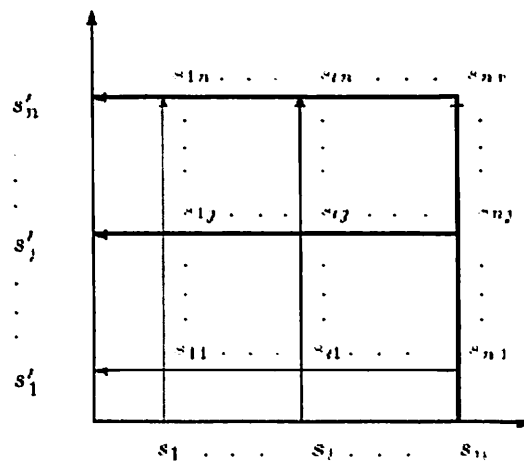


Figure.5.Share update mechanism

The figure.5 illustrates the share update mechanism. Each server i on the X axis generate n sub shares which constitute the i th column in the figure. Then, each server j on the Y axis adds all the subshares, which constitute the j th row, with its old one to compute its new one.

Share refreshing must tolerate missing subshares or erroneous subshares from compromised servers. A compromised server may ever send erroneous subshares in order to generate invalid shares, or not send any subshare at all. However, nodes can generate

new shares using only k subshares generated by the other servers. Furthermore, for servers to detect incorrect subshares, a verifiable secret sharing scheme is used. This scheme generates extra public information for each subshare using a one way function that can help to testify the correctness of the share.

3.2.3 Certificate issuing

An administrative authority provides offline any node that would like to enter the network with its own certificate as well as the CA's certificate. The node's certificate is thereafter stored in all the server's directories to be available to all the nodes in the network. The servers must also have a mechanism of synchronizing their certificate directories in the case of update and renewal.

3.2.4 Certificate Renewal

The certificates are only valid for a certain amount of time and therefore need to be renewed before they expire. When a node wishes to renew its certificate, it must request a certificate renewal to a minimum of k servers. If the request is granted, each of these k servers generates a partial certificate with a new expiration date. These partial certificates are then sent to a combiner that creates the complete certificate and then sends it to the servers for them to update their directories. If any of the servers is compromised, the partial certificate this server sends will be erroneous and this leads to a wrong new certificate. This type of denial of service attack is prevented by verifying the partial certificate before accepting it. If the combiner detects an invalid partial certificate, it will ask for another set of partial certificates until a valid certificate is obtained.

3.2.5 Limitations of the Partially distributed Certificate Authority

The Partially Distributed Certificate Authority is a good attempt at solving the problem of minimizing the chances of the CA being compromised. However, it is not without drawbacks and deficiencies.

- This solution requires a server and administrative infrastructure and so is applicable to only a subset of ad hoc networks, i.e. planned, long-term networks.
- It is assumed that a subset of nodes is willing to take on a specialized server role.
- As it is based on public key encryption, it requires that all nodes are capable of performing the necessary computations. This might be a stringent requirement as most mobile nodes are expected to have limited capabilities.
- The lack of a certificate revocation mechanism is the most critical functional drawback. Hence, in the case of a compromise, this system would have no suitable scheme to deal with security break-ins.
- The issue of routing to server nodes has not been addressed. Since the client node is interested in locating any k server nodes, the CA could be given a multicast address.
- A network split may occur during the system update phase, causing the two disjoint segments to update their shares independently of each other. Later, in the case of a re-merger, the servers would have inconsistent shares. Therefore, a mechanism must be in place to handle this situation.

3.3 FULLY DISTRIBUTED CERTIFICATION AUTHORITY

This solution is proposed by Luo and Lu in [8] [9]. It also uses a (k,n) threshold scheme to distribute the certification authority, but compared to the previous solution, the authority is shared by all nodes in the network. Each individual can then take part in the certification process and no differentiation is made between client and server, all nodes play the same role and can potentially provide other nodes certification services. Here as well, proactive secret sharing mechanisms are used to protect the system from being compromised.

3.3.1 System Description

In this architecture also, the certification system as a whole has a public/private key pair. The public key is assumed to be known by everybody in the network and the corresponding certificate-signing key is shared among all the nodes using Threshold cryptography scheme.

Each node is provided with its own certificate, so that nodes without valid certificates are treated as adversaries and denied of any network resource. Upon entering a new area, one node exchanges its certificate with all his one-hop neighbors who store it in their directories, either in the main part or in their Certificate Revocation List (CRL) if the certificate happens to be invalid. Then, every node controls and monitors his one-hop neighbors' certificates.

A node desiring to obtain another node's certificate needs to send a certification request to his one-hop neighbors and waits until it receives at least k answers. Then, after receiving k partial certificates, the node can combine them in order to get the target's authenticated certificate. The availability of the service is based on the assumption that a node has a least k one-hop neighbors at any time.

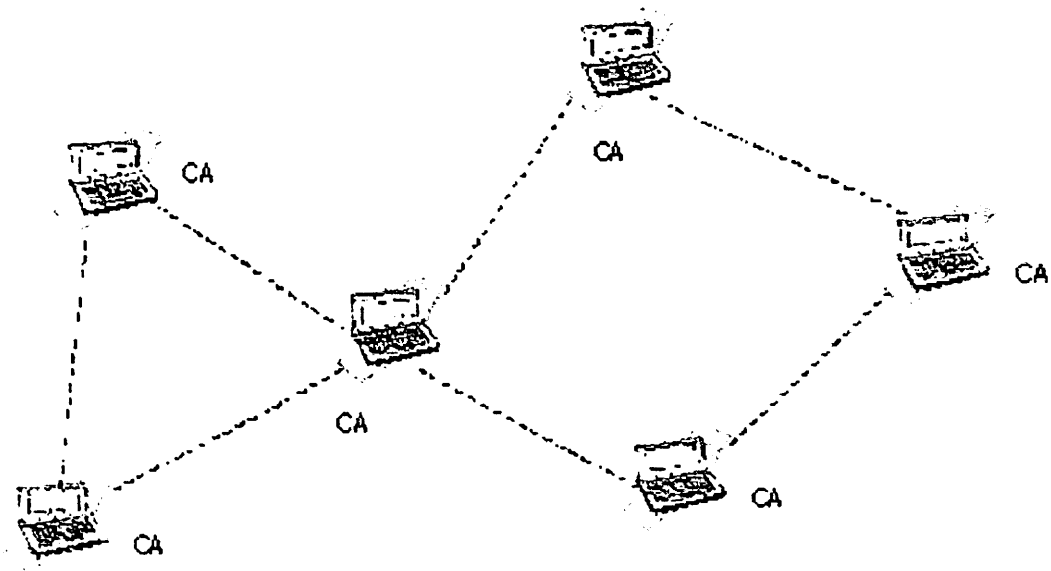


Figure.6. Fully distributed certificate authority

Figure.6. shows the basic architecture of fully distributed Certificate Authority scheme. All the nodes in this scheme share the Private Key of the certification authority.

3.3.2 System Maintenance

Initialization by the dealer

During the initial phase, the administrative authority, the dealer, provides the k first nodes with their own valid certificate, the system's public key as well as their share

of the system private key according to a polynomial of order $k-1$. Since the dealer is the only entity that knows the entire system private key, maintaining a dealer online to provide future nodes with the same parameters would compromise the system, as the dealer would become the only point of failure. After this initialization phase, the dealer simply quits the network and lets the initial nodes handle the next nodes initialization.

Self initialization

Any node joining the network after the dealer has left must obtain all the parameters detailed above, such as his share of the certificate signing key. However, an assumption is made that a node has already been given a certificate. The initialized nodes collaboratively initialize other nodes, typically their neighbors. As more and more nodes become initialized, all nodes in the network tend to obtain their share of the signing key.

If v_i is not initialized yet, it locates a coalition of nodes among its neighbors and broadcasts a request for initialization with the ID of these nodes. Once a node v_j receives such a request from a node v_i , it checks v_i 's certificate to know whether it's valid. If it decides to serve the request (certificate not in the CRL), it computes a partial share P_j for v_i . By combining k partial shares, the node is initialized with its valid share.

However, it is insecure for a node v_j to return directly P_j to the node v_i , because v_i could recover v_j 's share from the partial share P_j . Then, v_i could recover k partial shares and thereafter recover the whole system signing key. To protect such threat, the k nodes shuffle their partial shares before sending them to v_i , according to a shuffling factor previously established between a pair of nodes.

Share Update

As defined in the previous algorithms, proactive secret sharing is used to update the signing key shares. This prevents an attacker from discovering enough (k) partial signing shares to compromise the whole system, by constraining him to discover them within a short period of time. This update is done by distributing a new polynomial to the nodes,

$$f_{\text{update}}(x) = b_1x + b_2x^2 + \dots + b_{k-1}x^{k-1}$$

Whose coefficients are then encrypted, signed and flooded in the network. Each node then receives $\{E_{pk_{CA}}(b_1), \dots, E_{pk_{CA}}(b_{k-1})\}$ that it can verify using the system's public key. Then it computes the new share $S_p = f_{\text{update}}(id_p)$.

3.3.3 Certificate Retrieval / Renewal

Each node in the network holds a partial share of the system's signing key SK according to a random polynomial. It is now able to sign partial certificates to users. A certificate has a given validity period, and needs to be renewed before it expires.

The certification retrieval/renewal mechanism is the same as for self-initialization. For a node v_i to renew its own certificate or to retrieve any other node's certificate, it contacts a coalition of nodes among its one-hop neighbors and waits for k responses, hence k partial certificates. A node v_j answering the request first checks the requesting node v_i 's current certificate. If it is not a convicted node in its CRL (certificate already revoked) or if the certificate is not yet expired, v_j accepts to serve the request and

generates a partial certificate using his share of the system's private key before returning it to v_i .

$Certi = (cert)_{s_i} \bmod N$, where S_i is the node's signing key share

Upon receiving at least k partial certificates, the requesting node v then combines them to generate the certificate and verifies it by checking $cert = (CERT)_{PK} \bmod N$ (PK being the system's known public key).

This scheme works with any of k nodes within the one-hop neighbors, so that some of them can fail to answer without making the whole process fail. Moreover, each partial certificate is verified upon arrival, so that a compromised node cannot have any influence on the certification process.

3.3.3 Certificate Revocation and CRL

The certification revocation mechanism is based on the assumption that all nodes monitor their neighbors' certificates and behaviors. Revocation is conducted either if a certificate has expired or if a node has had a compromised behavior. In such cases, a node stores the given certificate in the Certification Revocation List with a "convicted" accusation and forwards the information to its neighbors who then consider the node as "suspect". These neighbors first check the accuser's certificate validity and store the suspected certificate in their respective CRL's as a "convicted" node as soon as they receive a coalition of k "suspicion" accusation.

The accusation should be propagated fast enough to prevent a node v_i from roaming to get certification services (hence renewal) from other nodes who don't know yet about it being compromised. However, as certificates have an expiry date, it is useless to forward the accusation for a longer period than the actual certificate's validity. In other words, after a certain amount of time, as a certificate will anyway be invalid, so it becomes meaningless to forward accusation packets about this particular certificate. The solution is then to set the Transmitting time level (TTL) of the IP packets containing the accusation based on the certificate validity period T_{cert} , the one-hop wireless transmission period D and the assumption on maximum node moving speed S_{max} . $TTL \geq \lceil T_{cert} \cdot 2S_{max} \rceil / D$.

3.3.4 Limitations of the fully distributed Certificate Authority

This solution is an improvement upon the Partially Distributed Certificate Authority in that it increases availability, doesn't require any special roles, and, unlike the other certificate-based solutions proposed, provides a certificate revocation mechanism. However, it is not without its drawbacks.

- As with the previous schemes, this solution, too, is aimed towards planned, long-term ad hoc networks capable of public key encryption.
- The cost of achieving high availability is a set of rather complex maintenance protocols like the share initialization and share update protocols.
- Since every node has its own share, a large number of shares are exposed to compromise. The threshold parameter, k , therefore, may have to be chosen larger. This, in turn, affects the availability of the service.

- The certificate revocation mechanism assumes that each node is willing to monitor the behavior of all its one-hop neighbors, which maybe too strong an assumption in certain adhoc networks.
- Again, in the case of a network split during system update, inconsistencies may arise. The system must, therefore, provide a synchronization mechanism.

The problems caused by these limitations described above and our proposed solution is presented in next two chapters.

Chapter 4

Problems in Adhoc Key Management Schemes

4.1 Problem Description

The limitations of the fully distributed certificate Authority are identified in the chapter 4. One of these limitations is the problem we address in this thesis. “Since every node has its own share, a large number of shares are exposed to compromise. The threshold parameter, k , therefore, may have to be chosen larger”. If the threshold parameter increases, the availability becomes a problem, as the success rate for reconstructing a certificate will decrease. Assume a war zone or a natural calamity, the mobile nodes are under high risk of being destroyed or compromised. A node which needs service for authentication cannot in all situations find k neighboring nodes. The existing authentication schemes always assume that the availability of the neighboring nodes is always a constant number and that the nodes are always readily available in its one hop neighborhood for service.

There may be situations when the node which needs authentication may not have k one hop neighbors available at the time. There would be the situation when the node is stuck in one geographical position and there are not many neighboring nodes. In the worst case there would be only one one-hop neighbor, in this case the nodes will have to trust only mobility and wait until it meets k one-hop nodes arrive in the neighborhood. There may even be situations when almost all the neighboring nodes are compromised and a denial of service attack exists.

Our proposal modifies the fully distributed certificate authority scheme by contacting multi-hop nodes every time it needs authentication and stores the information in a table called K-hop information table. This table stores the hop numbers and information about the nodes in each hop. When authentication is required, it contacts k nodes in multiple hops instead of just the one hop neighbors. This approach allows certificates to be constructed even in the presence of Denial of Service attacks as it ensures availability of K nodes needed for issuing partial certificate. Our approach therefore not only improves the success ratio, but it also increases the survivability of the system. The proposed solution is elaborated in chapter 5.

Chapter 5

Proposed Solution

5.1 Basic Scheme

The proposed scheme uses a Multi-Hop authentication technique for Key Management. This scheme, is an improvement on the fully distributed certificate authority [8], based on a (k, n) threshold scheme, and distributes the certificate authority service among all nodes in the network. In an attempt to improve the availability and work efficiently against DoS attacks, we have modified fully distribute certificate authority scheme proposed by H. Luo et al to use multi-hop nodes for getting k partial certificates instead of using only one-hop neighboring nodes to obtain the k partial certificates.

5.2 System & Adversary Models

This modified fully Distributed certificate authority scheme like the original scheme has mobile nodes which communicate with each other using bandwidth constrained, error prone and insecure wireless channels. We assume that the network has n mobile nodes where number n is not static and may vary due to several reasons such as new nodes joining the network, node leaving or compromised (attacked) over time. The reliability of the multi-hop packet forwarding is based on the underlying transport layer.

The other assumptions are the same as in currently existing scheme [8],

- Each node has a unique non-zero id number to identify itself to the other nodes or neighboring nodes.

- Mobility is characterized by a maximum node's maximum moving speed which is S_{max} .
- To get authenticated each node requires at least k nodes in the neighboring or in the reachable multiple hops.
- Each node is equipped with some local detection mechanism to identify the misbehaving nodes among its one hop neighborhood nodes. Some of these techniques are proposed in [10] [11].

Intrusion detection in Wireless Adhoc networks is more difficult than in the wired conventional networks [10]. The technique proposed by [11] shows that detecting intruders and misbehavior in one-hop neighborhoods is much easier than detecting intruders and misbehaviors in multi-hop nodes.

Like the existing scheme, our proposed scheme handles two kinds of attacks, DoS attacks and node Break-ins [8]. Adversaries may issue the DoS attacks from various layers of the network. The other attacks that are handled include *smurf*, *teardrop*, *transport layer flooding* and *SYN flooding* [8]. The underlying cryptographic techniques are assumed to be computationally secure to prevent attackers compromising nodes

The adversaries are characterized based on the following two models [12]

- Model I: During the entire lifetime of the network, the adversary cannot break into or control k or more nodes.

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

Our scheme like the original scheme works on the model II.

5.3 Proposed Architecture

In this section we present the overall architecture. We present the localized trust model first and also present the general overview of the whole modified key management scheme next.

5.3.1 Localized Trust Model

In wired networks key management is done using a dominant localized trusted third party [13]. In the Trusted third party (TTP) model, a node is trusted only if it is authorized by a central certificate authority. While TTP has great efficiency and manageability by using centralized system for authentication, [8] it suffers scalability and robustness problem as well as a single point of failure.

Our Localized Trust Model (LTM) is based upon the one proposed by Lou et al [8]. In our model an entity or node is trusted only if it is authorized by some k neighboring nodes. These may be one hop nodes if there is more than k one hop neighbor or they may be multi-hop nodes based up on the density of the nodes in at a particular time when the node needs authentication. The solution proposed by Lou et al makes use of only the one hop neighbor, whereas our LTM uses only one-hop neighbors if there are

at least k nodes in the neighborhood otherwise it makes use of the multi-hop neighbors also.

Two ways to set the k parameter are as follow [8],

- Global parameter i.e. the value of k stays the same for all nodes thought through out the network life. This is otherwise called as system wide trust threshold.
- Location dependent variable, where value of k changes depending upon various other parameters such has geographic density, mobility, the number of nodes in the network etc.

There are several complications in setting k as the location dependent variable. Although this option provides more flexibility to work in concert with diverse local network topologies, [8] there is no system wide trust criterion. Due to the lack of good mechanisms to determine a node's neighborhood in a mobile environment, the adversary may make take advantage of this defective feature.

In our architecture we use the first option rather than the second one with a network wide fixed k that is tuned according to the network density ad system robustness requirement. In the previous model proposed by Lou et al [8], if a node cannot find k nodes in the neighborhood, it roams around until it finds k nodes. This means that it may take a long time to obtain k partial certificates which is an advantage to the adversary. The adversary has more time to attack and may trap a node in some geographic location

thereby reducing the mobility of the node. This would in turn slow down or starve the node indefinitely from getting a digital certificate.

Trust management is distributed in both the space (k) and time (T_{cert}) domains in our localized trust model. This property [8] is particularly appropriate for large dynamic adhoc wireless networks where the centralized trust management would be difficult or expensive.

5.3.2 Overview

In our architecture, every node has a pairs of keys, where one pair is the public and private key of the prospective node and other is the Secret Key (SK) and Public Key (PK) of the certificate authority. Since the PK of the CA is well known to all the nodes in the network, every node will have a copy of the PK. PK is used for certificate verification. Each node in the network should also carry a valid certificate. A node without valid certificate is not allowed to use the recourses of the network [8]. Like the original scheme, the nodes forward their certificates to neighboring nodes. Using the certificate they make authenticated safe links between them and help each other to route data packets or use other node's resources such as files etc. Each node is also monitors neighbor nodes to detect possible misbehaviors such as DoS, Break-ins etc. The monitoring mechanism used is up to each individual node.

As in the PKI scheme, each certificate has to be stamped with the expiration time. Once the time is expired the nodes should attempt to renew the certificate by requesting a new certificate to the CA. In PKI nodes contact the CA, whereas in our scheme the nodes

contact the neighboring nodes or the multi-hop nodes depending upon the node density since the certificate signing key is distributed among all the nodes in the network. In our scheme nodes need at least k nodes to make a new certificate. Once a node receives a request from a node for the partial certificate, it checks for records of the requesting node in its certificate revocation list (CRL). If the records show that the node is well-behaving legitimate node, it returns a partial certificate by applying its share of SK, otherwise the request is denied. By collecting K partial certificate, the nodes combine them to form a full certificate.

A valid certificate like the original scheme [8] represents the trust from a coalition of k nodes. Nodes with a valid certificate are trusted globally. By distributing certificate issuing/renewing service to neighboring and the multi-hop nodes we assure that the service is available ubiquitous and robust against DoS attacks.

5.4 Localized Certificate Services

In this section we present the localized certificate services. The localized certificate services in our scheme uses the same technique dynamic coalescing [8] used by Lou et al. A node V_i first finds out k nodes out of n nodes in the whole network. Coalition [8] can be formulated dynamically from any k responding nodes.

In general, Adhoc routing protocols can be categorized into two types: (a) on-demand and (b) pro-active protocols.

On-demand protocols create routes only when desired by the source node [16]. When a node requires a route to a destination, a route discovery process is initiated

within the network. Once a route has been established, it is maintained until the route is no longer required.

Pro-active protocols attempt to maintain consistent, up-to-date routing information for all nodes in the network [16]. This is typically achieved through maintaining a set of routing tables. Changes in network topology brought on by mobility or node failures, are catered for by propagating updates throughout the network at periodic intervals to maintain a consistent view.

One of the popular pro-active protocols is Destination sequenced Distance Vector Routing protocol (DSDV) [15]. Since the pro-active protocols are table driven i.e. the neighboring node information is stored in a table and updated periodically, networks using these protocols will not have any extra overhead in finding out the k -hop neighboring nodes. Our proposed scheme will best integrate with DSDV because this protocol demands each node to have information about routes to each other nodes in the networks including the hop counts.

Below is the overview of the DSDV routing protocol [15]

- Each Routing Table list all destinations and number of hops to each other nodes
- Each Route is tagged with a sequence number originated by destination
- Updates are transmitted periodically and when there is any significant topology change
- Routing information is transmitted by broadcast.

The figure.7 below shows an example of an Adhoc network and the contents of the routing table of node H6.

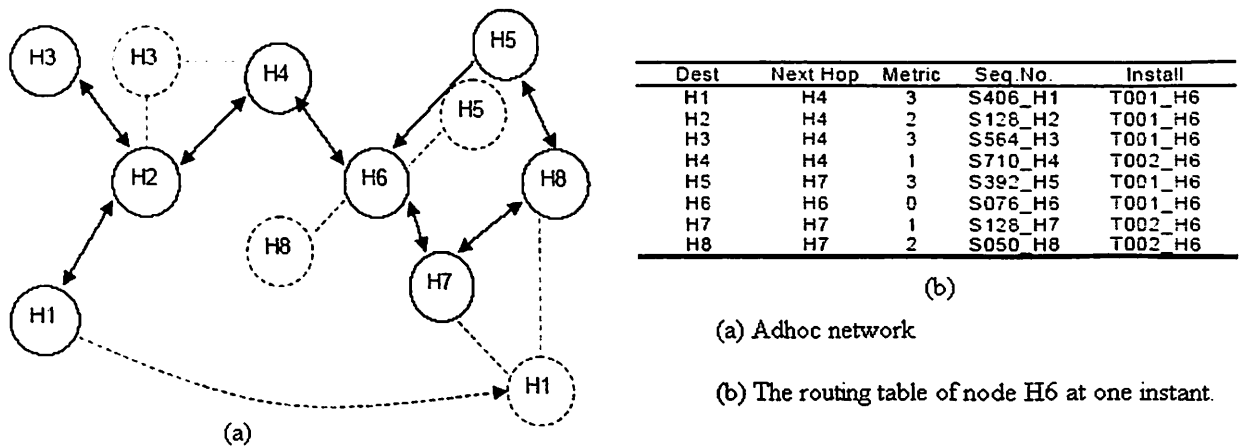


Figure 7 DSDV network and routing table

Figure 7.a shows the contents of the routing table of H6 at one instant of time. We sort the table based on number of hops to store the node information arranged in the order of metric (hops) as below in figure 8.

Dest	Next Hop	Metric	Seq.No.	Install
H6	H6	0	S076_H6	T001_H6
H4	H4	1	S710_H4	T002_H6
H7	H7	1	S128_H7	T002_H6
H8	H7	2	S050_H8	T002_H6
H2	H4	2	S128_H2	T001_H6
H3	H4	3	S564_H3	T001_H6
H1	H4	3	S406_H1	T001_H6
H5	H7	3	S392_H5	T001_H6

Figure 8 DSDV sorted routing table

From the table in the figure 8 we can count the number of nodes that can be reached in within a certain number of hops. Now in order to have k number of partial certificates, we try to contact the first k nodes in the table, this way we make sure there is no extra overhead. For example if we need 3 partial certificates, we have to contact 3

nodes in its neighborhood that is, nodes H4 and H7 in the first hop and H8 in the second hop.

In our modified scheme, a node H6 finds out the nodes to be contacted for certificates from the table in figure. 8. Now the node H6 sends request for the partial certificates with a hop limit of 2 and gets the partial certificates. The receive partial certificates are combined to form the full certificate. In case of failure because of the node break-in mobility or compromises the table is updated. The new hop limit is calculated and request is sent with an updated hop limit. This process is repeated until all the partial certificates are obtained.

The other category of protocols in Adhoc networks is pro-active protocols [16]. One of the well-known pro-active routing protocols is Dynamic Source Routing (DSR). These kinds of routing protocols do not maintain a table of routes to all the nodes in the network. For these networks we have proposed the following solution.

In DSR, the list of neighboring nodes is stored at a node. Only the route information is stored. Below is the overview of the DSR route discovery process [17]

- Source broadcasts route-request to Destination
- Each node forwards request by adding own addresses and broadcasts again.
- Requests propagate outward until Target is found, or a node that has a route to Destination is found.
- The destination node routes the packet containing route information back to the source

In our modified scheme for proactive Adhoc network protocols, a node V_i which needs a new certificate or a certificate to be renewed sends a pre-request signal similar to

a route discovery request to its neighboring nodes by broadcasting the request. The pre-request signal is propagated to k hop nodes where k is the certificate shares required to create a new full certificate .Once a pre-request signal is received, the receiving node sends back an acknowledgement signal to V_i with its own list of one-hop neighboring nodes information and broadcasts the pre-request signal to its neighbors and so on until k hops where k is the number of partial keys needed. The neighbor nodes information of k nodes is stored in k -Hop nodes information table.

<u>Hops</u>	<u>Neighboring Nodes Information.</u>
1	5, 9, 11, 33
2	12, 35, 34, 22, 3, 8, 19, 18
..
..
k	11,12,15,22,23,26,44,43,55,29,41,29,28,32.....

Figure 9 k -Hop information table.

The above table figure 9 is a K -Hop information table. All the one hop neighbors return their neighbors to the requesting node. For example, assume the one hop neighbors are nodes 5, 9 , 11, 13. These neighbors return their neighbors. For example node 5 returns its neighbors 12, 3, 19. Node 9 returns its neighbors 3, 18, 35, 34. Node 11 returns its neighbors 22, 8, 35, and node 13 returns its neighbors 18, 34, 34, 22. From this information the one hop and two hop neighbors can be constructed as shown above. V_i waits for T_{ack} time and counts the number of acknowledgements from its neighbor nodes. Here T_{ack} is the pre-calculated global constant time which is needed to contact k nodes in

the chain and get a reply in turn. If the count of the acknowledgements from neighboring nodes is greater than or equal to the threshold value k , then the node V_i sends an actual request for a partial certificate to all its neighbors with its valid id number specifying the number of hops.

After receiving the actual request the neighboring node checks the Certification revocation list for any bad records. If the history of the node V_i is clear, then the node computes the partial certificate using its share of PK and sends it back to V_i . Once all the k partial certificates are received the V_i combines all them to form a new valid certificate.

In node V_i does not receive k acknowledgements from its neighboring nodes it calculates the number of hops to be contacted to get k nodes using the k -hop node information table. From the k -hop node information table, the number of hops H to be contacted is calculated. Then V_i sends a special request to its neighbors. Once R_{sp} is received the nodes pass the request to their neighbors in turn and so on till H hops is reached. After passing the request the nodes check for the records of the node V_i and then create a partial certificate and sends to the V_i . The request R_{sp} is time stamped and expires at T_{ep} time. Each node also adds its id to the request for the partial certificate to trace the route back to V_i .

The certificate Issuing and renewal with Dynamic Coalescing is adapted from Lou et al [8].

5.4.1 Certificate Revocation List (CRL)

In order for any node V_j to authorize V_i before it issues a partial certificate, the records of the V_i should be verified for legitimacy. V_j has to maintain two kinds of records [17][18]. One is its direct monitoring data on neighboring nodes and the other is its certification revocation list (CRL). Each entry of the CRL has a node id and a list of accusers of the node. If the number of accusers is less than the threshold k and greater than 1 then the node is marked “suspect”.

The node V_j can convict a node in two ways, one by direct monitoring and other being accused from k legitimate nodes. If by direct monitoring node V_j finds that the node is misbehaving or broken, it marks the node as “convicted” and floods the message to all the nodes until T_{cert} time. If there is an accusing node, then before entering this information to the CRL, V_j checks if the accusing node is legitimate itself. If the number of accusing node is greater than k or equal to k then a node is marked convicted and V_j updates its CRL by taking away the id of the accused nodes from the accusers list of the other nodes in the CRL. If the number of accusers drops less than k then the node is marked down from “convicted” to “suspect”.

The proposed general Certification Service procedure for the pro-active Adhoc networks is given in the following steps,

1. Node V_i requesting a certificate renewal broadcasts pre-request signal R_{pre} to k hop neighboring nodes where k is the number of partial key shares needed to make a full certificate.

2. On receiving the signal R_{pre} the K-hop neighboring nodes send neighboring node information with acknowledgement.
3. The node V_i saves the nodes information in the K-hop information table.
4. After waiting Tack time, requesting node V_i counts the number of acknowledgments N_i from its neighbors. If the number N_i is equal to or greater than the threshold k , then the node V_i broadcasts actual request R to all the neighboring nodes. Node V_i gets the partial certificates from the one hop neighbors and uses Dynamic Coalescing to combine the partial certificates to create a full certificate.
5. If the number N_i is less than k then node V_i calculates the hops count H from the K-hop information table required to obtain k replies. V_i broadcasts a special request R_{sp} , asking the nodes to pass the request to its neighboring nodes also till a limit of H hops.
6. Once a node V_j receives R_{sp} , if it is within the H hop neighborhood from the requesting V_i node, it broadcasts the R_{sp} to its neighbor with its id attached to the request for the partial certificate. After broadcasting, the Request R_{sp} is processed and V_i is checked for legitimacy in the CRL. If it is legitimate, the partial certificate is sent to V_i using the route in the network.
7. Every time the R_{sp} is broadcast, the node sending R_{sp} adds its id to the R_{sp} . The node receiving R_{sp} similarly broadcasts R_{sp} to its neighbor adding its id. A node V_k receives a partial certificate from a node V_x which is one hop away from V_k , to be forwarded to V_j . Before sending this partial certificate to V_k , V_x checks if the route is secured, by verifying the validity of V_k by scanning V_k 's id in its CRL.

8. Repeats until H hop neighbors.
9. After receiving at least k partial certificates the node generates the full certificate.

Chapter 6

Simulation and Results

6.1 Performance Measures

To investigate the performance of our proposed solution over the fully distributed Certification Authority Scheme, both schemes are studied by varying several parameters. The simulation is created by having the area as a 2-D array and randomly placing nodes in the array. The required number of key shares (k) to construct a full certificate is pre-determined. We assume that we know parameters such as Transmitting-range and mobility. The simulator then randomly picks a node and attempts to make full certificate by getting k partial certificates using both Certification Authority Scheme and our proposed solution. We then measure the success and failure rate by the number of full certificates obtained among n nodes.

The following performance measures are used to evaluate the proposed approach

- Success% measured by the density of nodes in the network
- Success% measured by the key shares k (security)
- Success% measured by the transmitting-range(power)
- Success% measured by the percentage of compromised nodes in the network.
- Number of Hops required to obtain k shares by the density
- Number of messages by the key shares

6.2 Steps Involved in simulation

- 1) Create and randomly place nodes in the area represented by the two dimensional array.
- 2) Start moving the nodes based on the random-walk mobility model [20][21][22]. The random-walk mobility model is summarized as below,
A node moves from its current location to a new location by randomly choosing a direction and speed in which to travel. The new speed and direction are both chosen from pre-defined ranges, [*speedmin*; *speedmax*] and $[0; 2\pi]$ respectively. Each movement in the Random Walk Mobility Model occurs in either a constant time interval t or a constant distance traveled d , at the end of which a new direction and speed are calculated.
- 3) Randomly choose a node and try to get a full certificate for the node by obtaining k partial certificates. Use the fully distributed certificate authority scheme and also use our proposed scheme for all n nodes and get the average success rate.
- 4) Repeat the simulation by varying parameters such as
 - i) Density (number of nodes).
 - ii) Power (transmitting-range).
 - iii) Percentage of compromised nodes.
 - iv) Number of key shares (k).
 - v) Number of Hops.

6.3 Charts

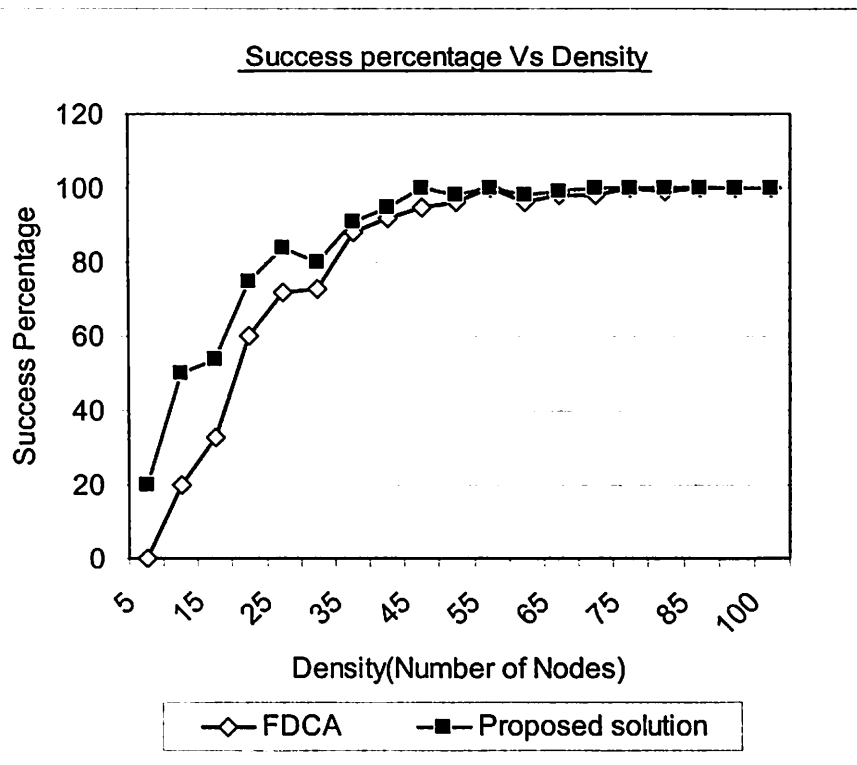


Figure 10 Density graph.

Figure 10 compares the success rate of the both fully distributed certificate authority and our proposed scheme. Here we vary the density of the nodes with the success rate (number of successful full certificates constructed). The other parameters such as transmitting range, number of key shares and mobility are constant and chosen as 3, 3 and 1 units respectively. The graph clearly shows our proposed solution has higher success rate than the original scheme (FDCA). Even at very low densities as low as 5 nodes in the whole network, our proposed solution was able to produce 20% success compared to the original scheme which totally failed at very low density. This result shows that our proposed scheme has better success rate even at very low densities.

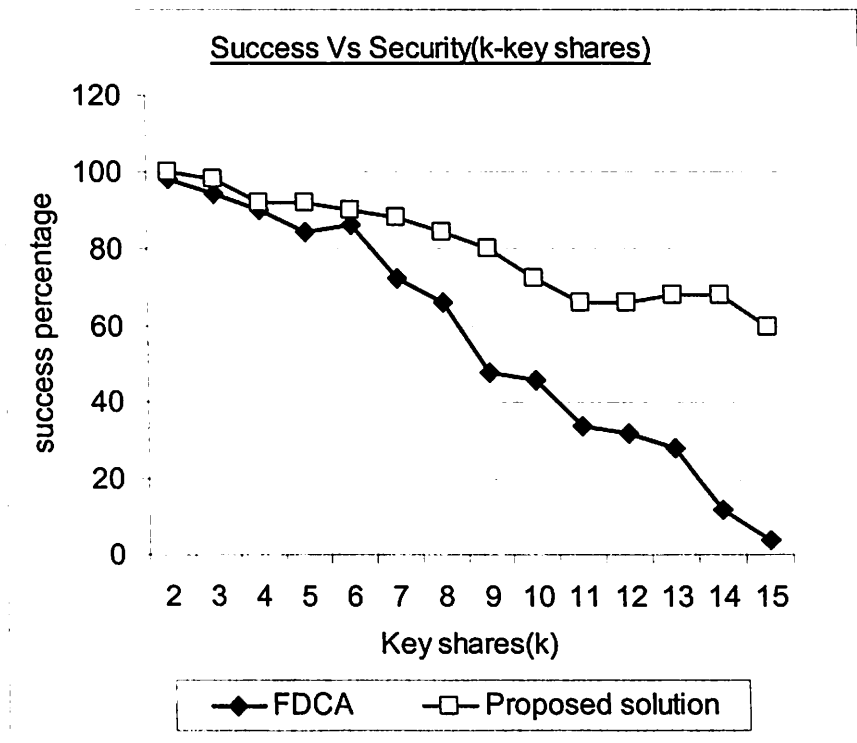


Figure 11 Security graph.

The figure 11 compares the success rate of both the fully distributed certificate authority and our proposed scheme. Here we change the number of key shares required by nodes to generate a full certificate. The other parameters such as transmitting range, number of nodes and mobility are constant and chosen as 3, 50 and 1 units respectively. Higher the number of key shares, higher the security. We varied the number of key shares between the range 2 and 15, and tried to measure the success rate of both the fully distributed certificate authority and our proposed solution. The graph clearly shows that, for as high number of key shares (15 key shares) FDCA totally failed whereas our proposed scheme had a 60% success. Clearly our proposed solution has better success rate than FDCA with higher security requirements.

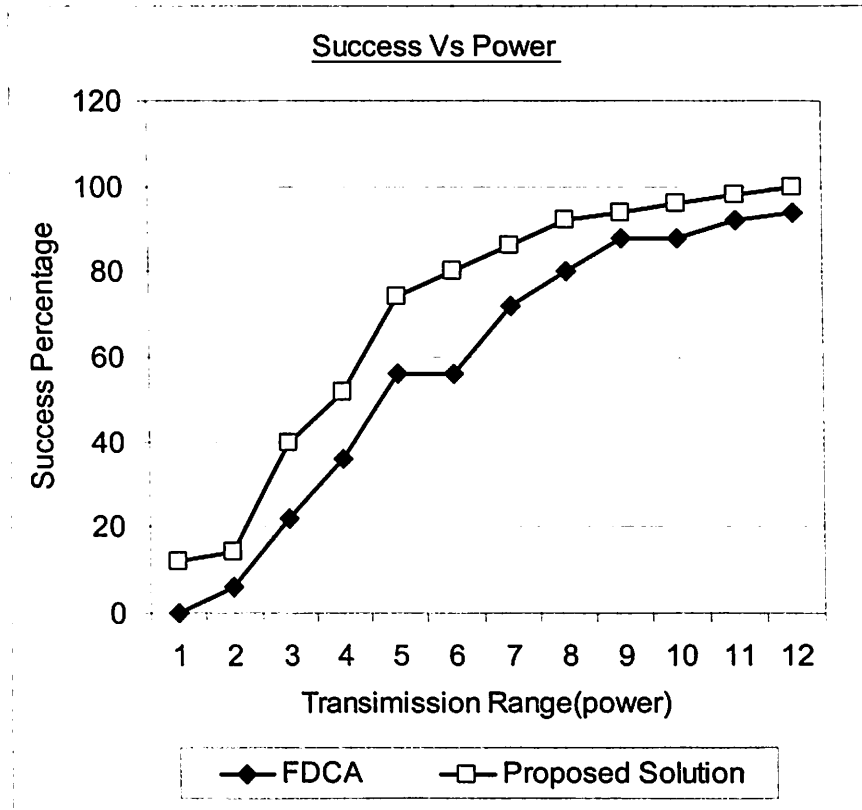


Figure 12 Power graph.

Figure 12 shows the success rate of both the fully distributed certificate authority and our proposed scheme. Here we vary the power i.e. the transmitting range of each node with in the range 1 – 12 units. The other parameters such as number of key shares, number of node and mobility are constant and chosen as 3, 50 and 1 units respectively. The FDCA totally failed for very low transmitting ranges (1 unit) whereas our scheme had 16% success. The graph clearly shows that even at low power, our proposed solution works better than the FDCA.

Clearly all the three charts given above show that our proposed scheme has better success rate than the fully distributed certificate authority (FDCA). Now we introduce bad nodes i.e. either compromised or dead nodes which pose denial of service attack or other malicious attacks and try to measure the success rate.

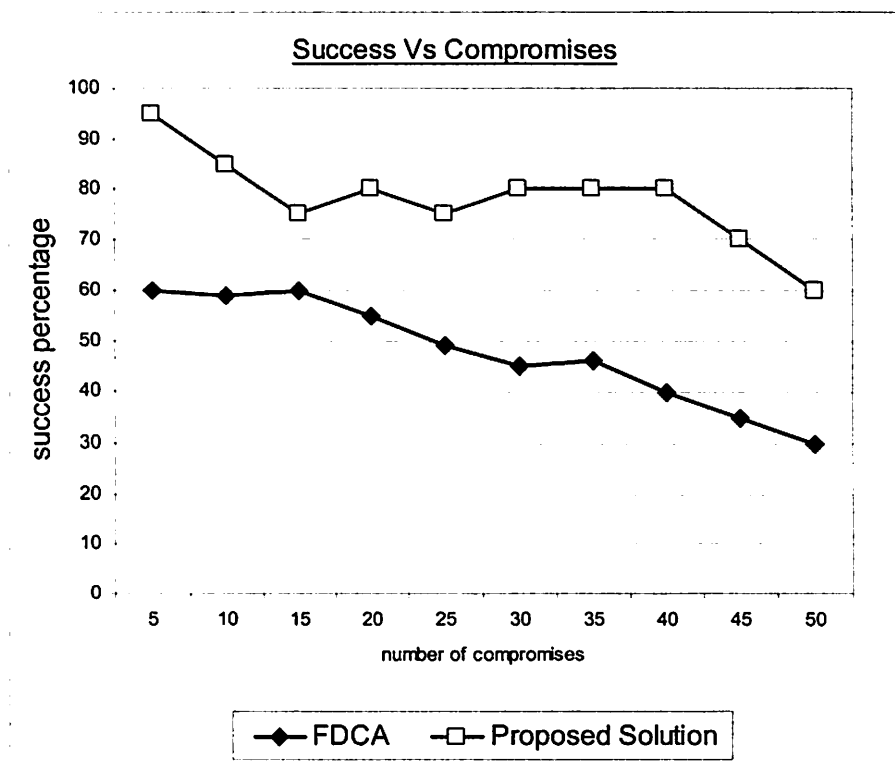


Figure 13 Compromise Graph.

Figure 13 shows the success rate of both the fully distributed certificate authority and our proposed scheme. Here we change the percentage of compromised nodes in each node’s neighborhood. We vary the percentage from 5 – 50%. The other parameters such as number of key shares, number of node and mobility are constant and chosen as 3, 50 and 1 units respectively.

The chart clearly shows that the fully distributed certificate authority cannot attain 100% success rate even for a minimal 5% compromised nodes in its neighbor. Our

proposed solution give 100% success rate for 5% compromises and gradually decreases as the percentage of compromised nodes increase. The graph shows that our scheme always performs better than the fully distributed certificate authority scheme.

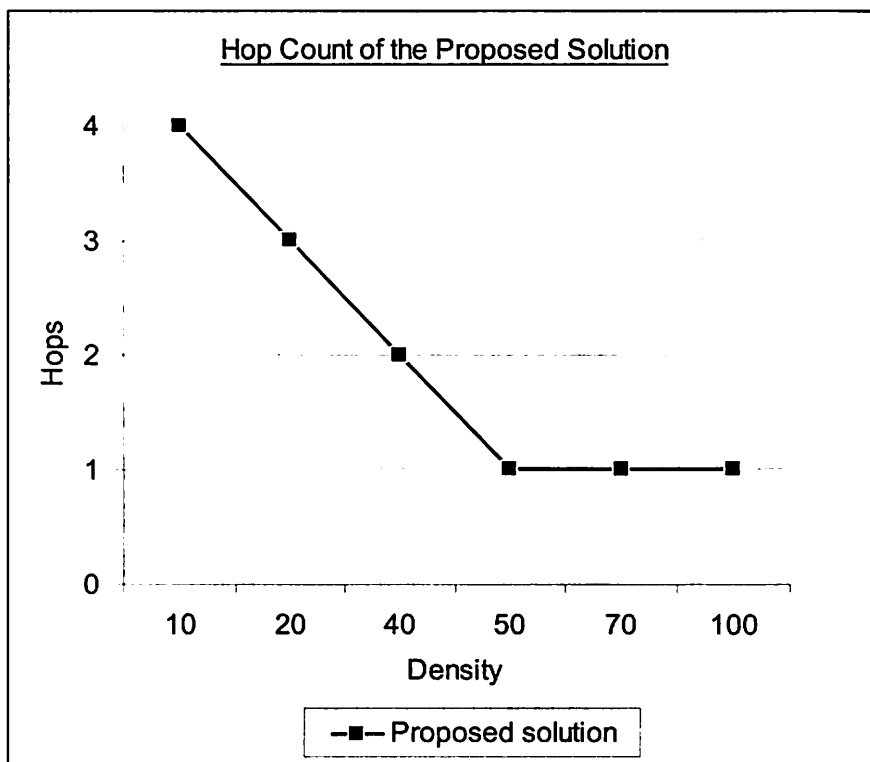


Figure 14 Hop count graph.

The figure 14 shows the chart which measures the number of hops required to construct a full certificate for various node densities. The fully distributed certificate authority will always need one hop to get all k certificates. When FDCA cannot find k nodes in its neighborhood, it has to move to another location and try again. Hence FDCA will more time than our proposed solutions since our scheme does not rely upon mobility for finding k nodes. In other words, even if k neighboring nodes are not available, our scheme can immediately start the process of obtaining k partial certificates. This is not

true for the FDCA which needs to wait until k nodes appear in its one-hop neighborhood. We see that the hops ranges from 4 to 1 are needed for obtaining 4 key shares on average. Though the fully distributed certificate authority schemes seems to have less message overhead because of one hop communication , the average success rate is about 58% whereas the average success rate for our proposed scheme is about 72%. This chart shows the overhead that is incurred to achieve a substantial increase in success rate using our proposed scheme.

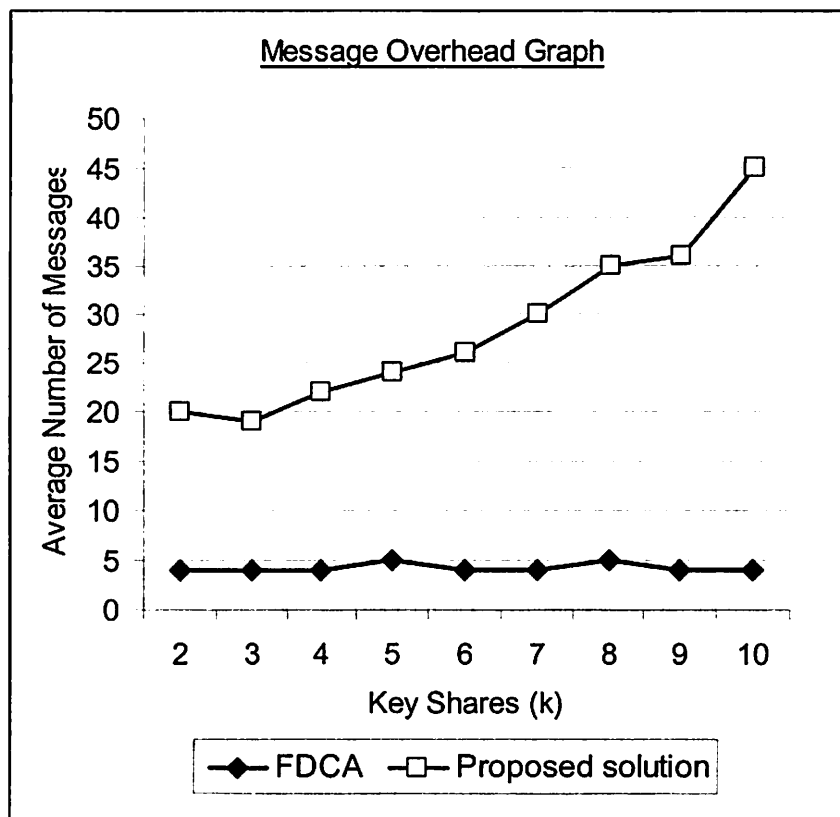


Figure 15 Message Overhead graph.

Figure 15 shows compares the message overhead of both the fully distributed certificate authority and our proposed scheme. We vary the number of key shares (k)

required to create a full certificate and get the average number of messages required by a success node to get k number of partial certificates. The other parameters such as transmitting range, number of nodes and mobility are constant and chosen as 3, 50 and 1 units respectively. The maximum number of messages in the FDCA scheme is 5 and 45 for our proposed scheme. Though the difference seems to be very high, the average success rate was 82% for our proposed scheme and 43% for the FDCA. Also when our scheme was used with DSDV there is no extra overhead because it uses only the already existing routing table.

6.4 Results

Using the simulation the following observations were made on our proposed scheme when compared to fully distributed certificate authority.

- Our scheme substantially increases the success rate in getting a full certificate
- Our scheme Utilizes more Channel Bandwidth
- Our scheme gives a better success rate when nodes are compromised. The success rate is better even if half the nodes are compromised. Our scheme is much better for network survivability.
- Our scheme gives better success rate even when the simulation is done in closed space (when nodes reach the four dead ends of the array, they cannot move forward and have to reverse back).
- Our scheme has faster key management because it does not have to wait until it meets k one hop neighbors.

Chapter 7

Conclusions

7.1 Conclusions

The fully distributed certificate authority cannot be used in all situations since an ad-hoc environment is very dynamic and network densities vary. It may not be possible to get enough key shares when there are few neighboring nodes . Our proposed scheme is well adapted to the highly dynamic nature of ad hoc networks. Consequently our multi-hop authentication technique handles many more critical situations such as node break-ins, obstacles, node compromises and power failures. The success rate using the multi-hop technique is significantly better than the fully distributed certificate authority. When implemented with the table driven routing protocols such as DSDV, the proposed scheme does not have any extra overhead. Although for proactive networks there is an overhead, the average success rate is always greater than the previous scheme.

7.2 Future Work

This work can be further improved.

1. A better technique can be used to limit the number of messages.
2. We have not investigated how to predict or find out a compromised node. This is an area that requires a lot of future research.
3. Faster technique to handle breaks in the routes discovered, due to faster mobility of the nodes.
4. The use of network simulators such as OPNET or ns-2 is also needed, for time calculations.

References

- [1] A.J. Menezes, P.C. Oorschot, S.A Vanstone, *Handbook of Applied Cryptography*, CRC Press.
- [2] http://www.cas.mcmaster.ca/~wmfarmer/SE-4C03-02/projects/student_work/grasicn.html#toc4
- [3] http://www.cio-dpi.gc.ca/pki-icp/beginners/whatisapki/whatisapki_e.asp
- [4] Y. Desmedt, *Threshold Cryptography*, *European Transactions on Telecommunications*, 5(4):449-457, July-August 1994
- [5] L. Zhou and Z. Haas, *Securing Ad Hoc Networks*, *IEEE Network Magazine*, 13(6), November/December 1999.
<http://citeseer.nj.nec.com/zhou99securing.html>
- [6] S. Yi and R. Kravets, *Practical PKI for Ad Hoc Wireless Networks*, August 2001
www.cs.uiuc.edu/Dienst/Repository/2.0/Body/ncstrl.uiuc_cs/UIUCDCS-R-2002-2273/pdf
- [7] S. Yi and R. Kravets, *Key Management for Heterogeneous Ad Hoc Wireless Networks*, July 2002
www-sal.cs.uiuc.edu/~rhk/pubs/tr-2290-1734.pdf
- [8] H. Luo, P. Zerfos, J. Kong, S. Lu and L. Zhang. *Self-securing Ad Hoc Wireless Networks*, *IEEE ISCC 2002*.
www.cs.ucla.edu/~jkong/publications/ISCC02.pdf
- [9] H. Luo and S. Lu, *Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks*, October 2000, UCLA-CSD-TR-200030
www.cs.ucla.edu/wing/pdfdocs/TR200030.pdf
- [10] Y. Zhang and W. Lee, "Intrusion detection in wireless ad hoc networks," *ACM MOBICOM*, 2000
- [11] S. Marti, T. Giuli, K. Lai and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," *ACM MOBICOM*, 2000.
- [12] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing," *Extended abstract*, 1995
- [13] R. Perlman, "An overview of PKI trust models", *IEEE Network*, p.38-43, vol.13, (no.6) Nov.-Dec. 1999

- [14] J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang, "Providing robust and ubiquitous security support for MANET," *IEEE ICNP* 2001, 2001.
- [15] Perkins Charles E., Bhagwat Pravin: Highly Dynamic Destination-Sequenced distance-Vector Routing (DSDV) for Mobile Computers, London England UK, SIGCOMM 94-8/94.
<http://www.cise.ufl.edu/~helal/6930F01/papers/DSDV.pdf>
- [16] Petteri Kuosmanen,"Classification of Ad Hoc Routing Protocols", Finnish Defence Forces, Naval Academy.
<http://eia.udg.es/~lilianac/docs/classification-of-ad-hoc.pdf>.
- [17] Y. Zhang and W. Lee, "Intrusion detection in wireless ad hoc networks," *ACM MOBICOM*, 2000
- [18] L. Gong, "Increasing availability and security of an authentication service," *IEEE Journal on Selected Areas in Communications*, Vol.11, No.5, Jun. 1993
- [19] Standard Performance Evaluation Corporation,
<http://www.specbench.org>
- [20] A. Abdul-Rahman and S. Hailes, "A distributed trust model", *ACM New Security Paradigms Workshop*, 1997.
- [21] R. Canetti, S. Halevi, and A. Herzberg, "Maintaining authenticated communication in the presence of break-ins," *Journal of Cryptology*, 13(1):61-105, 2000
- [22] Davies. Evaluating mobility models within an ad hoc network. Master's thesis, Colorado School of Mines, 2000.

Appendix

Source File:

```
=====
/*****
*      Simulation of Key management schemes in Adhoc Networks      *
*                                                                 *
*      FileName:          thesissimulation.cpp                    *
*      =====                                                  *
*                                                                 *
*      Purpose:                                                  *
*      =====                                                  *
*              This program simulates Adhoc Network Key management *
*      schemes one proposed in the thesis and also the fully distributed certificate *
*      authority                                                  *
*                                                                 *
*      Created by:                                              *
*      =====                                                  *
*              Jagadeeswaran Bhuvaneshwaran                      *
*                                                                 *
*      Last Modified:                                          *
*      =====                                                  *
*      11/18/2003                                              *
*****/
```

```
//Student Name: Jagadeeswaran Bhuvaneshwaran
```

```
//Student Id: 446155318.
```

```
//ADHOC Simulation
```

```
#include <iostream>
```

```
#include <stdlib.h>
```

```
#include <fstream>
```

```
#include <string>
```

```
#include <list>
```

```
#include <math.h>
```

```
#include <time.h>
```

```
using namespace std;
```

```
//variables defining the x and y axis of the area
```

```
int x=20;
```

```
int y=10;
```

```
//integer array defining the simulation area
```

```
int **area;
```

```

//variables defining the maximum and minimum mobility of the nodes
int maxspeed=3;
int minspeed=1;

//average speed of the nodes
int speed=1;

//variable defining the transmitting n range of the nodes
int k=3;

//variable defining the number of partial key defined
int nbr=3;//partial keys

//variable defining the number of nodes in the network
int nodes=50;

long average=0;

//variable to measure the hop count of each node
int hops=0;

//variables to measure the average success rates of the both schemes
int old_sucess=0;
int new_sucess=0;

//class defining the nodes
class node;

//class declaration of the node
class node
{
public:
    int id;//member for node id
    int x;//member for node x axis
    int y;//member for node y axis
    int t_range;//member for node transmitting range
    int **k_table;//member array to store k-hop neighboring node information

Public:
    static int nodenumber;//member to store number of nodes in neighborhood
    node();//constructor
    bool compramised;
    bool move();//member function to make the node move in random when called
    bool updatek_table();//member function to update its k-hop neighborhood table
};

```

```

int node::nodenumber=0;//initializing static variable

node *terminals1;//declaring the node pointer to create and allocate nodes

//implementing the constructor
node::node()
{
//initializing and creating k-hop table dynamically
    k_table=new int*[nbr];

//allocating space
    for( int i=0; i<=nbr;i++)
        k_table[i]= new int[::nodes];

//deciding the node is compramisid dynamically
    if(rand()%100<15)
        compramisid=true;
    else
        compramisid=false;

//allocating area for the node created and finding out the neighboring node if the space is
//occupied then finding a random space until we find one
    while(true)
    {
        int x_=rand()%(::x);//randomly generating x space
        int y_=rand()%(::y);//randomly generating y space
//finding if it is free and allocationg the node to it
        if(area[x_][y_]==-1)
        {
            this->x=x_;
            this->y=y_;
            this->id=nodenumber;

            area[x_][y_]=this->id;
            //cout<<x<<" "<<y<<" i"<<area[x_][y_]<<"\n";
            nodenumber++;

            break;//break if found
        }
    }

//initializing the transmitting range
    this->t_range=k;
}

```



```

//end of constructor

//implementing the member function to move the node in random
bool node::move()
{
    //saving the previous location in a new set of variables
    int x1=this->x;
    int y1=this->y;

    //finding out the new direction in random
    //selection 1 out of 5 number 4 representing 4 directions and 1 for same location
    int r=rand()%5;//random direction

    //changing direction with respect to speed (mobility)
    if(r==1)
        x1=x+speed;//finding out the new direction and distance

    if(r==2)
        y1=y+speed; ;//finding out the new direction and distance

    if(r==3)
        x1=x-speed; ;//finding out the new direction and distance

    if(r==4)
        x1=y-speed; ;//finding out the new direction and distance

    //if the new location is same don't change or else find if the space is free and change the
    //location
    if( x1<::x && x1>=0 && y1<::y && y1>=0 && area[x1][y1]==-1 )
    {
        area[this->x][this->y]=-1;
        area[x1][y1]=this->id;
        this->x=x1;
        this->y=y1;
    }

    return 0;//return back
}

//end of move function

//implementing the update k -table function
bool node::updatek_table()
{
    //finding the one hop neighbors first

```

```

for(int i=0;i<nbr;i++)
{
    for(int j=0;j<::nodes;j++)
        { this->k_table[i][j]=-1;
          //saving the new neighbor information in the k-hop table
        }
}

//variable for tracking hop count and the index of the k-hop table
int index=0;
int cnt=0;

for( i=0;i<::x;i++)
    {
        for( int j=0;j<::y;j++)
            {
                //check if the all nodes fall in the range of the transmitting range of the current
//node in the one hop neighbors if any save them in the table
                if( int(sqrt(pow((this->x-i),2)+pow((this->y-j),2)))<=this->t_range )
                    {

                        if(area[i][j]!=this->id && area[i][j]!=-1)
                            {
                                this->k_table[0][index]=area[i][j];//recording in table
                                cout<<"i"<<area[i][j];//printing them

                                if(!terminals1[area[i][j]].compramised)
                                    cnt++;//adding the count
                                index++;//going to next hop
                            }
                        }
                    }
}

// if the nodes in the neighborhood is greater than or equal to k shares then record
//success
    if(cnt>=nbr )
        old_sucess++;//incrementing

bool record=false;    int indexcount=index;
int cnt1=cnt;
//recording one hop node count

//in this loop we recursively go for every node in the hop table and record nodes in the
//next hop
    for(int i1=1;i1<::nbr;i1++)//loop until k hops

```

```

{

    int index1=0;

    //finding the id of each node in the previous hop
    for(int j1=0;j1<index;j1++)
    {
        int _x_,_y_;
        for(int i2=0;i2<::y;i2++)
        for(int j2=0;j2<::x;j2++)
        if(area[i2][j2]==this->k_table[i11-1][j1] )
        { _x_=i2; _y_=j2;}//getting the x,y axis of the node

        //checking if any nodes fall in the range of the selected node, if any record their
        //id in the k-hop table
        for( i=0;i<::x;i++)
        for( int j=0;j<::y;j++)
        if( int(sqrt(pow((_x_-i),2)+pow((_y_-j),2)))<=this->t_range)//checking for the
        //range
        {

            bool already=false;

            if(area[i][j]!=this->id && area[i][j]!=-1)//checking the area of
            //both nodes
            {

                for(int i12=0;i12<=i11;i12++)
                //checking the table if the node already exists if it is then ignore else record
                for(int i3=0;k_table[i12][i3]!=-1;i3++)
                    if(k_table[i12][i3]==area[i][j])
                        already=true;

                //if not in the table record
                if(!already)
                {
                    this->k_table[i11][index1]=area[i][j];
                    cout<<"nbr"<<area[i][j];
                    if(!terminals1[area[i][j]].compramised)
                        cnt1++;
                }

                //we also record the node hop count
                if(cnt1>=nbr && !record )
                {

                    cout<<"cnt"<<cnt1;
                    hops=hops+i11+1;//finding the hopcount
                    cout<<hops;//printing it
                    //getchar();
                }
            }
        }
    }
}

```

```

        record=true;
    }
    index1++; //incrementing index
    }
}
}
cnt=cnt1;
index=index1;
cnt1=cnt1+cnt;
//updating counts and indexes
    indexcount=indexcount+index;
}

//checking if the number of nodes in khop is equal or greater than the k share nodes If any
then record sucess
if(indexcount>=nbr)
    new_sucess++;

//finished updating table
    cout<<"\n";
    cout<<"finished";
//retun
    return 0;
}
//end of the function update table

//main funtion
int main()
{
//a temporary k table
    int **k1_table;
//initializing the random seed
    srand ( time(NULL) );

//initializing the area array
    area=new int*[x];

//allocationg dynamic area array space
    for ( int i=0; i<x ;i++)
    {area[i]= new int[y]; //allocationg
    }

//initializing the area with -1 indicating empty space (no nodes)
    for( i=0 ;i<::x ;i++)
        for(int j=0;j<::y ;j++)

```

```

        area[i][j]=-1;
//creating the number of nodes
    node *terminals = new node[nodes];

    terminals l=terminals;

//dynamically allocating k table
k1_table=new int*[nbr];

//2d array allocation
    for( i=0; i<=nbr;i++)
        k1_table[i]= new int[:nodes];
//taking one node in random and moving it once for five times
for(int l=0;l<5;l++)
{
    for( i=0 ;i<nodes ;i++)
        {
            terminals[i].move();//calling move function on the node selected

        }

//we then call the call updates fuction on every node to update its k-hop table
for( i=0 ;i<::x ;i++)
for( int j=0 ;j<::y ;j++)
{
    if (area[i][j]!=-1) {for every node if not a free space call the update fuction
        cout<<area[i][j]<<"\n";
        terminals [area[i][j]].updatek_table();//calling udate fuction
    }
}

//printing success rate for every node for FDCA scheme
cout<<l<<" "<<old_sucess;

}

//printing success rate of FDCA and new scheme
cout<<"old "<<((old_sucess/5)*100)/nodes<<"new
"<<((new_sucess/5)*100)/nodes<<"hops"<<(hops/5)<<" "<<(new_sucess/5);

/*
int sucess_main=0;

//loops to findout the number of messages and attempts made
for(int h=0;h<10;h++)

```

```

{
    int attempt=0;
    int messages=0;

//take one node in random out of 100
int node= rand()%100;

//checking for successes and get hop count until success
while(true)
{
    for( i=0; i<=nbr;i++)//looping until the k shares hops
        for( int j=0; j<=nodes++)
            k1_table[i][j]= terminals[node].k_table[i][j];
            messages=messages+::old_sucess;
//recording success and hops and messages

        for( i=0 ;i<nodes ;i++)
        {
            Terminals[i].move();//move in random

        }

        terminals[node].updatek_table();//update the table
        int cnt=0;

        for ( int j=0; j<=nodes;j++)
        {

            //loop for all space which are not free
            for( int j1=0; j1<=nodes;j1++)
                if(k1_table[0][j1]== terminals[node].k_table[0][j] &&
k1_table[0][j1]!=-1)
                    cnt++;

            }

            sucess_main=sucess_main+(:old_sucess-cnt);
            attempt++; adding attempts by old scheme
//finding the success percentage and exit if more than 90%
            if((sucess_main*100)/nbr >= 90)
                break;

        }

//printing the number of messages in total
cout<<"total messages="<<attempt+messages;*/
}
//end of main function

```

List of Figures

Figure		Page
1	Wireless LAN.....	1
2	Adhoc Networks.....	2
3	Asymmetric Encryption	11
4	Partially Distributed Certificate Authority.....	20
5	Share Update Graph	23
6	Fully Distributed Certificate Authority.....	26
7	DSDV Table.....	42
8	Modified DSDV Table	42
9	K-Hop Information Table.....	43
10	Density Graph.....	52
11	Security Graph.....	53
12	Power Graph.....	54
13	Compromise Graph.....	55
14	Hops Count Graph.....	56
15	Message Overhead Graph.....	57

List of Symbols

CA	-	Certificate Authority
DoS	-	Denial of Service
TTP	-	Trusted Third Party
LTM	-	Localized Trust Model
DSDV	-	Dynamic Sequenced Distance Vector
DSR	-	Dynamic Source Routing

Vita

①

Jagadeeswaran Bhuvaneshwaran

Candidate for Degree of

Master of Science

Thesis: Enhancing Self-Security in Wireless Adhoc Networks using Multi-hop Authentication

Major Field: Computer Science

Biographical:

Personal Data: Born in Kundrathur, Tamilnadu, India, On September 20, 1980, the son of Mr. S. Bhuvaneshwaran and Mrs.B.Shanthi.

Education: Graduated from Holy Angels High School, Tamilnadu, India in April 1997: Received Bachelor of Engineering in Computer Science and Engineering from the University of Madras, Chennai, Tamilnadu, India in April 2002. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University, Stillwater, Oklahoma in May 2004.