BAYESIAN NETWORK TRUST MODEL

FOR CERTIFICATE REVOCATION

IN ADHOC WIRELESS

NETWORKS

BY

SUDHA CHINNI

Bachelor of Technology

Jawaharlal Nehru Technological University

Hyderabad, India

2002

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
December, 2004

BAYESIAN NETWORK TRUST MODEL

FOR CERTIFICATE REVOCATION

IN ADHOC WIRELESS

NETWORKS

Thesis Approved:

Dr. Johnson Thomas

Thesis Advisor

Dr. John P. Chandler

Dr. Debao Chen

Dr. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my sincere and heart full gratitude to Dr.Johnson Thomas for giving me an opportunity to work under his guidance, I would also like to thank him for his consistent guidance, valuable suggestions and commendable support for the successful completion of my thesis.

I would like to extend my hearty thanks to Dr.John Chandler and Dr.Debao Chen for their support and cooperation.

I take pleasure to thank my parents for their love and support for pursuing my Masters at Oklahoma State University. Finally, I wish to convey my sincere thanks to all the teaching and non-teaching staff of the Dept of Computer Science, Oklahoma state university, without whose cooperation this venture would not have been a success.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

NOMENCLATURE

GSM	Global system for mobile communications
CDMA	Code Division multiple access
TDMA	Time Division multiple access
CA	Certification Authority
RA	Registration Authority
PKI	Public key infrastructure
CRL	Certificate revocation list
IDS	Intrusion detection system
Qos	Quality of service
T_{cert}	Certificate validity period
ns-2	Network simulator-2

CHAPTER I

INTRODUCTION

Today’s earth is called a global village since there is no criterion of distance and time for communications. Man can communicate almost from one place to any other place in no time. Wireless technologies like GSM, CDMA and TDMA as well as wired protocols such as TCP/IP make this possible. Business organizations as well as individuals are increasingly dependent on mobile communications and the Internet. The integration of the internet and wireless communication has led to the advent of wireless internet.

1.1 Wireless internet:

The Wireless internet is different from the wired internet in the aspect of maintaining communication channels between computers. In the wired internet they are maintained through cables, whereas in the wireless internet, they are maintained through radio waves or micro waves. The main reason for the popularity of the wireless internet is its mobility and the ease of installation and setup.

The wireless internet has advantages over the wired internet in the following scenarios:

- When a building is not pre-wired with network cable, it will be more cost effective to install and use wireless internet to network the computers rather than wiring the building and installing wired internet components.

- In business environments or at home when wireless internet is used, the user is not restricted to a particular location and is able to be mobile.
- Bulky cabling can be avoided for both business and home networks.[1]

The Wireless internet is broadly classified into two types:

1. Infrastructure mode
2. Ad hoc mode

1.1.1 Infrastructure Mode:

When this kind of mode is used for networking the computers, there are a set of wireless devices and wired devices in the network. The wireless devices can communicate with the wired devices through a central point called the “Access point”. Therefore, if a wireless device wants to transfer a file, share a file or access a resource in the network, it has to communicate with the access point first. In this mode, the configuration of the access point plays a crucial role.

To ensure proper working of the network the Access point should be placed at a point where the wireless device can communicate with minimal interference. Various standards have been proposed including 802.11a, 802.11b, and 802.11g. 802.11g is the current best standard for wireless internet due to the following reasons:

- Very high speed
- Supports More number of simultaneous users
- Better signal range with less interference

Applications:

The infrastructure mode is applicable in

- scenarios where a wired and wireless infrastructure is present
- It is therefore good for both home and business networking

1.1.2 Ad hoc Mode:

In this mode there is no attachment to a wired network; each device can talk to another without any central point (Access point). Here the wireless devices can only share files; they cannot share any resources as they are not connected to any network. The communication is similar to a peer to peer communication.

This mode offers a superior mobility than the former approach as there is no attachment with networking cables; the wireless card attached to the device passes radio signals to the access point for accessing the internet, hence maximum mobility is achieved without worrying about staying at a place where the network cable is attached. Any 802.11 standard can be used for the Ad hoc mode. [1]

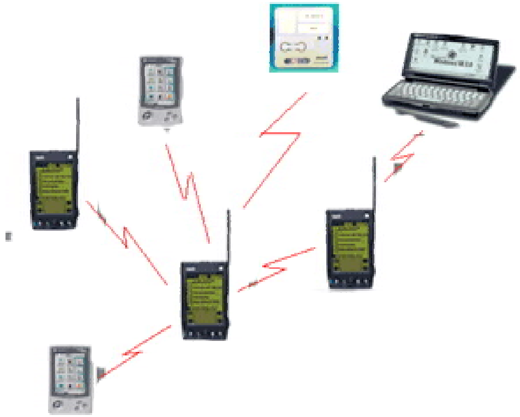


Figure 1: Example application of Ad hoc wireless network [2]

Applications:

The ad hoc mode is applicable where a wired infrastructure may not be available such as in:

- Military operations
- Emergency search-and-rescue missions
- Data collection/sensor networks
- Instantaneous classroom/meeting room applications

1.2 Security in networks:

The Internet has made shopping, banking, investment and leisure pursuits a simple matter of "clicking and dragging". Consequently, industries and business rely heavily on the internet to operate and conduct business. The online business has become a platform for purchasing goods, money transfers and credit card purchases to name a few business transactions, which makes online transactions more vulnerable to cyberspace criminals. The online business has resulted in different kinds of thefts of which identity theft is only one of the many potential kinds of attacks and thefts. There are a number of threats to computer security including viruses, worms, intruders, insiders, criminal organizations, terrorists and information warfare conducted by foreign countries. Due to these reasons it is highly essential to safeguard the networks and systems. In the case of a wired network steps can be taken to minimize the attacks by installing patches to the operating systems, installing firewalls or monitoring traffic at core routers. All this is possible as the network is connected and there is a central point for administrative operations.

In Ad hoc wireless networks, security is very difficult to handle since the network is wireless, which provides more scope for attacks and more room for intruders. Providing security to systems in an ad hoc network is very difficult for a number of reasons. The communications takes place in an open medium which permits eavesdropping, there is no central administrative point and the number of nodes in the network is never constant. As nodes can enter or leave the system in the absence of a central authority each node in the network needs to be authenticated before it can start communicating [3].

To ensure security, each node in the network is given a certificate which authenticates it for communicating in the network; this is valid for a fixed period of time after which the certificate needs to be revoked. The performance of each node in the network is monitored during the certificate validity period and if it is not up to expectations the node is not reissued a certificate the next time. This mechanism ensures that the nodes in the network are legitimate nodes and there is no malicious activity prevailing in the network. This is achieved with the help of “certificate revocation lists”. A certificate revocation list is a table maintained by each node in the network which contains the ID’s of the misbehaving nodes and the list of accusers. If the number of accusers against the node are greater than a threshold (a globally fixed value) then the node is marked as convicted.

If the number of accusers is below the threshold then the node is marked as a suspect. Certificates are not granted to convicted nodes. In this thesis we propose a novel scheme for certificate revocation which is implemented using Bayesian artificial intelligence. A comparison is made on two trust models, one with Bayesian Networks and one without Bayesian networks using simulation. This thesis is organized into the following sections. Chapter 2 provides a general introduction to ad hoc wireless networks and the associated mobility models. Chapter 3 introduces various security mechanisms used in networked applications. Chapter 4 gives a review of trust models for ad hoc wireless network and describes a trust model that is the basis for this work .Chapter 5 discusses the importance and nature of trust. Chapter 6 gives the solution to the various problems identified in chapter 4 by proposing the implementation of a novel trust model using Bayesian artificial intelligence. Chapter 7 compares two trust models using simulation. Chapter 8 concludes the thesis and provides a brief description of future work.

CHAPTER II

ADHOC WIRELESS NETWORKS

2.1 Introduction:

An Ad hoc wireless network or MANET (Mobile Ad hoc NETwork) is formed dynamically by a group of moving nodes. It has no infrastructure, and the network has no backbone or a central point of communication.

In this kind of network there are no routers, hubs or other intermediate devices for communication; each node acts as a router and transfers information. An Ad hoc network may consist of a mixture of heterogeneous devices that are capable of communicating over the wireless channel .Multi hop routing is performed by the devices in the network due to the limited transmission range of the interfaces. Hence an ad hoc network is also called a multi hop network [4].

Ad hoc wireless networks have proven to be very useful in scenarios where there is no fixed infrastructure, where it is very difficult to establish base stations and where timely updating and maximum mobility is required.

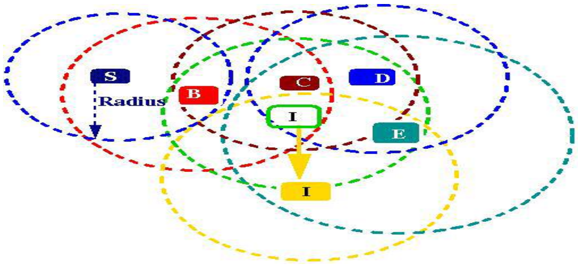


Figure 2: Nodes in ad hoc wireless networks [3]

2.2 Applications:

Figure 3: Applications of Ad hoc networks [5].

There are various areas where Ad hoc networking can be used; these include the following:

1. **Conferences:** The use of Ad hoc networks in such environments will enable the exchange of ideas at any time and at any place, providing portability and mobility in communication.
2. **Military operations:** In the battlefield, Communication of voice as well as data is required. Battles take place in isolated places where there are no civilians, generally on rough terrains where the establishment of base stations is very

difficult. Hence ad hoc networks can be very useful for communication because of the mobility it provides.

3. **Disaster / Rescue operations:** Whenever a disaster or a calamity occurs there is a high possibility that the communication links are damaged, hence any kind of communication relying on infrastructure is not helpful. In such situations ad hoc networks provide a means for communication.
4. **Civil Enforcement operation:** Whenever there is an emergency situation, law enforcement officials need to be in contact, wherever they are, Ad hoc networks are well-suited to these situations due to portability and mobility and non-interdependence on fixed infrastructure.
5. **Home networking:** Ad hoc networks may be used at home to inter-connect computers and to transfer files.
6. **Collaborative applications :** In collaborative applications, a set of people work on the same project and share their work on the network., In such situations ad hoc networks provide more mobility and portability and information can be shared anywhere at anytime.[2]

2.2.1 Integration of Bluetooth and Ad hoc wireless networks:

Bluetooth technology: This technology enables communication between portable devices without the requirement of cabling between them. This is done with the use of radio signals. This technology can be used in combination with ad hoc wireless networks as shown in Figure 4.

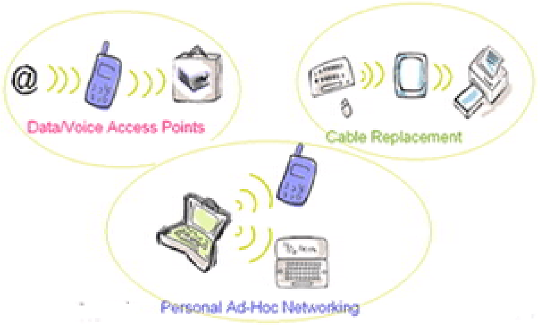


Figure 4: Ways of Utilizing Bluetooth Technology. [2]

2.2.2 Future application:

In future, with the help of ad hoc wireless networks and blue tooth technology various applications at home can be controlled in an ad hoc way, all with one device. [2]

This is illustrated in Figure 5:

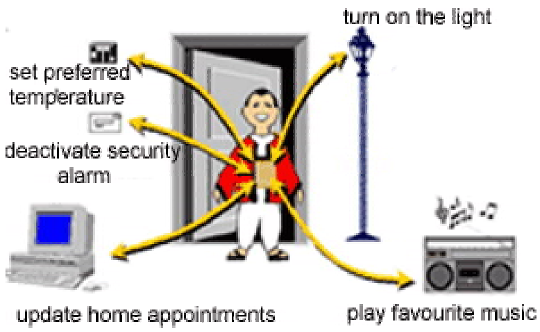


Figure 5: Home wireless ad hoc applications. [2]

2.3 Limitations:

Ad hoc networks have a number of limitations and constraints.

1. **Changing Topology:** In a mobile network the topology keeps changing as the nodes move. The interconnection or a link between two nodes is not fixed. This complicates various aspects related to communications including routing of data, security and authentication of nodes and quality of service to name a few.

2. **Interference:** A wireless communication media introduces a lot of interference. Interference, results in wasteful communication. Apart from the interference caused by the wireless channel there may also be disruption in communication due to the mobility of nodes.
3. **Link Capacity:** Manets have lower link capacity than wired networks, due to the noise and limited bandwidth of wireless channels.
4. **Limited Energy:** A Mobile node is battery operated, which has limited life time Hence every design issue should be taken with the view of conserving the energy of the devices.
5. **Less Security:** Since the devices are small and not connected to a fixed infrastructure, they are more vulnerable to malicious attacks and be stolen, hijacked or compromised. As the media is wireless, any device can enter or leave the network.
6. **No central authority:** Since there is no central authority or infrastructure each node is autonomous, hence it is difficult to synchronize the activities in the network [4].

2.4 Challenges of ad hoc wireless networks

Among the various challenges of ad hoc wireless networks, the most crucial one is security. The nodes in the ad hoc network operate over the wireless media; this is accessible to all users including malicious attackers. Hence, it is highly vulnerable to attacks. The attackers can exploit or possibly disable the network. In order to avoid these problems each node in the network needs to be authenticated using any cryptographic mechanism which will be supported by a certificate authority (CA). Any disturbance with the CA will affect the security of the entire network.

Another important challenge is to provide a constant and good quality of service in an environment that is constantly changing. An adaptive quality of service must therefore be implemented over the traditional resource reservation to support the multimedia services in a mobile ad-hoc network. Ad hoc networks should be able to balance traffic load among nodes so that the power constrained nodes can be put into sleep mode while traffic is routed through other nodes [6]. Another important challenge is scalability in the view of future expansion [7, 8]. Introducing hierarchy into the network topology will improve scalability [8].

2.5 Mobility Models for Ad Hoc Networks

Mobility models are required to simulate the behavior of the network. When an authentication mechanism is imposed on the nodes, performance can be analyzed by simulating the network behavior based on a mobility model. A mobility model describes the mobility pattern of the nodes that includes their speed, direction and also their state being either stationary or moving. Various other factors like transmission range of each node and reach ability are also considered. There are two types of mobility models [9]:

1. Traces
2. Synthetic models

Traces are the data obtained from an already existing network. But when a new protocol needs to be tested on a non-existent network, synthetic models are used. Synthetic models try to mimic the movement pattern of a node in an ad hoc network [9].

There are many synthetic mobility models such as:

1. Random walk mobility model
2. Random waypoint mobility model
3. Random direction mobility model
4. A boundless simulation area mobility model
5. Gauss-Markov mobility model
6. City section mobility model
7. Probabilistic version of the random walk mobility model

There are some other group mobility models that are used when the nodes move together as a group [10].

2.5.1 Random Waypoint Mobility Model

Random waypoint mobility model is chosen as the mobility model for this work. The model described in [9] is very widely used to evaluate protocols designed for ad hoc wireless networks. A random pause time is introduced for the nodes in between their movement. Maximum speed of a node, maximum pause time, number of nodes and simulation area are some of the inputs to this model. Each node present in the network follows the following steps [9]:

1. Each node is placed at a random location in the given simulation area initially.
2. A random speed is chosen for each node. The speed is less than the maximum speed. Also a random destination is chosen within the simulation area. The node begins to move towards the selected destination with the selected speed.

3. On reaching the destination, a random amount of pause time is chosen for the node that is less than the maximum pause time. The node pauses at that location for the selected pause time.
4. The node repeats steps 2 and 3 for the entire simulation time.

CHAPTER III

SECURITY IN NETWORKS

The nodes in an ad hoc network are wireless, therefore there is a high possibility that they can be stolen, captured or compromised. Such nodes can cause considerable damage to the network. Handling security issues in the wireless environment is difficult as the nodes are mobile and there is no centralized authority. [11]

3.1 Security Threats:

The following are the possible security threats in ad hoc networks:

1. **Passive Eavesdropping:** Listening to other's communication secretly is called Eavesdropping. In ad hoc networks, the media of communication between the nodes is wireless, hence any node can just listen to communication by tuning to that frequency.
2. **Denial of service attacks:** In this kind of attack, the attacking node will send enormously high requests to the service providing node. As a result, the performance of the service providing node will be slowed down. The service providing node may not be able to provide service to other nodes; this is called Denial of Service attack.

3. **Masquerading:** In this kind of attack, the attacker steals the outgoing packets from one machine and overwrites the source IP address by its address and then sends it. This could be done to the packets coming from an opposite direction. This could be done to all packets in the network, and the entire network can come under the control of the attacking node. This is called masquerading and it is also called "spoofing".
4. **Replay:** In this kind of attack the attacker tries to send the message to the sender as a legitimate message. For example, 'A' authenticates itself to 'B' using a password and the message is intercepted by an attacker 'C'. After 'A' and 'B' have finished communicating, 'C' replays the message and 'A' thinks 'B' is the sender.

In the light of the above threats, it is necessary for each node to be able to defend itself. Authentication is the first step to self defense. When one node is communicating with another node or exchanging some information it should be assured that it is communicating with the intended node only. This is accomplished by the public key cryptosystems, which becomes another approach to self defense.

3.2 Public key infrastructures:

A public key infrastructure provides confidentiality, integrity, non-repudiation and authentication among the two entities that are communicating. This infrastructure is based upon symmetric and asymmetric cryptosystems. The main advantage of a public key infrastructure is that it provides interoperability among the various products and technologies to incorporate various security mechanisms.

Each entity will create a public, private key pair for it .When an entity wants to send a message to a particular entity it will encrypt the message using the public key of the target entity. When the target receives the encrypted message it will decrypt it with its private key. The public keys of the entities in the network can be obtained from a "Directory of public keys" or can be requested from the target entity. The secret key of an entity is only known to the entity and it is not revealed to anyone.

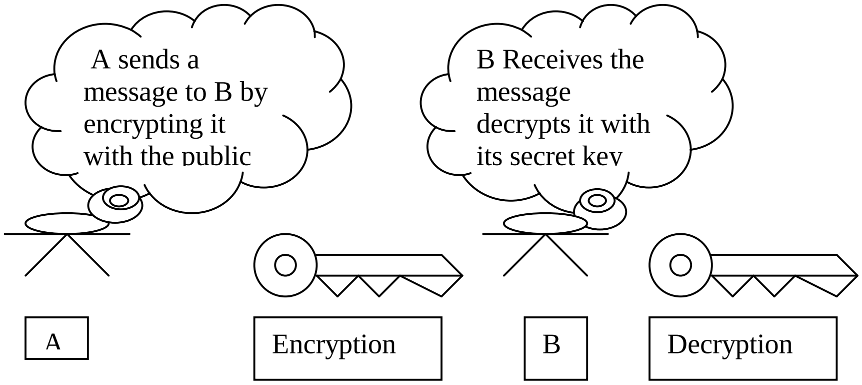

A sends a message to B by encrypting it with the public
B Receives the message decrypts it with its secret key
A
Encryption
B
Decryption

Figure 6: Working of a public key cryptosystem

In this method we can be assured that the communication is confidential, but for a small glitch. There is a possibility of "man-in- the-middle attack". In this kind of attack the attacking node can replace its own public key in the place of other entity's key. As a result all entities send message using the fake key, there by the attacker will read all the messages by decrypting it with its secret key. In this attack, suppose A sends it public key to B. This is intercepted by an attacker C. C then sends a public key for which it has the secret key to B claiming to be C. Attacker C then decrypts the message. After reading it, the attacker will encrypt the message with the original public key of the destination node (the node that it attacked) and forward it to the recipient. There is no way for the entities in the network to know if the public key in the directory belongs to the actual entity or not. In order to solve this problem a PKI has (RA) registration authorities and (CA) certificate authorities .The RA and CA ensure that the entity's identity is bonded to its public key. Whenever a node wants to communicate in the network it needs to get a certificate of authentication from the RA and CA. First, the entity has to prove its identity to the RA, when the RA is satisfied with it, it will send a message to the CA that a certificate can be issued to authenticate the entity. The CA issues the certificate by signing it with its private key.

When 'A' wants to communicate with 'B' it will look at B's certificate, when 'A' finds that it has been signed by the private key of the CA, it believes the identity of 'B' and starts communicating as usual by encrypting the message with B's public key.

The RA and CA provide a trust among the nodes in the network; this is also called a trusted third party model [3].

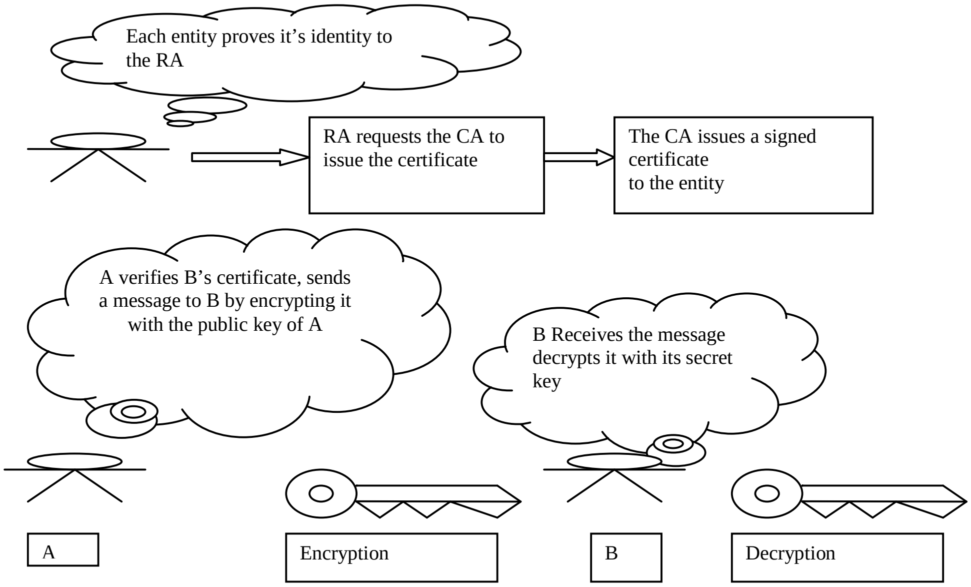

Each entity proves it's identity to the RA
RA requests the CA to issue the certificate
The CA issues a signed certificate to the entity
A verifies B's certificate, sends a message to B by encrypting it with the public key of A
B Receives the message decrypts it with its secret key
A
Encryption
B
Decryption

Figure 7: working of a public key infrastructure

3.3 Digital Certificate:

A digital certificate is a source of authentication for an entity to communicate in the network. A digital certificate also binds the entity's identity with its public key. A digital certificate also contains additional information to assure the other entities the identity of the public key owner. Finally, the digital certificate is signed by the private key of the CA.The various fields of a digital certificate are as follows:

- Version number: The version of X.509 standard used to create the certificate
- Subject: This describes the owner of the certificate
- Public Key: This will give the public key of the entity and also explains the algorithm used for creating the public/private key pair
- Issuer: The CA that issued the certificate
- Serial Number: A unique number given to identify each certificate.
- Validity: The time period for which the certificate is valid.
- Certificate usage: This indicates the various purposes for which the certificate can be used
- Signature algorithm: This indicates the algorithm used by the CA to digitally sign the certificate
- Extensions: The functionality of the certificate can be expanded by encoding additional data into the certificate.[3]

3.4 Certificate Renewal:

Every entity needs to get its certificate renewed. The same keys can be used for the next cycle of certificate revocation, but the various purposes for which it can be used may be modified. The certificate is renewed only if the entity is not present in the certificate revocation list (CRL). The certificate revocation list contains a list of nodes whose certificates have been suspended due to their malicious behavior in the network .They no longer exist in the network, for reasons aplenty. An entity whose certificate has expired doesn’t come under the revocation list .It is the responsibility of the CA to renew the certificate. CRL is maintained by the CA and it is the responsibility of the CA to update the list. Whenever an entity in the network wants to communicate with any other entity, it has to check if certificate of the opposite node is revoked. This can be done by requesting the CRL from the CA.

3.5 Types of Certification authorities (CA):

1. Centralized Infrastructure: The certification authority generates the public key for each entity; thereby it knows the private key of each entity. It gives the key to the requestor so that it can use its certificate.
2. Decentralized Infrastructure: In this mode every entity forms the key by itself. The certification authority doesn’t know the private key, and it is given the public key to build the certificate [3].

3.6 Threshold Cryptography:

In this section we describe threshold cryptography to create a decentralized CA. For achieving encryption, any encryption algorithm like RSA, symmetric encryption, asymmetric encryption can be used. Here the public and the private key’s are very important. With the help of the keys the original document can be recovered, but any of the above algorithms safeguard only the document but not the keys, hence if the keys are known the security is breached. Threshold cryptography is used to safeguard the keys, which in turn safeguard the documents that need to be protected.

The main idea behind “Threshold cryptography” is to break the key into several pieces and distribute among “n” entities such that the key can be recovered only by joining k or more of those pieces. The value k is called the threshold as we need at least k or greater number of pieces to reveal the secret.

In the following example, consider a company that digitally signs all its checks. For a check to be valid there should be three signatures. This is called a (3, n) threshold scheme. The secret company signature is divided into “n” pieces and divided among the employees. There is a signature generating device which needs three parts of the shared secret to sign the check. Hence any employee who wants to counterfeit, has to compromise other two employees which would be difficult. Hence this system is safe and secure, under the assumption that most of the employees can be trusted. [12]. Hence if there is a Data D and if it is split into “n” pieces and if “k” is determined to be its threshold

- The Data D can be reconstructed only with the knowledge of “k” or more pieces.
- If fewer than “k” pieces are known, then D cannot be revealed. [12]

The selection of threshold level “k” is very important and it can depend on the following factors:

- Extent of security desired.
- Type of application (for example, the entities are mobile or stable).
- The number of entities that are available all the time (in the case of mobile environment)

- The vulnerability of the system as well as the degree of trust on each entity.

3.6.1 Principles of threshold Cryptography:

A technique called Polynomial interpolation is used to share the secret among “n” entities. This method chooses a value for “k” which should obviously be less than “n”. This technique is devised in such a way that (k-1) entities cannot collaboratively reveal the secret and it needs at least “k” entities to generate the complete secret. [12]

Let D be the secret to be shared:

Consider a polynomial of (k-1) degree

$Q (x) = a [0] + a [1] *x + a [2]*x^{2 +} \ldots\ldots + a[k]*x^{(k-1),}$

D1, D2, D3….Dn, “n” number of shares are generated such that

$D1 = Q (1), D2 = Q (2), D3 = Q (3)\ldots Dn = Q (n).$

A prime number “p” is selected such that “p>D” and “p>n”. The coefficients in the polynomial are chosen randomly from a uniform distribution over the range [0, p). In order to construct D (complete secret) from its shares (Di’s) that are shared by “n” number of entities at least “k” number of Di’s are required. The coefficients in the polynomial equation can be determined by using “k” number of Di’s. Using the coefficients, the secret (a [0]) can be determined. [12]

Some of the important features of this technique are:

- By fixing the threshold level (k), the number of Di pieces can be increased or decreased. Hence dynamic addition or deletion of the shareholders for the secret is possible making it suitable for most of the applications. [12]

- The size of each share (Di) can never exceed the size of the complete secret (D). [12]

3.6.2 Applications of Threshold Cryptography:

Threshold cryptography is very suitable for enhanced security purposes. It is mainly used for Digital certificates, in which a user is said to be authorized only if he possesses a valid digital certificate. To sign a digital certificate a secret key is required, which can be shared using “Threshold cryptography”. A value of “k” can be selected based on the number of entities that will always be present, under the assumption that most of them can be trusted.

Hence this concept is used in Ad hoc wireless networks to authorize a node to communicate in the network. Each node needs to get a certificate from the certification authority (CA). The number of nodes in a CA is the threshold, which is selected based on the number of neighbors each node has, most of the time. Hence each node in the CA has a part of the secret signature and when all of them agree to sign the certificate a new node is authorized to communicate in the network. The node receives pieces of the secret key and when it receives “k” pieces it can construct a certificate. Threshold cryptography is very well suited for this application as the number of nodes on the network are not fixed and it works well under these situations as it enables dynamic addition or deletion of the shareholders for the secret. Threshold cryptography can also be used in public key cryptosystems.

3.7 Intrusion Detection Systems:

The above cryptographic systems are mainly for the authentication of the entities in the network. If a particular authenticated entity starts behaving maliciously, there is no way for the entity to protect itself. For this purpose intrusion detection systems (IDS) are implemented. An Intrusion detection system is like a burglar alarm in the physical world. This will safeguard the node from any intrusions and attacks .The host node can take the required steps to reduce the effect of the attack. An intrusion detection system maintains a log of normal behavior. When the host entity communicates with any other entity it monitors the behavior of the target node. If the pattern of its behavior is different from the predefined behavior the IDS signals the host system and the host takes the necessary action.

CHAPTER IV

TRUST PROBLEM IN AD HOC WIRELESS NETWORKS

4.1 Trust models:

Chapter 3 “Security in networks” emphasized the importance of security in a network. In order to ensure security in a wireless network each node in the network needs to be authenticated. Since, an ad hoc network has no infrastructure; a distributed trust model to authenticate the nodes in the network is required. The ad hoc network survives only by mutual cooperation among the nodes in the network, hence it is highly essential to have good trust relationships among the nodes in the network.

A trust model is said to be good, if it is able to handle as many threats as possible and if the underlying trust model is good enough to remove the malicious nodes in the network. A number of trust models have been proposed. This section gives an overview of those models.

Cr´epeau and Davis [13] proposed a trust mechanism where the nodes in the ad hoc network maintain profile tables which contain the details of the other nodes in the network including the nodes they have accused, and their behavior index. The behavior index is the measure of reputation of each node. If a node accuses a large number of nodes its behavior index or reputation is decreased, and as a result the other nodes don’t believe the accusations made by that node. Each time a node communicates it will check its own profile table with the profile tables of the other nodes and updates it. Each will request the profile tables of the other nodes in the network and update its own table if there are any new entries. The nodes being accused and the date of accusation must be consistent in all the profile tables. If any inconstancies are discovered accusations are launched against those nodes. The model suggested by [13] establishes good trust relationships, but the process of exchanging profile tables each time may be time consuming. It should be done in real-time on the fly. The behavior index only handles hijacking nodes, the adversary model is not extensive and it is not specified how the nodes are authenticated and what is done with the mistrusted nodes.

Capra proposes a model [14] where each node exchanges recommendation letters which describes the performance and satisfaction from the node in a specific context. This model emphasizes on context specific trust. Each node in the network communicates with other node by trusting a node blindly or if has any direct experience with it, it can also get recommendations from other nodes. At the end of communication the two nodes involved in the communication will exchange recommendation letters which describe their opinions on each other. The context of the trust is also specified in the recommendation letter. These letters are signed with the private key of the node issuing the recommendation. This is how the nodes built trust on each other, they initially communicate by trusting a node blindly later on they communicate based on their experience. When any node seeks recommendation about the performance of a particular node in the network the other nodes in the network pass on their recommendation which is also signed. After each communication the trust on the nodes is updated using a mathematical equation described in the paper. The trust formation, evolution and dissemination are all well described. Communication among the nodes is based on trust.

But, it’s not described how the model works under various threats; it’s all left to the fate of the node. Mistrusted nodes prevail in the network.

According to Buchegger and Boudec [15] the trust relationships can be effectively established using a Bayesian network model. Each node has some reputation which increases or decreases based on its performance in the network. High preference is given to first hand observations than the second hand observations or the information obtained from other nodes. When ever an accusation against a particular node is received, each node marks it as a second hand experience and compares this with other nodes in the network by comparing its Bayesian network with the Bayesian networks of the other nodes; if the node finds any inconsistencies the reputation of the accusing node is lowered. If the reputation of the nodes is lowered below a threshold the node is said to be decided (or convicted). The model safeguards the nodes from rumors .This model only specifies how the reputation of the nodes is formed. It doesn’t have a good adversary model and doesn’t specify how the behavior of the node is estimated.

The trust model by Anthony and Thompson [16] describes how the trust among the nodes is formed. Each node is classified under the following categories: compromised, unknown, Minimum, Medium, High, Highest (trust level).Each node picks a node to communicate based on the category. The nodes exchange the information in their local environment in regular intervals and update their information. This exchanging process is done only among the trusted nodes. Here each node has intrusion detection systems for detecting the threats, but the kind of threats that are handled has not been specified. It only mentions that the trust on misbehaving nodes has been reduced. The mechanism used for the authentication of the nodes in the network is not described.

In the model described by George and Baras [17] establishes a trust relationship among the nodes based on a theory of semirings.It is shown that two nodes can establish indirect trust relationships without direct interaction. The trust along the various paths of the network is evaluated, but how the good and bad nodes are isolated has not been mentioned.

Asad and Chris [18] proposed the trust model that is based on categories of trust; there are several categories that the model assumes they are passive acknowledgments, packet precision, gratuitous route replies, blacklists, salvaging information.

The trust on each node is developed as a result of interactions among the nodes, the trust is derivate based on the frames received, data packets forwarded, control packets forwarded.

The following issues have not been addressed:

- Trust decay/consolidation over time
- security analysis of the model against attacks
- Trust acquirement through malicious behavior

From the above discussion it is evident that each of the proposed approaches do not have a good adversary model or a good distributed trust model to authenticate the nodes in the network, They do not properly specify how the threats are handled and how the malicious node is handled. Hence, an effective trust model is required the model described in [19] is taken as the basis. The working of the model is explained in the next section. It is quite efficient but has several drawbacks which are examined. The major drawback is the trust relationships among the nodes. Hence a new trust model is proposed in the next chapter which addresses all the problems faced by this model and serves as an effective solution for providing security in an ad hoc wireless network.

4.2 Distributed trust model:

This trust model is also called a “Trusted third party model”. In this trust model, a node can communicate only if it is authenticated by “k” nodes. These are trusted nodes and they together form the Certification authority (CA). Each of the “k” nodes in CA has a part of the secret key for signing the digital certificates. When a node is new to the network it floods a certificate request message. When the “k” nodes see this, each one signs the certificate with the part of the secret key that it carries. Once all the “k” nodes sign, the certificate is complete. Hence the new node can start communicating. Here there are two important factors T_{cert} and k. T_{cert} is the time for which the certificate is valid. After this T_{cert} period of time, the node has to renew its certificate to continue communicating in the network. The value “k” is the number of nodes in the CA that can be decided in two different ways. In the first method, the value k is a fixed number. This method may not be very good, as there may not be “k” number of nodes at all times in the network. In the second method, the value of k is selected based on the majority, i.e. based on the number of nodes each one has in its neighborhood. This trust model will be referred as scheme1 in the entire document.

4.3 Dynamic Coalescing:

In this method when a node wants to get a certificate, it waits for a reply from “k” nodes. If it doesn’t get enough replies, it can move from its position and can get reply from the nodes over there. Normally the certificate is requested from the neighboring nodes i.e.

one-hop neighbors. If enough number of nodes is not present, then certificates can be requested from the next hop neighbor, which is called “Dynamic coalescing”.

When the certificate of a node is to be renewed, it is issued based on its behavior and is recorded in a table by each node in the network. [19]

4.4 Trust Model:

Each node maintains a table which looks as shown below:

Node ID	**List of Accusers**	**Comments**
A	B,C,D, E	Convicted
B	A	Suspect

Table 1: Certificate revocation list maintained by each node

If the number of accusers is at least k (no of nodes in the CA), then the node is marked as convicted If the number of accusers is less than k, then it is marked as a suspect.

Whenever a node observes that a particular node in its neighborhood is misbehaving, it will mark the node as convicted. When it sees this kind of a behavior, it will flood the message so that, other nodes in the network will make a change in their tables. The certificate is not issued to the convicted nodes.Hence, with the help of the above scheme we can ensure that only legitimate nodes exist in the network and any node that tries to misbehave will not have its certificate renewed.

4.5 Disadvantages:

1. The certificate revocation is not stopped until k nodes issue a convicted message. So until there are “k” entries in the list of accusers, the misbehaving node manages to get a certificate potentially inflicting considerable damage to the network.

2. There is no trust among the nodes i.e. if any particular node tries to misbehave, it deliberately floods information (that a particular node is not good) to other nodes and they make the corresponding entry in the table. No measure is taken to accept messages only from trusted nodes. There can be a problem here; an innocent node might be denied a certificate.

3. There can be certain nodes, which try to disturb the network just by issuing messages on the misbehavior of a particular node. No measure is taken to avoid hearing such possibly false malicious accusations. There should be a mechanism to differentiate the legitimate messages from malicious messages.

4. The nodes that try to misbehave (or those with malicious intentions) may create a slow and steady impact on the network i.e. they would behave well initially then slowly start causing availability problems or reducing the quality of transmission. This model has no parameter to measure trust based on availability or quality of transmission.
5. Here all the messages a node receives is a flooded message, so if it fails to receive the message, it will lose that valid information. This can lead to certificate renewal for a

malicious node. In this scheme, a node doesn’t seek recommendations on the performance of other nodes; it just listens to the flooded messages.

The trust on a particular node is a learning process and each and every interaction with a node should be weighted to track the behavior of the node. The above cannot be achieved with the trust model described above.

CHAPTER V

TRUST

5.1 Introduction:

In the trust model described earlier, there was no trust in the relationship among the nodes in the network. The communication was carried without any trust being established. This could be highly risky, especially when the transactions are critical.

If there is trust among the nodes in the network, the risk factor is considerable reduced, as nodes in the network communicate only with other trusted nodes. This minimizes the chances of being compromised or becoming vulnerable to attacks. Hence Trust is very critical, and is one of the important attributes in ad hoc networks.

5.2 Definition of Trust:

There is no precise definition for trust. Trusting is nothing but relying on some thing or somebody believing that it is legitimate in providing service, maintaining confidentiality of information, making correct use of available resources and keeping up the expectations. In other words, the main reason for having trust is to keep up good security. Hence Trust can be something that addresses the three important attributes of security namely, availability, confidentiality and integrity. While availability means the time for which a system will be available for use, integrity defines how reliable the

Information on the system is and confidentiality defines how well the information is protected from unauthorized access. [20]

5.3 Types of Trust:

There are various types of trust. They are as follows:

- **Self Trust:** Self trust is the trust in oneself or trust within the system.
- **Direct Trust:** It is the trust between two nodes, which is formed by direct experience.
- **Third party Trust:** A trusted node will take the responsibility for the formation of trust between two entities. The trust formed should be verifiable by the node relying on the trusted third party.
- **Proxy Trust:** It is a kind of trust that is formed based on a recommendation from another trusted node. [21]

5.4 Prevalence of Trust:

Trust is present everywhere in the world, starting from our daily routine to forming alliances among nations; trust is the foundation of any relationship in the world.

We would normally prefer to buy a product of a well known company to a new one and we would like to go to a movie of a popular star rather than of a debutant. But, the extent to which trust is required is important, since in every step that we take there is considerable risk involved. For trivial things like buying a coffee or going to a movie, trust doesn’t play a role. But in transactions involving money, trust is required .Therefore, in the case of ad hoc networks, if a node talks to an unknown node, there is some amount of risk based on the kind of transaction. If the transaction is very trivial, then the node can communicate with any node as there is no risk, but highly confidential transaction should take place only when good amount of trust is present, since confidentiality and integrity of the system is important. Moreover if the communications with mistrusted parties take place, there is a risk of being hijacked and compromised. From the above discussion it is evident that prevalence of trust is required for any transaction. However, the extent of trust required will be based on the amount of risk involved. As trust is very important, it is vital to understand the nature of trust.

5.5 Nature of Trust:

Trust is subjective, which means that each individual has a different perception of trust. This means that, if one trusts a company for timely transactions, another might trust it for the quality of production. Therefore the trust of the two individuals may conflict.

As trust is multifaceted, the context of trust is very important.

By introducing trust in ad hoc wireless networks, each node in the network will communicate with a node that it trusts. As trust can be multifaceted, trust of one node in other will be based on a specific context. For example if node A trusts node B for a good quality of service, it could mistrust it for availability. When a node is new to the network all the other nodes in the network are new to it. Hence there is no node that it trusts. Under such situations, it can ask for recommendations from other nodes. After the communication takes place, if the node that asked for recommendations finds the communication satisfactory, then its trust on the recommenders that recommended positive will increase, trust on the nodes the negative recommenders will decrease.

CHAPTER VI

PROPOSED SOLUTION

6.1 Trust management framework:

The task of the trust management framework is accomplished with the help of Bayesian Networks. Consider an ad hoc network shown in Figure 8:

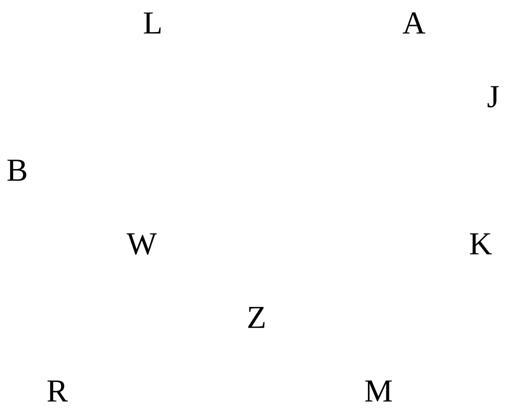


Figure 8: A sample ad hoc network

The above illustrates a set of nodes in an ad hoc network. In this network, when a new node enters the network, it requests a certificate from the certification authority. No node in the network can communicate without the certificate and the certificate is valid only for a specified period of time. All the basic steps in this model are the same as the trust model described in chapter 4.The special feature of this scheme is the introduction of trust in the model.

Any node in the network will communicate, if it has trust in the target node. Trust is built on each node in two ways, one is by direct experience and the other is through recommendations.

Furthermore, there can be parameters (i.e. context of trust) which influence the trust on the nodes. When any node gets a request for certificate renewal, it will be renewed only if the node is trustworthy. Every node has trust values for every other node in the network, if the trust value is greater than or equal to a threshold the node is trusted. For enhanced security of the nodes there is an Intrusion detection unit in each node for checking any attacks that may take place when it is communicating with another node. . When an Intrusion is detected, it signals the source node and the source immediately ceases communication and reduces its trust on the target node with which it is communicating.

The behavior of the target node may be perfectly legal and benign. However, due to interference or some other problems it may be suspected to be malicious. The source node therefore cannot take any strong decision as to if it is malicious and flood an accusation message that it is suspicious. This is because the main objective of this scheme is that “no innocent node should suffer and no bad node should survive”. Hence unless the target’s behavior is very obvious for it to be marked as malicious, the source node doesn’t flood any message against it. However, as it is suspecting it to be bad, it will reduce the trust on the target node. For accomplishing a secured network of this kind, we propose to implement the trust as Bayesian networks.

6.2 Motivation:

A Bayesian network is a relationship network that uses statistical methods to represent probability relationships between different nodes.

The main reasons for using Bayesian network are as follows:

A Bayesian network works with probability. Trust can be best represented using probability. This is because although it may not be possible to represent the trust between two nodes with absolute certainty, a probability measure of trust can be derived. In this scheme, an assumption of different contexts is made, hence when two agents are exchanging their opinions on a particular node there may be conflicts, if the contexts are different. The other reason for using Bayesian network is that the network learns from experience, the trust values get updated with each communication.

If the two parameters of trust are Availability and Quality of service and if a node wants to find out if a particular node is good at quality and also good at availability, these are the two contexts of trust. So combining both of them would be difficult. In such a scenario a Bayesian network would make things easy.

The naive Bayes theorem is as follows:

$$P(h \mid e) = \frac{P(e \mid h) * P(h)}{P(e)}$$

P (h) is called the prior probability of hypothesis and *p (e)* is the prior probability of evidence *P (h/e)* is the probability of *h* given *e*; *P (e/h)* is the probability of *e* given *h.*

Hence when all the values are supplied (i.e. prior probabilities) the Bayes theorem computes the posterior probability.

6.3 Working mechanism of the model:

A node in the ad hoc network has three different functions:

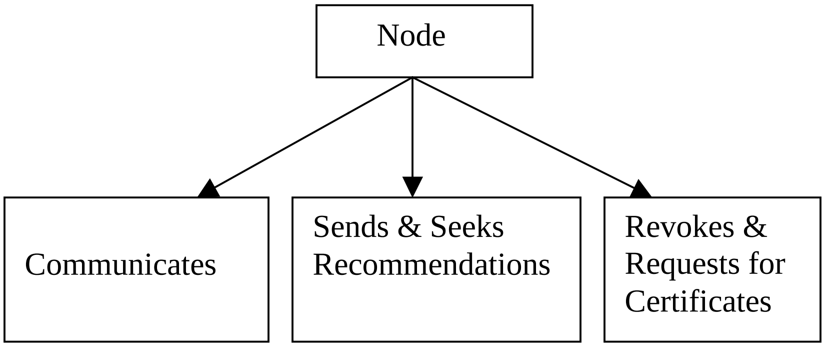

Node
Communicates
Sends & Seeks Recommendations
Revokes & Requests for Certificates

Figure 9: Flowchart describing the functions of a node

- **Communicating:** A node in the network communicates with other nodes based on trust.
- **Seek & send recommendations:** When a node wants to communicate with a node of which if it has no direct experience or if it has little or outdated experience it will seek recommendations from other nodes in the network. It will also send recommendation if it receives any requests for the same from other nodes in the network.
- **Revokes & requests Certificates:** Any node in the network can communicate only for the period the certificate is valid. After that, it needs to be renewed. Hence it sends a request for certificate and if a threshold of nodes replies to the request, it can then form its certificate. It also has to revoke a certificate if it receives a request .A certificate is revoked only if the node is trust worthy.

6.4 Communication of the node:

Before understanding the details of a node's communication, it is important to understand the local environment (i.e. information that a node maintains) of a node.

For the sake of building trust on each node in the network, each node maintains a Bayesian network for every other node in the network.

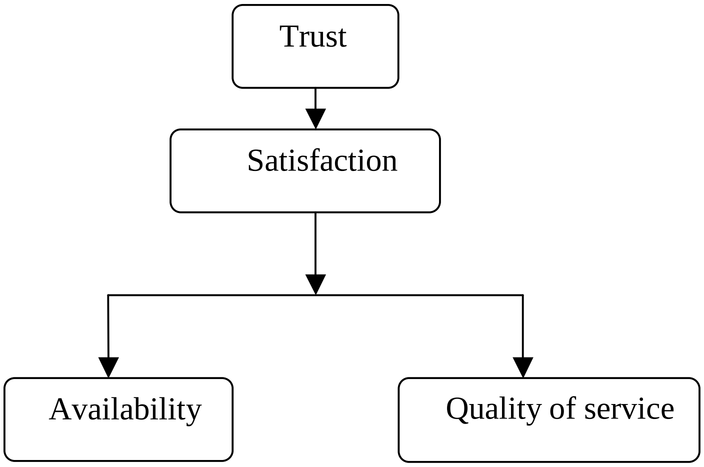


Figure 10: Bayesian Network model

The two parameters of trust are:

1. **Availability:** This describes how long the target node is available for service, an important attribute of security.

2. Quality of service: The meaning is quite in evident that it describes the quality of service of the target node. This is also a very important parameter of trust. This is because if the quality of service is poor, there is wastage of communication and also excessive loss of battery power that is very scarce and needs to be conserved in an ad hoc network.

Again as described earlier each node has it own perception of trust; hence each node is free to give its own weight for the two parameters. The weight could be any value between [0, 1].

Once a node communicates it records the target nodes performance and calculates satisfaction using the following mathematical expression:

$satisfaction(S) = W_Q * S_Q + W_A * S_A$ eq. (1)

Where

$W_{Q:}$ weight for Quality of transmission

$S_{Q:}$ Satisfaction from the Quality of transmission

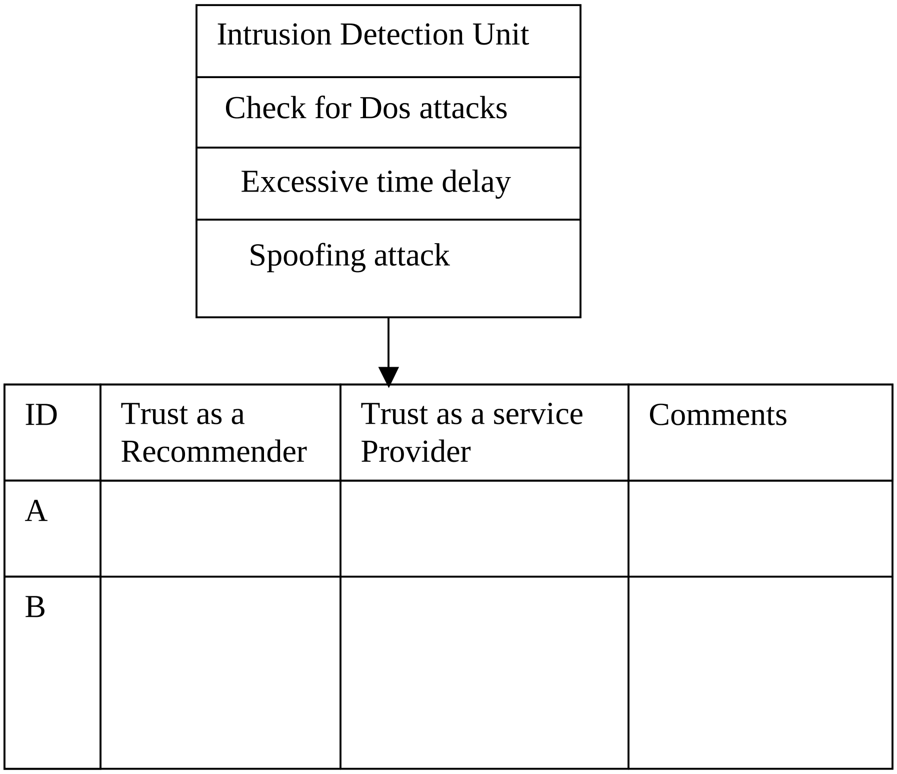
$W_{A:}$ weight for availability

$S_{A:}$ Satisfaction from availability

If this value is greater than or equal to threshold then the node is trustworthy. Satisfaction is nothing but the trust on the node. All this information is stored in the local environment of the node.

6.5 Local environment of a node:

Trust can be stored only if the node has a direct experience. If it has no direct experience it can seek for recommendations, hence a node in an ad hoc network acts as a service provider as well as a recommender. Therefore two Bayesian networks are needed for a single node, one to represent its trustworthiness as service provider and the other as a recommender. This is illustrated in Figure11.



ID	Trust as a Recommender	Trust as a service Provider	Comments
A			
B			

Figure 11: Local environment of a node in an ad hoc network

It is important to note the devices that operate in an ad hoc network are battery constrained and we therefore cannot always make an assumption that when a request is sent, it will receive a reply. Usually nodes in the network can be selfish. In order to save their battery power they may not respond to all the requests that they receive.

Hence taking this into consideration there may be a situation, where a node is unable to form a good trust level on a particular node in the network even though the node may not be untrustworthy. Hence to overcome such a situation, the following is done. When two nodes are communicating in the network, at the end of the communication, they exchange a “portfolio of credentials” which means that each node will receive a recommendation letter from thenode it has communicated, which will be as follows:

<*a, b, c, t*> which means that *"a"* trust's*"b"* in the context *c* and at a time stamp *t*. [14] This letter will be signed by the private key of the recommender, so that any node will not make fake portfolios. Hence when a node encounters a situation where very few number of nodes reply when it seeks for recommendation, it would request the target node for its portfolios and make a decision based on the portfolios it receives. For security reasons, when any node sends a recommendation in the network, it should be signed by its private key to avoid repudiation.

6.6 Intrusion detection unit:

Intrusion detection systems are security systems just like burglar alarms in the physical world. They keep a close eye on the network, and watch for abnormal activities on the network. It has a pattern of normal behavior, so whenever an activity takes place, it matches the pattern to its database and if it finds any deviation, it alarms. Intrusion detection systems are now becoming very popular security mechanisms. The Intrusion detection unit is added to each node in the network as an extension to Scheme 1 for enhanced security in the ad hoc wireless network. [22]

The intrusion detection unit checks for 3 attacks:

- DOS attack
- Excessive time delays
- Spoofing Attack

1**. Large no of continuous requests (DoS attack):**

There is a threshold on the number of requests a node sends. Once the threshold is exceeded, the intrusion detection unit will alarm. The assumption here is that when any two nodes are communicating in the network, if any of the two nodes sends a recommendation letter (the letter that is to be issued at the end of the communication), the other node should stop sending messages and understand that the communication has ended thereby the trust on the misbehaving node will be reduced. Although this may not stop the node from sending more messages, it will know that the other node has reduced its trust level for the node.

2**. Time of response:**

There should be a minimum waiting time for a response, when two nodes are communicating and if one of the nodes takes an unexpectedly long time to reply then the Intrusion detection unit will beep and communication will end. That node might purposefully try to delay communication to conserve battery power for low priority communication or the delay may be due to network congestion. Irrespective of whether the node purposely or otherwise reduces communication, trust is decreased for the node.

3**. Spoofing:**

In a spoofingattack when two n odes are communicating, one of the nodes may change the source IP address field in the IP packet. [23]. When the communication ends, the nodes have to exchange recommendation letters, which need to be signed by the private keys. When the Intrusion detection unit finds from the signature that spoofing has occurred, it alarms and the trust on the node is reduced and an accusation message is flooded in the network. Once such a message is flooded, the source node marks the target node as convicted in the comments field of the corresponding node ID in its local environment.

6.7 Communication of a node under different scenarios:

There are three different scenarios of communication with another node:

1. When there is direct experience with the other node

2. When there is no direct experience or if the node doesn't have enough information to make a decision

3. When nodes are selfish and very few nodes respond to the request for recommendation

When a node wants to talk to a node, it should have some kind of trust on it. Otherwise, it can't communicate, as it doesn't want to end up in wasting its resources doing a useless communication. Hence, if the node had ever communicated with the target directly it will have some kind of Trust on it. If it had no direct experiences, it will seek for recommendations from the rest of the nodes in the network.

1. **Direct experience:**

When a node has some direct experience with the target, it measures the performance of the target and the amount of satisfaction it gets after communicating with it, by using its own weight for the parameters as stored in the local environment. If the trust level is greater than or equal to threshold then it communicates, else it picks up some other node and repeats the same step. An important factor is the time of the experience with the other node. If it is outdated, it has to neglect its experience and seek for recommendations (explained below) that are fresher. At the end of the communication both the nodes exchange a portfolio of credentials.

2. **Absence of Direct Experience:**

When there is no direct experience a node seeks recommendations from other nodes. Different nodes in the network might respond to such queries, out of which some nodes could be already known nodes (the nodes with which the host node communicated and has trust values for the nodes), some of which, are trusted nodes and some are mistrusted nodes. The, recommendations from known (trusted nodes) are given more weight out of which the more recent information is given more weight. The latest information from unknown nodes is given less weight.

Once these recommendations are taken, a decision is taken whether communication can proceed or not. Once the communication is complete, the nodes exchange recommendation letters. On completion if the communication was satisfactory the trust on the nodes that recommended positive is increased and for the others it is decreased.

3. **Handling Selfish nodes:**

When a node doesn’t have a direct experience or if the trust values are outdated it seeks recommendations from other nodes. Some of these nodes may not respond to the requests if they are selfish in view of saving their battery power. Under such situations the following is done: The node wishing to communicate can ask the target node for recommendation letters that it has got so far, in order to decide whether it can communicate with it. Then the source verifies the time stamp of the recommendation letter, and also the source of the letter and if the source is trust worthy, then it calculates the trust. If it is greater than or equal to the threshold, the source can communicate.

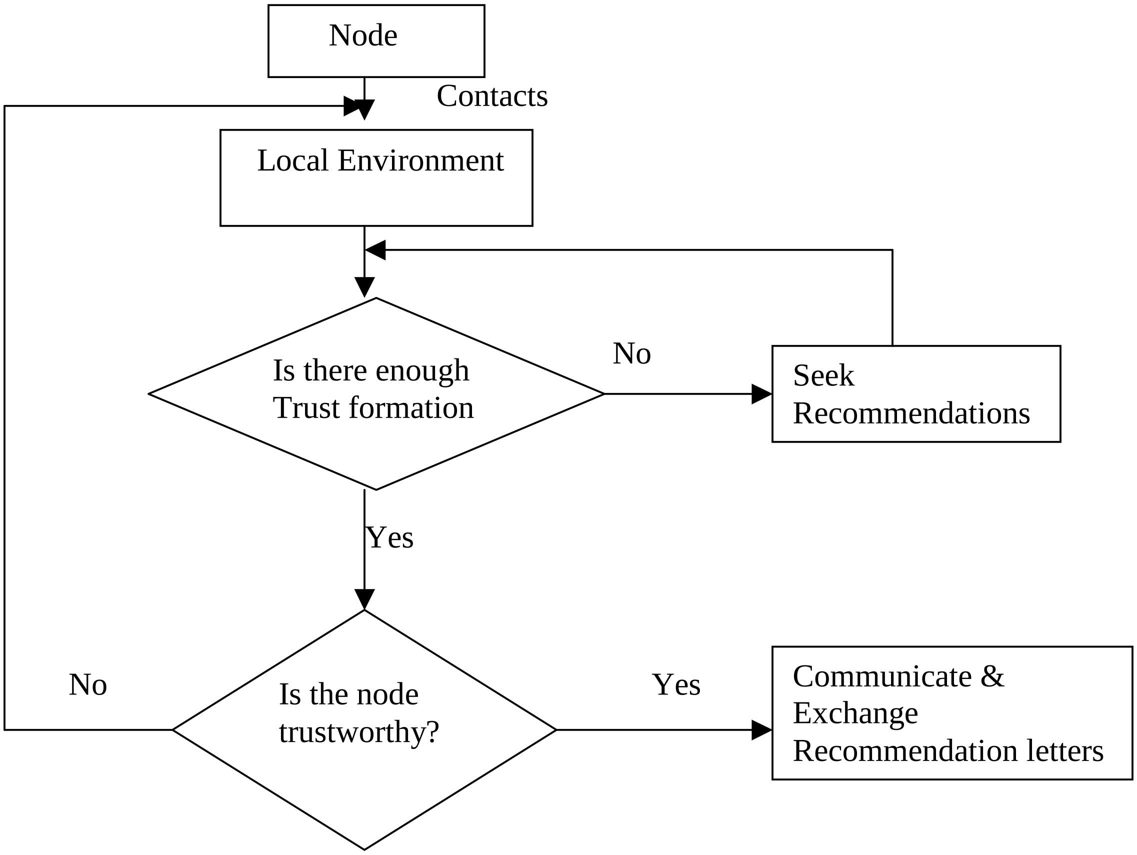


Figure 12: Flowchart for a node communicating in a trust environment

6.8 Sending & seeking recommendations:

In an ad hoc network at any point of time, a node may have to request for recommendations on a particular node or it may have to send a recommendation. Each node maintains the following table in its local environment

Node Id	Availability	Quality Transmission	No of Interactions				Satisfaction	Trust	Time stamp
			Total	Sa	Sq	Si			
A	20%	80%	3	1	1	1	0.35	0	20 sec
B	100%	90%	5	5	5	5	0.95	1	40 sec

Table 2: A table in the node's local environment

The time stamp indicates the time with respect to the time the certificate was issued. 20 seconds implies 20 seconds after the certificate was issued.

Sa- No of interactions in which the availability was satisfactory

Sq- No of interactions in which the Qos was satisfactory

Si- No of interactions that was satisfactory

Satisfaction is calculated using the formula described earlier

$$satisfaction(S) = W_Q * S_Q + W_A * S_A$$

Where

$W_{Q:}$ weight for quality of transmission

$S_{Q:}$ Satisfaction from the quality of transmission

$W_{A:}$ weight for availability

S_A : Satisfaction from availability

If satisfaction is greater than or equal to threshold *St* Then the trust is set to 1 otherwise it is 0, a node will communicate with a node whose trust is non-zero.

Each node does the following calculation when it has to send the recommendation.

Conditional probability is used to find the satisfaction from availability.

This is represented as follows:

P (Attribute=”Availability” /T=1) = P (Attribute=”Availability”, T=1)/ P (T=1) eq. (2)

Where
P (T=1) = m/n

P (Attribute=”Availability”, T=1): no of interactions in which the attribute availability was satisfactory.

m: no of satisfying interactions

n: total no of interactions

The same is done for quality of service

P (Attribute="QOS" /T=1) = P (Attribute="QOS", T=1)/ P (T=1) eq. (3)

P (T=1) = m/n

Where

P (Attribute="QOS", T=1): no of interactions in which the attribute QOS was satisfactory

m: no of satisfying interactions

n: total no of interactions

When the recommendations are sent, a trust value is sent, which is the probability that the node provides good availability and is trustworthy in providing quality of service.

Mathematically,

P (T=1/ Attribute="Availability") and P (T=1/ Attribute="QOS") needs to be calculated

This can be obtained by naïve Bayes theorem

$$P(h|e) = \frac{P(e|h)*P(h)}{P(e)}$$

P (h): No of availability or Qos cases satisfying / total no of interactions

P (e): No of Satisfying interactions/ Total no of interactions

P (e/h) is obtained form equation 2 or 3

This value is sent by the node as a recommendation, the timestamp of the observation is also sent. The recommendation letter is signed by the private key of the node issuing the certificate. This is called "Trust Dissemination".

The following evaluation is done by the node that receives the recommendations:

$$r_{ij} = w_{t1} * \frac{\sum_{l=1}^{k} tr_{il} * t_{lj}}{\sum_{l=1}^{k} tr_{il}} + w_{t2} * \frac{\sum_{l=1}^{k} tr_{il} * t_{lj}}{\sum_{l=1}^{k} tr_{il}} + w_{s1} * \frac{\sum_{j=1}^{g} tz_{j}}{g} + w_{s2} * \frac{\sum_{j=1}^{g} tz_{j}}{g} \quad \text{[24]} \quad \text{eq. (4)}$$

Where

r_{ij} = the total recommendation value for the j^{th} node that i^{th} node has

k= no of trustworthy references

g= no of unknown references

tr_{il} = the trust that i^{th} node has on l^{th} recommender

tr_{lj} = the trust that l^{th} node has on j^{th} node

t_{zj} = The trust that z^{th} unknown node has on j^{th} node

w_{t1} = weight given to known references (latest)

w_{s1} = weight given to unknown references (latest)

w_{t2} = weight given to known references (old)

w_{s2}= weight given to unknown references (old)

More weight is given to trusted reference and then to unknown references in which fresh recommendations are given more weight then old ones

$$w_{t1} + w_{s1} + w_{t2} + w_{s2} = 1$$

$$w_{t1} > w_{t2}, \ w_{s1} > w_{s2}$$

$$\text{Where } w_{t1}, w_{t2} > w_{s1}, w_{s2}$$

This is done twice once for availability and then for quality of service.

Then satisfaction is calculated and if it is greater than or equal to threshold communication takes place.

This is called "Trust Formation." After the communication takes place, the trust on the recommenders needs to be updated and the trust on the target node needs to be updated as a service provider. This is called "Trust evolution", which is done using the following formula

$$tr_{ij}^{\ n} = \alpha * tr_{ij}^{\ o} + (1-\alpha) * e_{\alpha}\ [25] \quad \text{eq. (5)}$$

Where

$tr_{ij}^{\ n}$ = new trust value that the i^{th} agent has in the j^{th} reference after the update

$tr_{ij}^{\ o}$ =denotes the old trust value.

α = learning rate – a real number in the interval [0, 1].

e_{α} is the new evidence value, which can be -1 or 1. If the value of recommendation is greater than threshold and the interaction with the node was satisfying, e_{α} is Equal to 1 otherwise it is -1.

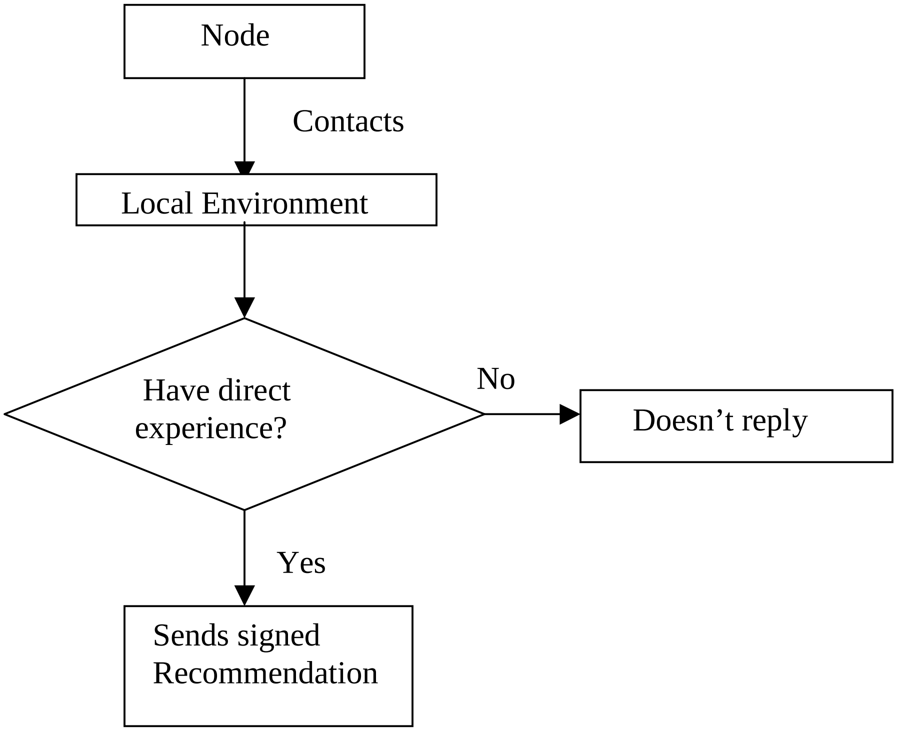


Figure 13: Flowchart describing how a node sends recommendations

6.9 Certificate Revocation:

In the ad hoc network each node communicates only for the period the certificate is valid, once the certificate expires, it has to be renewed. It should get a reply from at least k (threshold) nodes to continue its communication in the network.

When a node receives a request for certificate renewal the procedure as shown in Figure 14 is adopted.

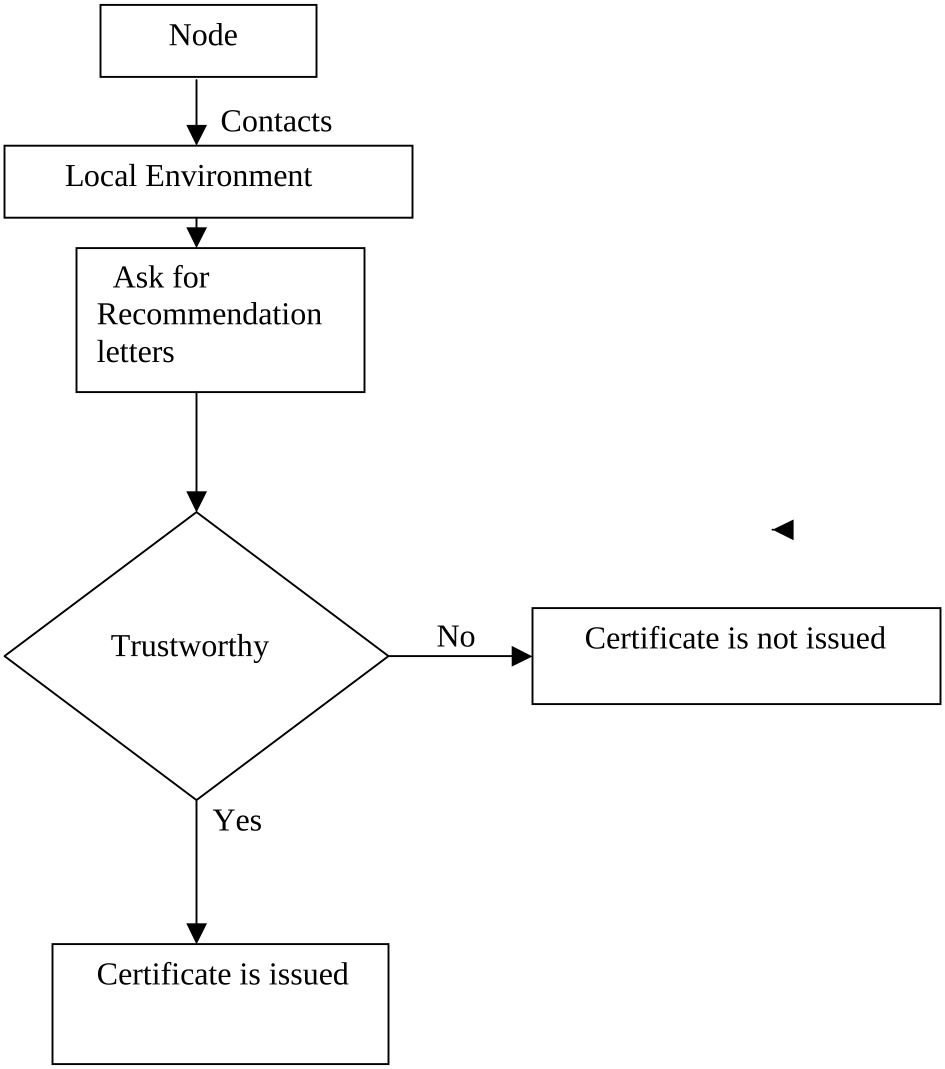


Figure 14: Flow chart describing certificate revocation

When a certificate is to be renewed a node requests the other nodes to renew the certificate. The requested nodes reply to the requesting node with a request for a “Portfolio of credentials”. This “Portfolio of credentials” determines the overall behavior of the node in the network during a certificate validity period.

Suppose if A’s certificate needs to be renewed, it sends its request to all the other nodes then the node J on receiving its request, requests for all the recommendation letters A has got. J gives more importance to the fresh ones. It adds up all the trust values, if the trust is greater than or equal to threshold, then certificate can be given. J gives a partial certificate to A. A sends similar request to other nodes in the network, if it is able to get k or more partial certificate it can form a valid certificate to communicate in the network.
Some times the node may have one or no recommendation, as it has not participated in communication. In this case the host node checks its local environment to verify if there is any information about the target nodes misbehavior. If the node has misbehaved certificate is not issued.

CHAPTER VII

SIMULATIONS AND RESULTS

The objective of the simulation is to compare the proposed Bayesian Trust model for certificate revocation with the approach described in [19]. The simulation will aim to show that as trust is gradually built up in the proposed model, fewer innocent nodes will have their certificate revoked and more malicious nodes will have their certificates revoked compared to the approach taken in [19]. The simulation for this work is performed in the 'C' programming language on Microsoft.Net platform. The complete source code is listed in the Appendix. An ad hoc network with nodes moving at random speeds and directions is simulated using the Random waypoint mobility model.

The following attacks are simulated in our work:

- DoS attacks
- Spoofing
- Time delay

Availability and Quality of service parameters are included in the simulation. The performance of the proposed scheme and the scheme described in chapter 4 are compare under the influence of attacks. In particular the following comparison is made for the two approaches:

- The time taken to remove a malicious node from the network

- The number of innocent nodes suffered by the policies used
- The amount of useful communication performed by the network the useful communication is limited by the certificate availability, quality of service and availability of the nodes in communication.

The proposed scheme and the scheme described by [19] are implemented separately under the same environment and the performance is compared.

The rest of this chapter gives an in-depth description of the simulation and the comparison of the two schemes.

7.1 Ad hoc network model:

For our simulation purposes, the ad hoc network model consists of a mixture of genuine and malicious nodes. Each node in the network has a unique ID. The nodes are placed at random positions in the simulation area. As this simulation deals with certificate revocation lists, the nodes in the network are assumed to have entered the network and have acquired a certificate for communication. Each node has a destination to reach, and each node is allocated some random traveling time and speed. Once a node reaches a position at the end of its travel time, it will pause for some time and this is called the pause time of the node, during which the node communicates with the other nodes or requests a certificate renewal, floods accusations etc. Each node has its own random pause time. The threshold value ‘K’ is globally fixed. The transmission range of the nodes is also globally determined. Each node has its own file to transfer the size is chosen randomly and every node is assigned a random download speed.

When two nodes communicate, the initiator of the communication will try to transfer a file which is received by the destination. The file size transferred will be based upon the pause time, download speed, certificate validity of the nodes in communication.

Hence, each node is also assigned a file to transfer at a particular download speed. When a node wishes to communicate with a particular node in the network, it has to send a request to the node and wait for the reply, where each node has its own reply time. The simulation is characterized by a cycle of communication, which means that each node in the network is given a chance to travel to a new position .After reaching a position it can flood accusations or based upon its certificate validity it will communicate with other nodes in the network. If the certificate has expired it will send request for renewal of the certificate. These sequences of events constitute a cycle of communication. The simulation is repeated for a number of cycles of communication to analyze its performance.

The network model remains the same for the simulation of the two schemes except for some additional parameters used in the simulation of the Bayesian network trust model. They are as follows:

- Each node has its own weight for the two parameters of trust (availability, quality of service).
- The threshold for satisfaction from a node (this is globally fixed).

7.2 Implementation of a distributed trust model:

The following steps describe the implementation of the trust model described in [19].

1. Initially, each node in the network travels to a random position in the simulation area based on its speed and time of travel. After reaching a destination, the node scans for its neighbors or surrounding nodes. Only the nodes that are within the transmission range of the node are assumed to be the neighbors of the node. Hence, every node will maintain a list of its neighbors each time it reaches a new position.
2. Each node has a destination to reach. Once it reaches its destination it picks up a new destination.
3. The node will monitor its surroundings to check if there is any malicious activity taking place. If it observes any such activity it will flood accusations to all its neighbors (these are one-hop neighbors). Each node maintains a "Certificate Revocation List" In the list it maintains a record of accusers who accused each node. When the no of accusers is at least k the node is marked as convicted otherwise it is marked as a suspect. These are called comments in the CRL. This is done in the CRL of the accuser as well as its neighbors. If the accused node has accused some other nodes in the past, all those entries are cleared and the comment on each node is updated.
4. The node selects a nearest neighbor to communicate. Once the neighbor is chosen it will wait for a reply from it. Once it gets a reply it will start sending the file that it wants to transfer. The file transfer will continue as long as the certificates of the two nodes are valid. Suppose if 'A' is communicating with 'B', 'B' can receive

the file only for the time the certificate of ’A’ is valid; similarly ’A’ can send a file to ‘B’ as long as ‘B’ has a valid certificate. This is again dependent on pause time; the pause time of the nodes cannot be greater than their certificate validity periods.

5. Based on the above factors the useful communication performed is calculated as follows:

   Actual file transfer: The actual file size that node ’A’ is capable of transferring is based on its certificate validity period and pause time.

   File transfer performed: The file size that ‘A’ is able to send is calculated as

   Time for communication* download speed of the receiver.

   The time for communication is based on the pause time of the receiver which cannot be greater than its certificate validity.

   File transfer performed = Time for communication* download speed of the receiver.

   This file transfer performed by the next set of communicating nodes is added to this value.

   Actual file transfer: Pause time of ‘A’* Download speed of ‘A’

   From these two factors (list the factors) the percentage of useful communication is calculated. Percentage of useful communication = File transfer performed / Actual file transfer *100.

6. If a node has no neighbor to communicate it will not perform any file transfer. It will wait for the next cycle of communication during which it can move to a new

position in the search of new neighbors to communicate. If it still doesn’t have any neighbors it will continue to move to a new position.

7. When node ‘A’ picks up its nearest neighbor, it has no special factors for making its choice except for the distance; it doesn’t have any kind of intrusion detection system to judge the target node. Hence if the receiving node is a DoS attacking node it will perform a DoS attack on A. Consequently, A cannot respond to any requests from other nodes and the node A will continue to be under the DoS attack as long as it is in the proximity of the attacker. A node under a DoS attack cannot communicate and this will affect the percentage of useful communication in the network.

8. When a node tries to communicate with a DoS attacked node it will not get a reply from it. Hence, for the entire pause time the node will be waiting for the reply from the target node. This will affect the percentage of useful communication

9. Hijacking attack: In this kind of an attack, a malicious node chooses a victim to falsely accuse in order to remove it from the network, so it compromises the node with which it communicates together the compromised node and the malicious node accuse an innocent node. The Hijacking node repeats this until the victim is removed from the network. This kind of an attack is common in the networks where the threshold value k is fixed. If the Hijacking node can compromise k

nodes it can successfully remove the victim from the network. Even if the value of k is dynamic it is not difficult for a node in the network to determine it.

Every node maintains a record of all the nodes it has accused and the reason for accusing. The intrusion detection unit before every cycle of communication checks the list. If for any accused node there is no reason listed then the intrusion detection unit assumes that the host node was compromised by the previous node with which it had communicated. Hence the source node floods accusation against the hijacker.

10. There are many other malicious activities like spoofing, time delay attacks where the target node purposely provides low quality of service. These attacks are undetected in this trust model.
11. A node can communicate in the network only if it has a valid certificate. If the certificate expires, the node floods a request for certificate renewal to its one hop neighbors. The neighbors who receive the request will check if the requesting node is convicted or not and if it is not convicted they reply otherwise they do not reply. The requesting node will collect all the replies and if the replies are at least k, it forms a certificate otherwise it is removed from the network. If a node couldn’t get k replies due to insufficient neighbors it moves to a new location to get more replies.

7.2.1 Advantages:

- The memory requirements are not very high, since each node maintains only its list of neighbors and a certificate revocation list.
- Each node picks a node to communicate based on the distance. Hence, much of the computational capacity of a node is not utilized.
- Due to the above two reasons the battery usage is not very high.

7.2.2 Shortcomings:

- There is no trust relationship among the nodes. Each node accepts a flooded message from any other node in the network. Due to this, a large no of innocent nodes could be removed from the network due to wrong accusations against them.
- Each node in the network is not equipped with an Intrusion detection mechanism to protect itself from DoS attacks, spoofing or time delay attacks.
- The trust model has no mechanism to remove the DoS attacking nodes and hijacking nodes. They are removed only if any hijacking node hijacks DoS attacking node or another hijacking node. Hence, the malicious nodes are removed by a chance of luck.
- The attacks like spoofing, time delay are undetected by the trust model.
- Unless k nodes realize that a node is bad , it is not removed from the network, this delay can cause considerable damage to the network

7.3 Implementation of a Bayesian network trust model:

The following steps describe the implementation of the Bayesian network trust model:

1. Initially, each node in the network travels to a random position in the simulation area based on its speed and time of travel. After reaching a destination, the node scans for its neighbors or surrounding nodes and only the nodes that are within the transmission range of the node are assumed to be the neighbors of the node. Hence, every node will maintain a list of its neighbors each time it reaches a new position.
2. Each node has a destination, once it reaches its destination it picks up a new destination.
3. Each node maintains a table of information about its neighbors. The table has several fields namely, the no of interactions among the nodes, no of interactions in which the availability was satisfying, interactions in which the quality of service was satisfying, total satisfaction from the node, trust on that node and comments, satisfaction as a recommender. The comments field is used to make an entry if there are any accusations against it. The time stamp of the latest experience is also recorded.
4. Each node in the network communicates based on trust, which means that a node will communicate with a node in the network only if it has some measure of trust on it. Suppose if a node wants to communicate with a mistrusted node it will get an opinion from a set of nodes that it trusts. If they trust the node, this node will also trust the node. At initialization, during the first cycle of communication each

node is assigned some trust on its neighbors. This is purely for startup as the nodes communicate based on trust.

5. After the two nodes communicate, they exchange their opinions on each other using recommendation letters which describe their trust and satisfaction with each other. The timestamp is also recorded and the recommendation letter is signed by the private key of the node.

6. During the communication the node will monitor its surroundings to check if there is any malicious activity taking place by sensing the routes of communication in the network. If it observes any such activity it will flood accusations to all its neighbors (these are one-hop neighbors).

   When accusations are flooded in the network, a node will believe the message if it trusts the accusing node, otherwise it ignores the message. If it trusts the message it marks the accused node as convicted.

7. When a node wants to communicate, it selects up the node that it trusts the best. The selection is based upon the weight the node gives for availability and quality of service .Hence the node tries to pick a node that has the best value for the required parameter. Then the satisfaction is computed using the equation1 described in chapter 6. If the satisfaction is greater than or equal to threshold (a global parameter) then the node checks if the target node is convicted. If it is convicted it will select the next best node and checks if there has been any direct experience. If the experience is outdated or if there is no experience, the node will seek recommendations. Then it will evaluate trust based on the equation 1

described in chapter 6 and if the trust is within the threshold value the node communicates otherwise it will pickup another node for communication.

8. There could be a situation where a node may not get enough recommendations, if the nodes are selfish and they want to save their battery power. Under such situations when there are less then the threshold of k replies, the node requests the target node for the recommendation letters it has got and based on those recommendation letters the node will decide whether to communicate with the target node or not.
9. When a node receives a request for recommendation about a node it will decide whether or not to reply, if it wants to reply it will check if it has any direct experience with the node and if there is direct experience it will send its recommendation signed by its private key to avoid non-repudiation.
10. When a node communicates based on the recommendations that it has obtained from the other nodes, it has to update its trust on the recommenders based on the satisfaction from the communication. If the communication was satisfying the trust on the positive recommenders is increased, while the trust on the negative recommenders is decreased and vice versa. This is done by using the trust evolution equation 5 described in chapter 6

11. Once the neighbor is chosen for communication, the node will wait for a reply from it. On getting the reply it will start sending the file that it wants to transfer. The file transfer will continue as long as the certificates of the two nodes are valid. Suppose if ’A’ is communicating with ‘B’, ‘B’ can receive the file only

until the time the certificate of ’A’ is valid, similarly ’A’ can send file to ‘B’ as long as ‘B’ has a valid certificate. This is again dependent on pause time; the pause time of the nodes cannot be greater than their certificate validity periods.

12. Based on the above factors the useful communication performed is calculated as follows:

    Actual file transfer: The actual file size that node ’A’ is capable of transferring is based on its certificate validity period and pause time.

    File transfer performed: The file size that ‘A’ is able to send is calculated as

    Time for communication* download speed of the receiver.

    The time for communication is based on the pause time of the receiver which cannot be greater than its certificate validity.

    File transfer performed = Time for communication* download speed of the receiver.

    This file transfer performed by the next set of communicating nodes is added to this value.

    Actual file transfer: Pause time of ‘A’* Download speed of ‘A’

    From these two factors (list the factors) the percentage of useful communication is calculated.

    Percentage of useful communication = File transfer performed / Actual file transfer *100.

    Each node keeps track of the number of requests it receives during communication and if the no of requests exceed a threshold of requests, then the intrusion detection system of the host node will signal a DoS attack .As a result, the node will flood an accusation in the network. This is how DoS attacking nodes are handled.

13. Hijacking attacks**:** In this kind of an attack, a malicious node chooses a victim to falsely accuse in order to remove it from the network, so it compromises the node with which it communicates together the compromised node and the malicious node accuse an innocent node. The Hijacking node repeats this until the victim is removed from the network. This kind of an attack is common in the networks where the threshold value k is fixed. If the Hijacking node can compromise k nodes it can successfully remove the victim from the network. Even if the value of k is dynamic it is not difficult for a node in the network to determine it.
14. Every node maintains a record of all the nodes it has accused and the reason for accusing. The intrusion detection unit before every cycle of communication checks the list. If for any accused node there is no reason listed then the intrusion detection unit assumes that the host node was compromised by the previous node with which it had communicated. Hence the source node floods accusation against the hijacker. Each node will make a note of the nodes that it has accused and the reason for accusation. The reasons can be a DoS attack, spoofing, or a message from a trusted node. The node with which it communicated in that cycle of communication is also noted. No entry is made if a node is accused as a result of being compromised by a hijacking node. Hence it will accuse the node with which it had previously communicated.

15. A node attempting a spoofing attack changes its ID during communication, so the other node believes it is communicating with a different node. But, at the end of communication the nodes exchange signed recommendation letters and when the opposite node discovers that the signature of the source node doesn't match its ID, the intrusion detection unit signals that it is spoofing attack. The destination node will therefore flood accusations against the source node.
16. Other nodes may try to uncooperative like providing low quality of service or time delayed replies. These attacks are automatically handled by the trust values built by the nodes in the network. The trust level is constantly updated and when the trust values reach a very low value they are removed from the network. However, it is possible that a particular node may appear to have behaved badly due to some node being uncooperative or causing disturbance in the network. , In order to give consideration for such nodes each node is given k chances to prove its performance and if it fails it will be removed from the network.

17. A node can communicate only if it has a valid certificate. If its certificate expires, the node floods a request for certificate renewal to its neighbors that are one hop away. The neighbors who receive the request will check if the requesting node is convicted or not. If it is not convicted and if they trust the node requesting the certificate, they reply, otherwise they do not reply. The time stamp of experience is given importance. .If node 'B' is requesting for a certificate from 'A','A' will reply only if it has a previous direct experience which is not outdated. Otherwise it will request 'B' for all the recommendation letters that it has received from the

other nodes in the network. Based on those letters ‘A’ will decide whether to reply or not. The same procedure is followed when there is no direct experience. The requesting node will collect all the replies and if the replies are at least k it forms a certificate; otherwise it is removed from the network. If a node couldn’t get k replies due to insufficient neighbors it moves to a new location to get more replies.

7.3.1 Advantages:

- This trust model handles various kinds attacks including DoS, spoofing, hijacking and Time delay attacks
- The number of innocent nodes suffered are minimal, since the hijacking nodes are removed by the trust model
- The malicious nodes are removed as soon as there is an accusation against them .There is no need to wait until k nodes report the accusation as in scheme1 described in chapter 4
- The model is more realistic as it considers the selfish nature of the nodes. This takes into account the situations where nodes don’t reply to save their battery power.
- The real essence of trust is brought by this model by making the trust context specific. This model has two contexts, availability and quality of service
- The percentage of useful communication will be more dominating compared to scheme1 as each node selects a node that has good availability and Qos to communicate.

7.3.2 Shortcomings:

- Each node needs to maintain a lot of information including its own recommendation letters. This will require more memory than scheme1.
- The computational requirements are higher as each node has to calculate the trust values, satisfactions etc.
- This scheme may therefore result in increased consumption of battery power.

7.4 Simulation assumptions:

The following are the assumptions made for the simulation of the trust models:

- When the nodes initially enter into the network they are granted certificates for communication
- Each node acts a router to route the packets to the destination. It is assumed that the node that it picks for communication will send the packet to its destination. The trust mechanism is therefore only applied at the end-points
- Threshold cryptography is used to share the secret key in the network. Every node in the network will be a part of the Certification Authority and every node in the CA will follow the rules for granting the certificate
- The threshold value is globally determined, for all nodes and for all network conditions.
- There is no congestion or noise in the network which may distort the communication among the nodes.

- Whenever a node receives a set of request for certificate, they are processed in the order received without giving any priorities.
- The network is assumed to have 30 nodes out of which 15 are good nodes, 5 are spoofing nodes (Bad nodes) 5 are DoS attacking nodes and the reaming 5 are hijacking nodes.
- When a Hijacking node compromises a node , the compromised node is set free after a cycle of communication
- When a node is under DoS attack it will be freed from the attack once it is out of the proximity of the attacker.
- Only DoS attacking nodes send enormous number of requests.
- The same no of nodes remain for the entire length of simulation , no additional nodes enter or leave the network
- All nodes are assumed to have the same memory capacity, battery life and computational capacity
- All the nodes in the network are assumed to be homogenous
- Only DoS attacks, Hijacking, Spoofing and Time delay attacking nodes are present in the network.
- When any node observes malicious activity it is assumed that the node will flood an accusation in the network.
- Only one hop neighbors reply to certificate requests and the flooded messages are sent only to the one hop neighbors.

7.5 Results:

This section gives a detailed description of the performance and the comparison of the two trust models. The two trust models are compared in various aspects as described below:

- percentage of useful communication
- The number of malicious & innocent nodes removed
- The number of malicious nodes remaining
- The number of good nodes remaining
- The network under the influence of DoS attacking nodes
- The network under the influence of hijacking nodes.

In all the graphs below scheme1 refers to the distributed trust model described in Chapter 4 and scheme 2 is the proposed trust model.

1. Comparison of the percentage of useful communication in the two trust models:

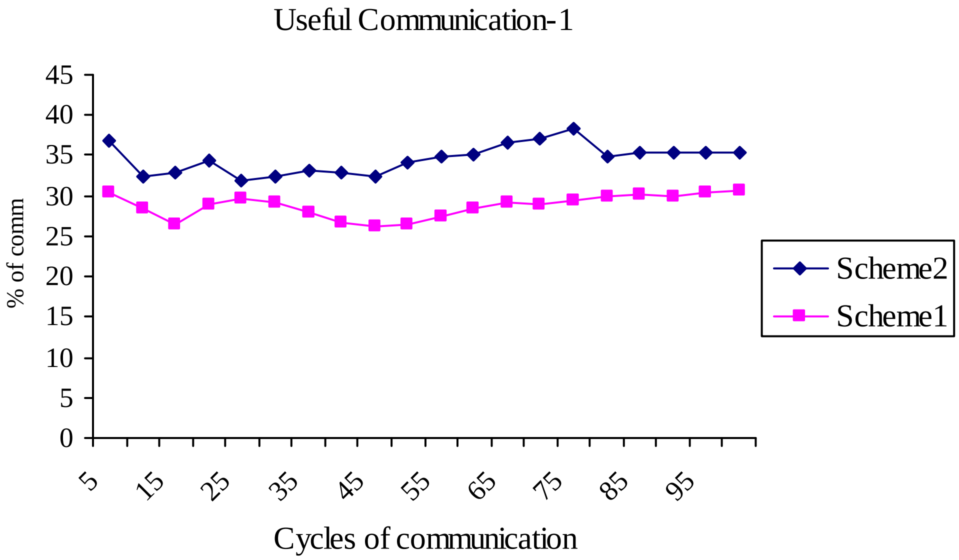


Figure 15: Comparison of the useful communication in the two trust models

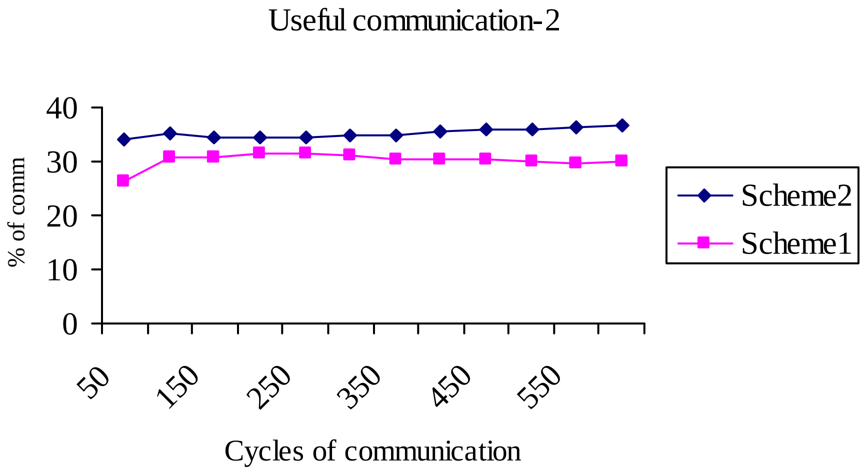


Figure 16: Comparison of the useful communication in the two trust models

Explanation:

The graphs in figures 15, 16 compare the percentage of useful communication in the two models. The experiment was performed for 600 cycles of communication and was repeated for 30 times. Each value in the graph is an average value. From the two graphs above we can come to the conclusion that the proposed Scheme2 achieves a higher percentage of useful communication. Each time it is at least 10% greater than the performance of scheme1. All this achieved by maintaining good trust relationships among the nodes in the network.

2. Malicious and innocent nodes removed in the two models:

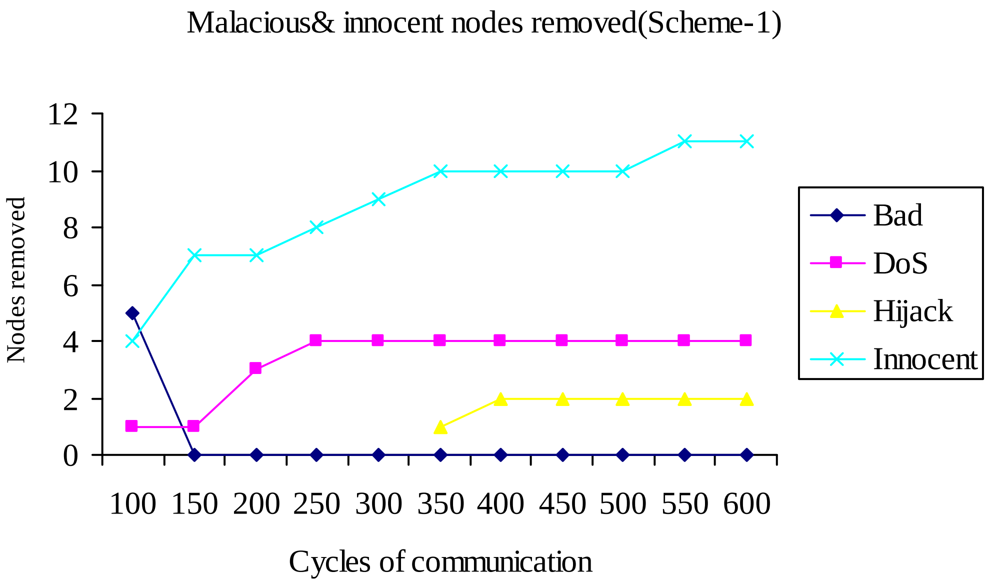


Figure 17: Malicious and innocent nodes removed in scheme-1

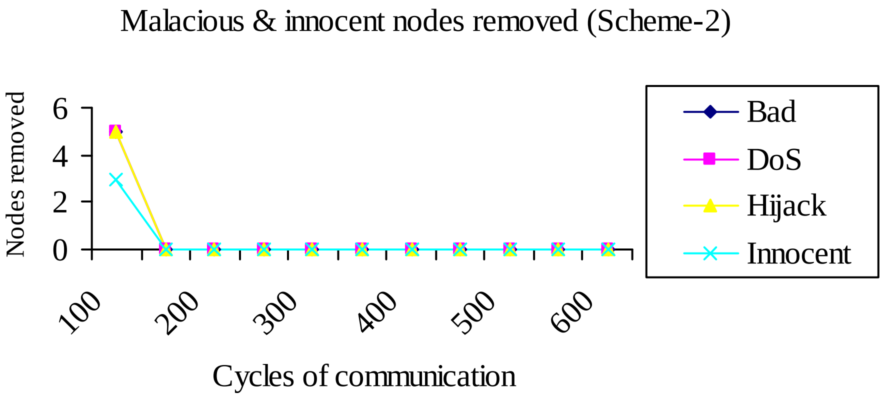


Figure 18: Malicious and innocent nodes removed in scheme-2

Explanation:

The malicious nodes are completely removed in scheme-2 whereas there are some malicious nodes remaining in scheme-1 even after 600 cycles. The number of innocent nodes removed in scheme-2 is very few, around 3, while in scheme-1 about 11 nodes are removed. This is because in scheme1 the nearest node is selected for communication whereas in scheme2 a node is selected based on its availability and quality of service.

3. Malicious nodes remaining in the two models

Explanation: The two graphs below shown in figures 19, 20 illustrate the malicious nodes remaining in the two schemes. The malicious nodes in scheme-2 are all removed by the end of 100 cycles. But in scheme-1 they are not completely removed even by the end of 600 cycles. This shows that scheme-2 is very effective in handling the malicious nodes with the presence of trust among the nodes and the intrusion detection system.

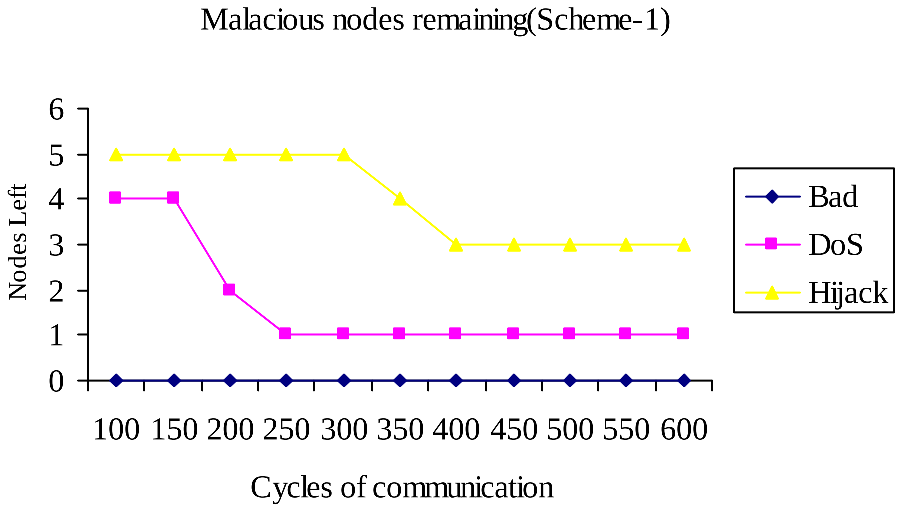


Figure 19: Malicious nodes remaining in Scheme-1

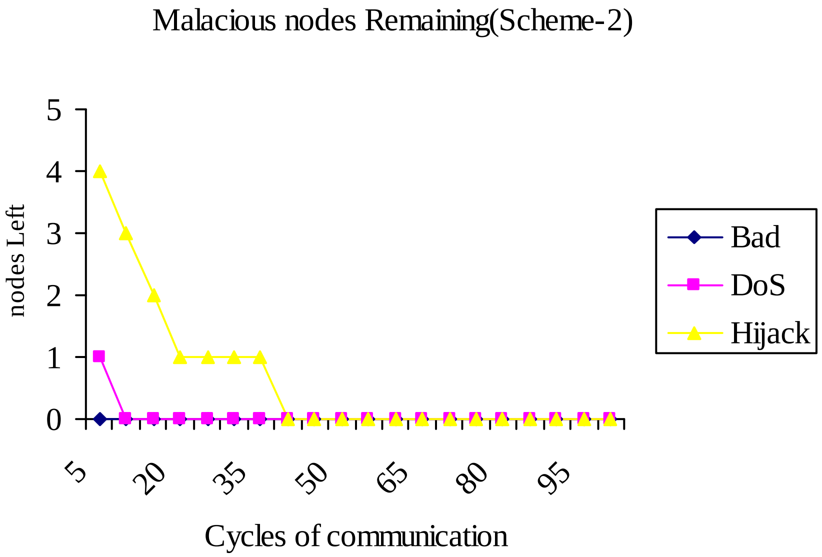


Figure 20: Malicious nodes remaining in Scheme-2

4. Good nodes remaining in the two models:

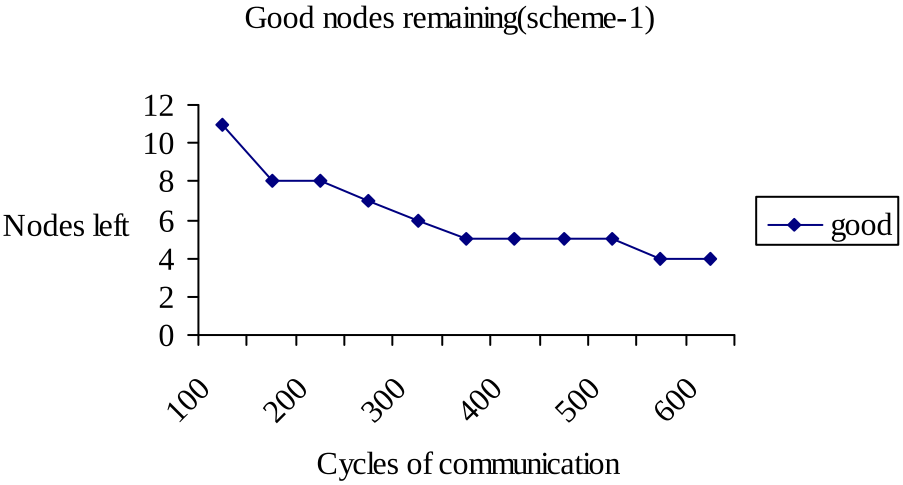


Figure 21: Good nodes remaining in scheme-1

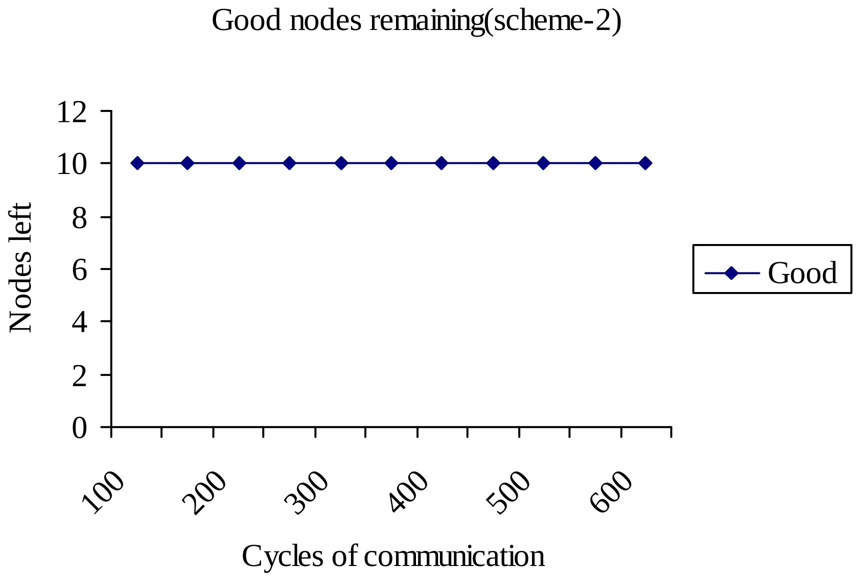


Figure 22: Good nodes remaining in scheme-2

Explanation:

The above graphs in figures 21, 22 show the number of good nodes remaining in the two schemes. The number of nodes in scheme-2 is 10, while there are only 4 nodes remaining in scheme-1 and these will be completely removed due the presence of hijacking nodes. These hijacking nodes are already removed in scheme-2

5. Network under the influence of DoS attacking nodes:

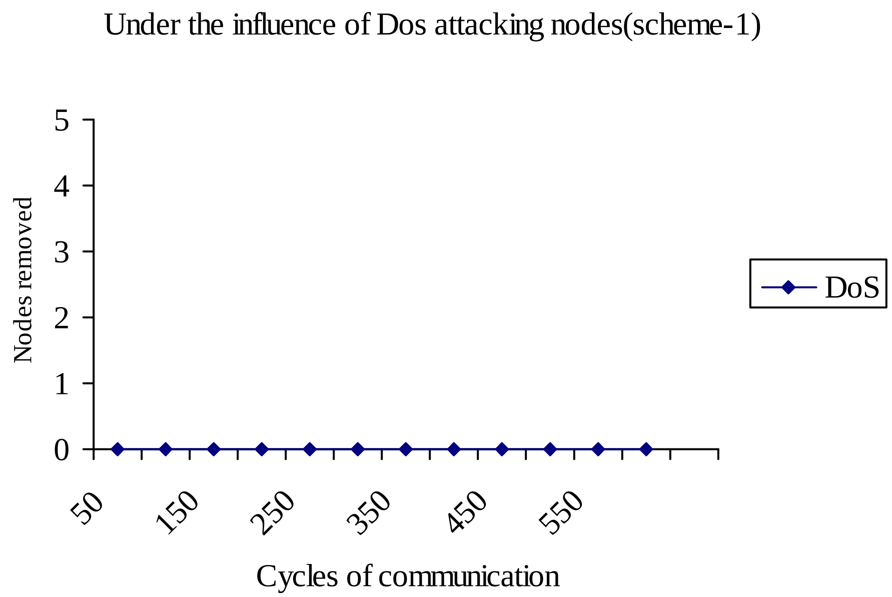


Figure 23: Network under the influence of DoS attacking nodes (scheme-1)

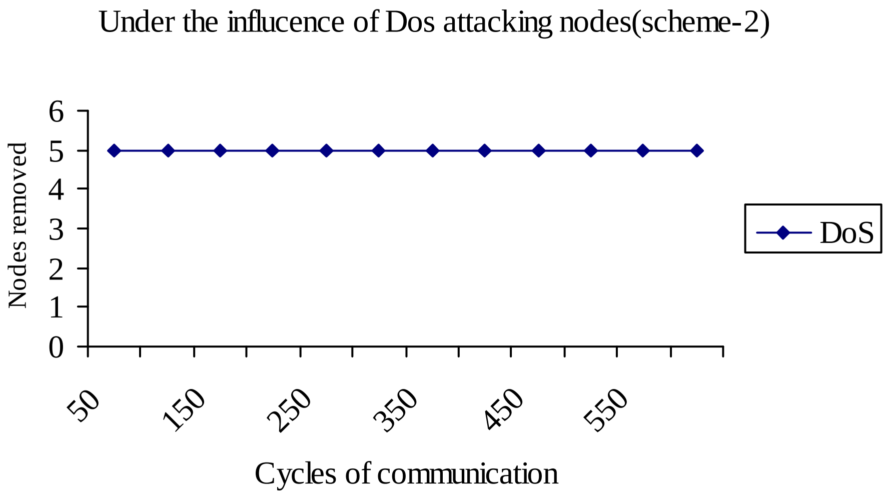


Figure 24: Network under the influence of DoS attacking nodes (scheme-2)

Explanation: The composition of the nodes is changed as there are 25 good nodes and 5 DoS attacking nodes and there are no other nodes in the network. Under such situations the performance is compared. We can clearly see that scheme-1 doesn't remove any DoS attacking nodes since it has no mechanism to remove them. When there is a mixture of malicious nodes they are removed by the hijacking nodes by pure chance of luck. Hence scheme-2 is effective in handling the DoS attacking nodes.

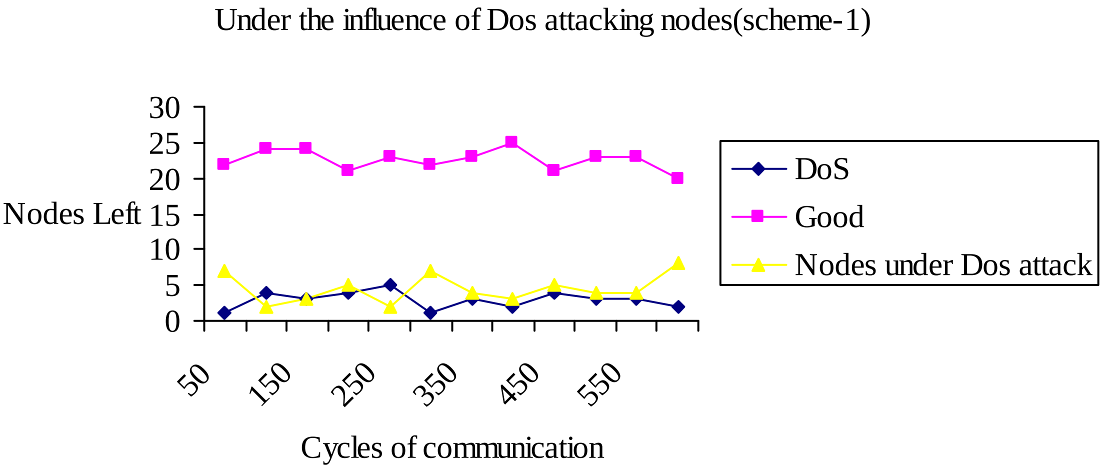


Figure 25: Network under the influence of DoS attacking nodes (scheme-1)

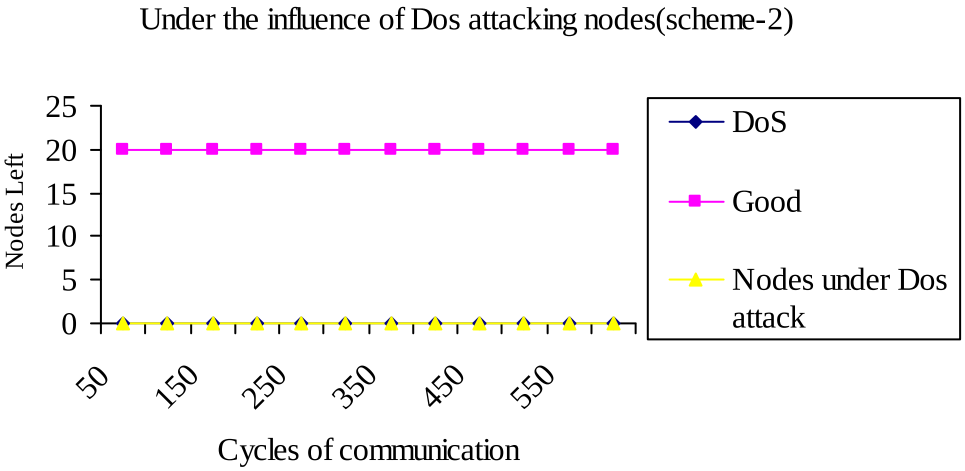


Figure 26: Network under the influence of DoS attacking nodes (scheme-2)

Explanation:

The above graphs show that in scheme-1 the DoS attacking nodes are not removed and there are about 8 nodes under DoS attack. In scheme-2 all the DoS attacking nodes are removed and there is no node under Dos attack due to the presence of intrusion detection system in each node.

6. Network under the influence of hijacking nodes.

Explanation: The composition of the nodes is changed as there are 25 good nodes and 5 hijacking nodes and there are no other nodes in the network. Under such situations the performance is compared. The graphs reveal on close observation that in scheme-1 the hijacking nodes are removed only by a chance of luck whenever a hijacking node hijacks another hijacking node. At 50 cycles it can be observed that there are 20 good nodes remaining, and 4 hijacking nodes remaining. The number of innocent nodes removed should be 5 but it is 6 which show that a hijacking node has attacked another hijacking node.

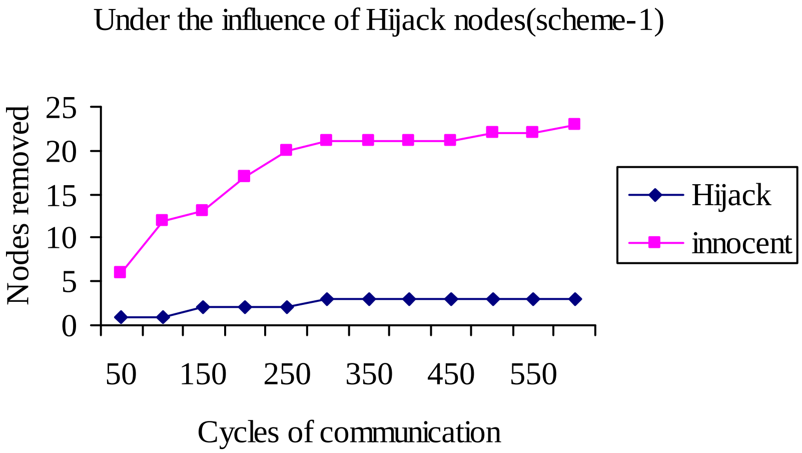


Figure 27: Network under the influence of hijacking nodes (scheme-1)

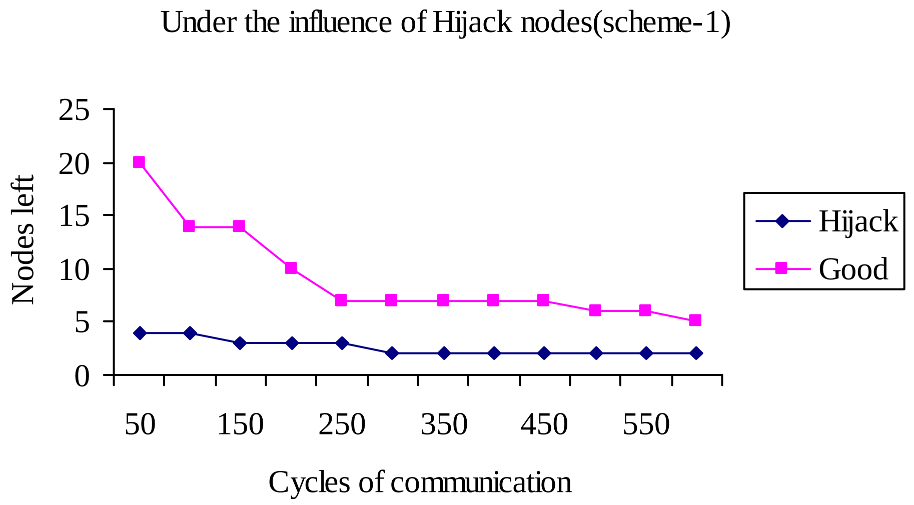


Figure 28: Network under the influence of hijacking nodes (scheme-1)

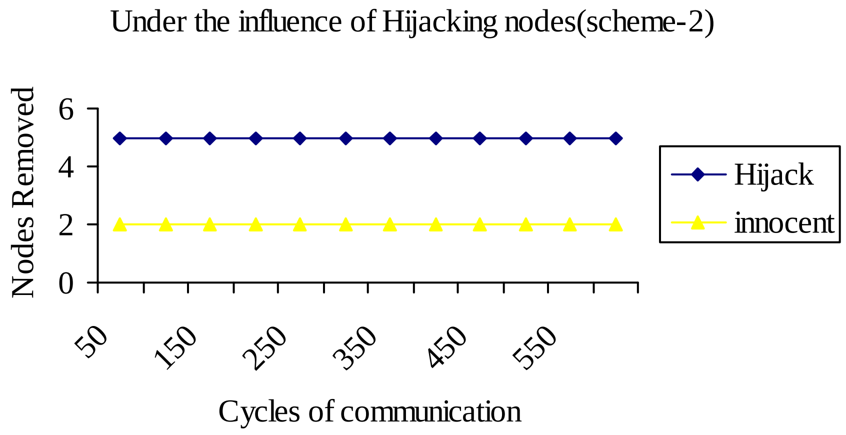


Figure 29: Network under the influence of hijacking nodes (scheme-2)

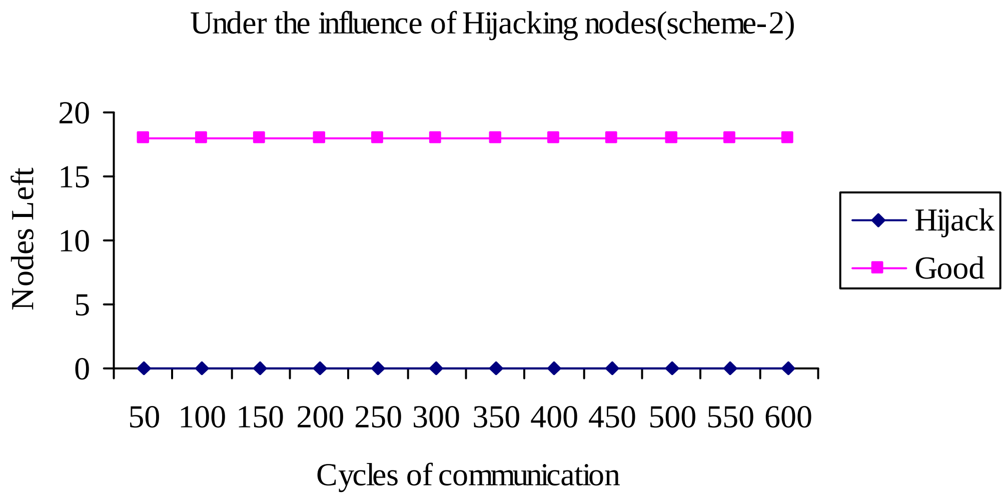


Figure 30: Network under the influence of hijacking nodes (scheme-2)

Explanation: Here it is clearly seen that the number of innocent nodes removed is 2 and the good nodes remaining is 18 and there are no hijacking nodes remaining. Hence from all the above graphs we can conclude that the proposed trust model works efficiently in removing the malicious nodes and protecting the innocent nodes and also in increasing the productivity of the network.

The graphs illustrate that the performance of the proposed trust model is superior to the performance of the trust model described in [19]. The Proposed model proves that by maintaining trust among the nodes the malicious nodes are effectively removed from the network. Since an ad hoc network is infrastructure less the nodes in the network should have some means of relying on the other nodes in the network. Hence trust relationships among the nodes should be established.

CHAPTER VIII

CONCLUSION AND FUTURE WORK

In this thesis we propose a Bayesian network trust model for certificate revocation in ad hoc wireless network. Furthermore, a detailed analysis is conducted of the performance of two different trust models for secure ad hoc wireless networks. This work has proved that a Bayesian network trust model outperforms the distributed trust model described in [19] by establishing good trust relationships among the nodes in the network.

The percentage of useful communication in the network is improved by introducing context specific trust relationships among the nodes. The intrusion detection system of each node combined with its trust relationships with the other nodes effectively removed the malicious nodes in the network. The Bayesian network trust model handled various attacks like hijacking, Dos, spoofing and time delay. The goal of maximizing the removal of malicious nodes and minimizing the removal of innocent nodes from the network is achieved fairly by the proposed trust model.

8.1 Future work:

- The trust model in this work assumes only two parameters for trust (availability, quality of service) .The work can be improved by adding more parameters.
- The model can be enhanced by handling more attacks.
- More realistic results can be obtained by simulating the trust models in ns-2.

- The battery usage of the nodes is not optimized which can be taken into consideration.
- This work doesn’t determine the threshold value K; this work can be combined with the determination of K.
- The nodes in the trust model maintain enormous amount of information; the model can be improved by efficient memory management schemes.
- The Trust model can be modeled under the influence of noise and the performance can be analyzed.

REFERENCES

[1] Bradley Mitchell, Wireless networking, http://compnetworking.about.com/od/wireless,2004.

[2] Ad hoc networking, Kyoto University, Japan. http://www-lab14.kuee.kyoto-u.ac.jp/~aolim/text/adhoc/text/1_1.html

[3] Arthur Conklin.W.M, Gregory B.Whit, Chuck Cothren, Dwayne Williams, Roger L .Davis, “Principles of computer security”, 2004

[4] Nikaein Navid, Mobile Ad Hoc Networking & Computing at Eurecom, http://www.eurecom.fr/~nikaeinn/adhocNetworks/introduction.html,2001.

[5] MANET Group, Wireless Ad hoc Networks. http://w3.antd.nist.gov/wctg/manet.

[6] Humayun Bakht, “A focus on the challenges of mobile ad hoc networks” http://www.computingunplugged.com/issues/issue200408/00001346001.html,2004

[7]Ramanathan.R, Redi.J, “A brief overview of ad hoc networks: challenges and directions”, *IEEE Communications Magazine*, pp. 20-22, Volume: 40, Issue: 5, May 2002

[8] Charles E. Perkins, “Ad hoc networking”, Boston, Addison-Wesley, 2001

[9] Tracy Camp, Jeff Boleng, Vanessa Davies, “A survey of mobility models for ad hoc Network research”, Dept. of Math. And Computer Sciences, Colorado School of Mines, Golden, CO., September 2002.http://toilers.mines.edu/papers/pdf/Models.pdf

[10] Xiaoyan Hong, Mario Gerla, Guangyu Pei, Ching Chuan Chiang, “A group Mobility Model for ad hoc wireless networks”, *Proceedings of the 2nd ACM international workshop on modeling*, pp.53-60, August, 1999

[11] Yongguang Zhang, Wenke Lee, “Intrusion detection in wireless ad-hoc networks”, *Sixth annual international conference on Mobile computing and networking,* pp 275- 283, 2000.

[12] Nick Szabo, Shamir’s secret sharing,
http://szabo.best.vwh.net/secret.html, 1997.

[13] Claude Cr´epeau, Carlton R. Davis, “A certificate revocation scheme for wireless Ad hoc networks”, *1st ACM workshop on Security of ad hoc and sensor networks*, pp 54-61, 2003.

[14] Licia Capra, “Engineering human trust in mobile system collaborations”,
www.cs.ucl.ac.uk/staff/l.capra/publications/fse04.pdf.

[15] Buchegger.S, Le Boudec.J.Y,”The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks”*, In the proceedings of WiOpt `03,* March 2003.

[16] Zhaoyu Liu, Anthony.W.Joy, Thompson.R.A, “A dynamic trust model for mobile Ad hoc networks”, *10th IEEE International Workshop on Future Trends*, pp 80 – 85, 2004

[17]George Theodorakopoulos, John S. Baras,”Trust evaluation in ad hoc networks”, *ACM workshop on Wireless Security,* pp 1-10, 2004

[18] Asad Amir Pirzada, Chris McDonald,” Establishing trust in pure ad hoc networks”, *Proceedings of the 27th conference on Australasian computer science,* pp 47-54, 2004

[19] Haiyun Luo, Zerfos.P, Jiejun Kong, Songwu Lu, Lixia Zhang, “Self-Securing ad hoc wireless networks”, *Seventh International Symposium Proceedings ISCC*, pp 567-574, July, 2002.

[20] Definition of trust, Mtech identity management solutions.
http://mtechit.com/concepts/trust.html.

[21] Delaat, Definition of trust,
www.aaaarch.org/dublin/salowey/definition_of_trust.htm.

[22] Tanenbaum.S.A, “Computer Networks”, Third Edition, 1996.

[23] William Stallings, “Network Security Essentials, Applications and Standards*”*, 2000

[24] Cornelli.F, Damiani.E, “Implementing a Reputation-Aware Gnutella Servant”, *In Proceedings of the International Workshop on Peer-to-Peer Computing,* Pisa, Italy, May 24, 2002.

[25] Jovanovic.M, “Modeling Large-scale Peer-to-Peer Networks and a Case Study of Gnutella”, University of Cincinnati, Master’s thesis, April 2001.

APPENDIX

SOURCE CODE

The following program is the simulation of the Distributed Trust model without Bayesian network (Scheme 1) for ad hoc wireless networks using random waypoint model.

```
/*************************Simulation of scheme1*************************/

 Name: Sudha Chinni
 ID    : 000-408-765
 File  : scheme1.c
/*********************************************************************/
/***********************Declaration of Header files********************************/
#include <stdio.h>
#include<conio.h>
#include <stdlib.h>
#include <time.h>
#include<math.h>
#define nos 31
/**********************Declaration of Global variables***************************/
int lposition[50],m2=1,thresh=5,cycle;
int i1=1, usefulcommu1=0, usefulcommu=0,compromised1=0;
int malacious=0,innocent=0,bad1=0,dos=0,dos1=0,hijack=0,good=0,noise=0,compromised=0;
double commucycle;
int bposition[50];
int certreq[50];
FILE*fp;
/*********************Declaration of functions************************************/
void nodeinit(int);
void newposition(int);
void communication(int);
void revocation();
void intruders(int);
void accusation(int,int);

/* Declaring a global structure for each node in the network*/
        struct node

        {
               int id;//Node's Id
              int accusers[50][50];//Certificate revocation List
               int comments[50];
             //comments about each node whether it is suspected or convicted
              // in comments 1 means convicted, 0 means suspected
               int tcert;//Certificate validity period
               int speed;//Speed at which the node travels
               int time1;// travel time
               int length;//The x coordinate of the nodes position
               int breadth;//The y coordinate if the nodes position
               int destlength;//The x coordinate of the destination
               int destbreadth;//The y coordinate of the destination
               int distcovered;//The total distance covered by the node
               intpausetime;//The total time the node pauses after reaching a
                              destination
               int neighbors[50];//The list of neighbors for the node
```

```
                int request[50];//List of nodes requesting certificates
                int reply;//The number of replies the node gets for its request for
                             Certificate
                int mode;//Defines the type of node (good, bad, Dos , hijacking)
                int replytime;//The time the node takes to reply
                int prevmode;
                //this mainly to used to restore the mode of nodes under dos attack
                //for the next communication cycle.
                int filesize;//size of the file to be transferred
                int dspeed;//download speed;
                int live;// for marking the nodes that are kicked out of network
                int prevnode;
                 //The node with which the communication took place in previous cycle
                int inode;
               //The innocent node accused(in case of hijacking node or compromised node)
                int cycle;//The cycle of communication

        };
//defining a structure of 50 nodes
        struct node n[50];
/******************************Function main()***************************************/
// This function sequentially calls all the functions and at the end prints the
statistics
void main()

{
         int ch=1,j,k,i,k1;
          fp=fopen("summary.txt","w");//opening the summary file in write mode
         for(k1=1;k1<601;k1++)//Repeating the program for a number of cycles
        {fprintf(fp,"\n The cycle of communication is %d",k1);
         cycle=k1;
         i1=1, m2=1;
         for(i=0;i<50;i++)
         certreq[i]=0;//initalizing the certificate request array, this array indicates
                          //nodes requesting certificate renewal


        for(ch=1;ch<6;ch++)
        {

            switch(ch)

            {case 1:    fprintf(fp,"\n\nINITIAL POSITIONS");
                        for(j=1;j<nos;j++)
                        {
                         nodeinit(j);
                                    }
                         break;


            case 2:     fprintf(fp,"\n\nNEW POSITIONS OF THE NODES AFTER TRAVELLING");
                           for(j=1;j<nos;j++)
                         {
                                    if(n[j].live==1)
                          newposition(j);
                                }
                              break;


             case 3 :    fprintf(fp,"\n\nEACH NODE STARTS COMMUNICATING");
                            for(k=1;k<nos;k++)
                                 {
                        communication(k);
                                   }
                               break;

             case 4 :   revocation();
                              break;
             }
        }

fprintf(fp,"\n\nNo of bad nodes removed = %d",malacious);
fprintf(fp,"\n\nNo of innocent nodes removed =%d",innocent);

              for(i=1;i<nos;i++)
          {
       if (n[i].mode==2&&n[i].live==1)
         bad1++;
           if(n[i].mode==4&&n[i].live==1)
              dos++;
           if(n[i].mode==3&&n[i].live==1)
              dos1++;
           if(n[i].mode==5&&n[i].live==1)
              hijack++;
       if(n[i].mode==1&&n[i].live==1)
              good++;
           if(n[i].mode==6&&n[i].live==1)
           compromised++;

         }
//printing the type of nodes remaining after each cycle
fprintf(fp,"\n\nNo of bad nodes remaining = %d",bad1);
fprintf(fp,"\n\nNo of Dos attacking nodes remianing = %d",dos1);
fprintf(fp,"\n\nNo of Hijacking attacking nodes remianing = %d",hijack);
fprintf(fp,"\n\nNo of nodes under Dos Attack = %d",dos);
fprintf(fp,"\n\nNo of good nodes remaining = %d",good);
fprintf(fp,"\n\nNo of compromised nodes remaining = %d",compromised);


bad1=0;
dos=0;
dos1=0;
hijack=0;
good=0;
compromised=0;
compromised1=0;

}// end of major for loop

  }//end of main

/*******************************Function nodeinit()************************************/
//This function initializes the various fields in each node
void nodeinit(int i)
     {   int j,k;
        srand(50+i+cycle);//seeding the random generator
               fprintf(fp,"\n\nNode %d",i);

         n[i].id=i;//assigning Id to the node
               if (cycle==1)
               n[i].live=1;
               //assiging the mode to the node

//Let mode 1 : good, 2 : Bad ,3: Dos attacking nodes ,4:node under Dos attack,5:
Hijacking nodes  6:compromised node
               if(cycle==1)
                 {
                     if(i<=15)
                        n[i].mode=1;
                   if(i>15&&i<=20)
                        n[i].mode=2;
                  if(i>20&&i<=25)
                        n[i].mode=3;
                   if(i>25&&i<=30)
                         n[i].mode=5;
                  //initializing the comments array
                     for(k=0;k<50;k++)
                     {n[i].comments[k]=-1;
                     }
```

```
                      n[i].prevmode=n[i].mode;
            //randomly initializing the reply time
                     if (n[i].mode==1)
                       n[i].replytime=rand()%10;
                    if(n[i].mode==2||n[i].mode==5)
                       n[i].replytime=rand()%20;
                    if(n[i].mode==3||n[i].mode==6)
                       n[i].replytime=rand()%5;  }
   // Determine the initial position of the node

          if(cycle==1)
          {
                  n[k].cycle=1;
          lposition[i]=rand()%600;
       bposition[i]=rand()%600;
          n[i].length=lposition[i];
          n[i].breadth=bposition[i];

          }


        fprintf(fp,"\n\nPosition :%dX%d",n[i].length,n[i].breadth);

           if (cycle==1)
        n[i].tcert=20;//initalizing the certificate time

               if(n[i].live==1)
               {
               fprintf(fp,"\n\nTcert=%ld seconds ",n[i].tcert);
        //initializing speed and time
           n[i].speed=rand()%11;
           n[i].time1=rand()%21;

        fprintf(fp,"\n\nSpeed =%dm/sec,Traveltime=%ldseconds",n[i].speed,n[i].time1);
        // initializing the destination
       if(((n[i].length==n[i].destlength)&& (n[i].breadth==n[i].destbreadth))||cycle==1)
               { n[i].destlength=rand()%600;
             n[i].destbreadth=rand()%600;
               }
               //initializing the file size to be transferred and the download speed
               n[i].filesize=rand()%500;
               n[i].dspeed=rand()%30;

           fprintf(fp,"\n\nDestination:%dX%d",n[i].destlength,n[i].destbreadth);
           fprintf(fp,"\n");
               }

               fprintf(fp,"\n\nMode =%d",n[i].mode);
}

/**********************************Function newposition()********************/
//The main aim of this function is to find the new positions of the nodes after they
reach a destination
void newposition(int i)

{

       double theta,x,y,totaldistance;
       int distancecovered;

    fprintf(fp,"\n\nNode%d",i);
   //calculating the distance between the nodes initial position and destination
       y=(n[i].destlength-n[i].length)*(n[i].destlength-n[i].length) + (n[i].destbreadth-
n[i].breadth)*(n[i].destbreadth-n[i].breadth);
       totaldistance=sqrt(y);
       fprintf(fp,"\n\nDestination:%dX%d",n[i].destlength,n[i].destbreadth);
       fprintf(fp,"\n\nTotal Distance to travel =%g",totaldistance);

    // calculating the distance covered
       distancecovered=n[i].speed *n[i].time1;
       fprintf(fp,"\n\nDistance Covered:%d",distancecovered);
       fprintf(fp,"\n\nOld position:%dX%d",n[i].length,n[i].breadth);
       n[i].distcovered=distancecovered;
       if(distancecovered >= totaldistance) //rounding off the distance
       {
              n[i].length=n[i].destlength;
              n[i].breadth=n[i].destbreadth;

       }
       // finding the nodes new position
       else
       {
              //1st quadrant
              if((n[i].destlength >=n[i].length) && (n[i].destbreadth >=n[i].breadth))
              {
              if((n[i].destlength-n[i].length)==0)
                     theta=3.142/2.0;
              else
              {
       x=(double)(n[i].destbreadth-n[i].breadth)/(double)(n[i].destlength-n[i].length);
             theta=atan(x);
              }
                     n[i].length=n[i].length + (distancecovered * cos(theta));
                     n[i].breadth=n[i].breadth + (distancecovered*sin(theta));
              }
              //2nd quadrant
       else if((n[i].destlength <= n[i].length) && (n[i].destbreadth >=n[i].breadth))
              {
                     if((n[i].destlength-n[i].length)==0)
                            theta=3.142/2.0;
                      else
{x=(double)(n[i].destbreadth-n[i].breadth)/(double)((-1)*(n[i].destlength-
n[i].length));
           theta=atan(x);
                     }

                     n[i].length=n[i].length - (distancecovered * cos(theta));
                     n[i].breadth=n[i].breadth + (distancecovered*sin(theta));
        }


       //3rd quadrant
       else if((n[i].destlength <= n[i].length) && (n[i].destbreadth <= n[i].breadth))
              {
                     if((n[i].destlength-n[i].length)==0)
                            theta=3.142/2.0;

       else
       {x=(double)((-1)*(n[i].destbreadth-n[i].breadth))/(double)((-1)*(n[i].destlength-
n[i].length));
           theta=atan(x);
                     }
                     n[i].length=n[i].length - (distancecovered * cos(theta));
                     n[i].breadth=n[i].breadth - (distancecovered*sin(theta));
        }

//4th quadrant
       else if((n[i].destlength >= n[i].length) && (n[i].destbreadth <=n[i].breadth))
              {
                     if((n[i].destlength-n[i].length)==0)
                            theta=3.142/2.0;
       else
{x=(double)((-1)*(n[i].destbreadth-n[i].breadth))/(double)(n[i].destlength-n[i].length);
           theta=atan(x);
                     }
                     n[i].length=n[i].length + (distancecovered * cos(theta));
                     n[i].breadth=n[i].breadth - (distancecovered*sin(theta));
              }

       }
        lposition[i]=n[i].length;
       bposition[i]=n[i].breadth;

       fprintf(fp,"\n\nNew position:%dX%d",n[i].length,n[i].breadth);
       if (n[i].time1>n[i].tcert)
              n[i].tcert=0;
       else
       n[i].tcert=n[i].tcert-n[i].time1;
       //finding out the certificate validity period
       fprintf(fp,"\n\nThe time remaining is %d",n[i].tcert);
       //initalizing the pause time for each node
       n[i].pausetime=rand()%20;
       while(n[i].pausetime==0)
              n[i].pausetime=rand()%20;
       fprintf(fp,"\n\nPause time :%d",n[i].pausetime);
       fprintf(fp,"\n\nReply time :%d",n[i].replytime);
}
/**********************************Function communication()***********************/
// This function describes the nodes communication in the network
void communication(int k)
{
       //Declaration of local variables
       int
l=1,reflength,refbreadth,flag,refnode,j,min[50],min1,p=1,m1,m,destnode,i,target,flag1=0;
       int tfile1,tfile,flag2=0,pausetime1,pausetime,inode;
       double totaldistance=0,y,commucycle;

       //intially a check is made if there is a valid certificate or not
       if(n[k].live==1)
       fprintf(fp,"\n\nNODE %d",k);
           //initializing the array
                 for(j=1;j<50;j++)
                       n[k].neighbors[j]=0;


   //since there is a valid certificate it has to determine the node with which it wants
to communicate
   //each node finds all its neighbors and picks nearest node to communicate to save
battery power.
         refnode=k;
         reflength=lposition[k]+100;
         refbreadth=bposition[k]+100;


              for(j=1;j<50;j++)
                     min[j]=0;

   for(j=1;j<nos;j++)

   {
           if (refnode==j&&j!=(nos-1))
        j=j+1;
           if(refnode==j&&j==(nos-1))
                 break;

    if ((lposition[j]<=reflength)&&(bposition[j]<=refbreadth))
         {
                flag=1;
                if(n[j].live==1)
                {
                n[k].neighbors[l]=j;
                l++;
                }
         }

   }
   if (l==0)
   {fprintf(fp,"\n\nThe node%d has no neighbors to communicate",k);
       flag1=1;
   }

   //if a node is under Dos attack, it will be under Dos attack as long as the it is in
   //the vicinity of the Dos attacking node
   if(n[k].mode==4)
   { for(i=1;i<nos;i++)
       {
          if(n[k].neighbors[i]==n[k].prevnode)
            {  flag2=1;
                  n[k].mode=4;

            }
        }
   if( flag2==0)
   n[k].mode=n[k].prevmode;

   }

//A node communicates only if it has a neighbor, if it has a valid certificate and if it
is not under
// a Dos attack
   if(n[k].tcert>0&&n[k].mode!=4&&flag1!=1)
   {
          //now the node looks around its neighborhood it see any oberservable malacious
activity

  intruders(k);
        //the node calculates its distance to each of its neighbor
          for(m=1;m<nos;m++)

          {
                 m1=n[k].neighbors[m];
                 if(m1!=0)
                 {
         y=((n[m1].length-n[k].length)*(n[m1].length-n[k].length) + (n[m1].breadth-
n[k].breadth)*(n[m1].breadth-n[k].breadth));
          totaldistance=sqrt(y);
          //fprintf(fp,"\n\nThe distance to node %d is %f",m1,totaldistance);
              min[p]=totaldistance;
                  p++;

                 }

     }


  //The node picks up its nearest node to communicate
          for(p=1;p<nos;p++)
          {
                destnode=n[k].neighbors[p];
                if(destnode!=0)
                {
                if(n[destnode].tcert==0)
                min[p]=-1;
                if(n[destnode].live==0)
                       min[p]=-1;
          }
          }


    min1=min[1];
    for(p=1;p<nos;p++)


        {
      if(min1>0)
         {
       if(min[p]<=min1&& min[p]>0)
         { min1=min[p];
           destnode=n[k].neighbors[p];

         }

        }
```

```
          else
          {
                 if(min[p]>=min1&& min[p]>0)
                 { min1=min[p];
                destnode=n[k].neighbors[p];

               }
          }

         }
  if(destnode==0)
  fprintf(fp,"\n\nThe node %d has no node to communicate",k);

  n[k].prevnode=destnode;

        //a node can communicate only if has another node to communicate with
   if(destnode!=0&&n[destnode].mode!=4)
   {

   fprintf(fp,"\n\n The destnode is %d",destnode);
   n[destnode].prevnode=k;
   //if the node is a hijacking node , it picks up an innocent node to accuse, it then
   // floods an accusation in the network, it compromises the target node with which it
communicates
   // the compromised node also accuses the innocent node, the hijacking node will do the
   //same for threshold number of cycles to remove the innocent node.
   if(n[k].mode==5)
   {
           n[k].cycle=n[k].cycle+1;
           if(n[k].cycle==thresh+1)
                  n[k].cycle=1;

           if(n[k].cycle==1)
           {

         do
        {
        inode=rand()%nos;
        }while(inode==0&&inode==k);

        while(n[inode].live==0)

        {inode=rand()%nos;
        }

        n[k].inode=inode;
           }


    inode=n[k].inode;
        n[destnode].mode=6;
        n[destnode].inode=n[k].inode;
        //accusing the innocent node
        n[k].comments[inode]=1;
        accusation(k,inode);
        fprintf(fp,"\n\nInnocent node  is %d",inode);

   }
   //the node in compromised mode
   if(n[k].mode==6)
   {inode=n[k].inode;
    n[k].comments[inode]=1;
    accusation(k,inode);
        //after accusing it comes out of the compromised mode and returns back to its
previous mode
    n[k].mode=n[k].prevmode;
        n[k].inode=0;
   }
//calculating pause time
 // the maximum time for which the communication can take place is the time of Tcert of
the
```

```
 //source node which is again based on its pause time and the reply time of the
destination

   pausetime1=n[k].pausetime;
   pausetime=n[destnode].pausetime;

   if(n[k].mode==3)
          n[destnode].mode=4;
   if(n[k].prevmode==3)
          n[destnode].mode=4;//since a dos attacking node may be in mode 5 or 4
   if(n[k].pausetime>n[k].tcert)
   pausetime1=n[k].tcert;

    if(n[destnode].pausetime>n[destnode].tcert)
        pausetime=n[destnode].tcert;

   n[k].tcert=n[k].tcert-pausetime1;
   n[destnode].tcert=n[destnode].tcert-pausetime;

       if(n[destnode].replytime >= n[destnode].pausetime)
       pausetime=0;
       else
               pausetime=n[destnode].pausetime-n[destnode].replytime;


  // if the certificate of the source node expires it sends a request for it


 if(n[k].tcert==0)
 {
       certreq[m2]=k;
                  m2++;
   }

   //calculating the useful communication, which implies that the file that could be
transferred by
   // the node out the size that was expected to be transferred , since it is delimited
by Tcert , reply time
    //and pause time of the two nodes in communication
 tfile=n[k].dspeed*pausetime1;
   if(tfile>n[k].filesize)
          tfile=n[k].filesize;
   usefulcommu=usefulcommu+tfile;
   if(pausetime1<=pausetime)
   tfile1=n[destnode].dspeed*pausetime1;
   else
   tfile1=n[destnode].dspeed*pausetime;
   if(tfile1>tfile)
          tfile1=tfile;
   usefulcommu1= usefulcommu1+tfile1;
   if(pausetime1==0)
          fprintf(fp,"\n\n Pause time of node %d is 0",k);
   if(pausetime==0)
          fprintf(fp,"\n\n Pause time of destnode %d is 0",destnode);


   fprintf(fp,"\n\n The tcert remaining is %d",n[k].tcert);
   fprintf(fp,"\n\n The tcert of dest node is %d",n[destnode].tcert);
   }
   if(n[destnode].mode==4)
   {  n[destnode].tcert=0;//since it will not reply and pause untill its tcert is over
  //calculating the % of useful communication
   tfile=n[k].dspeed*n[k].pausetime;
   if(n[k].tcert<n[k].pausetime)
          n[k].pausetime=n[k].tcert;
   if(tfile>n[k].filesize)
          tfile=n[k].filesize;
   usefulcommu=usefulcommu+tfile;
   }
   commucycle=((double)(usefulcommu1)/(double)(usefulcommu))*100;
   fprintf(fp,"\n\nThe '%' of useful communication is %f",commucycle);
   }



   else
   {
          int flag3=0;
          if(n[k].mode==4&&n[k].live==1)
          {fprintf(fp,"\n\nThe node %d is under DOS attack",k);
          if(n[k].tcert<n[k].pausetime)
                  n[k].pausetime=n[k].tcert;
          tfile=n[k].dspeed*n[k].pausetime;
       if(tfile>n[k].filesize)
          tfile=n[k].filesize;
          usefulcommu=usefulcommu+tfile;
          flag3=1;
       commucycle=((double)(usefulcommu1)/(double)(usefulcommu))*100;
       fprintf(fp,"\n\nThe '%' of useful communication is %f",commucycle);
      }
          if(n[k].tcert==0&&n[k].live==1&&flag3==0)
          {fprintf(fp,"\n\nNo valid certificate to communicate");
          certreq[m2]=k;//requesting for certificate
           m2++;
               commucycle=((double)(usefulcommu1)/(double)(usefulcommu))*100;
       fprintf(fp,"\n\nThe '%' of useful communication is %f",commucycle);
          }

   flag1=0;
   flag2=0;
   flag3=0;
   }

}/*********************************Function intruders()********************************/
//Each node uses this function to scan the malicious activity in its neighborhood and
// pass accusations
void intruders(int k)
{
       int i=k,target,j,k1,k2,sum;

        for(j=1;j<nos;j++)
        {
                target=n[i].neighbors[j];
           if(target!=0&&n[target].mode==2)
          {
                  n[i].comments[target]=1;
     // when a node is found to be convicted, all the accusations sent by the  convicted
                 //node are cleared in the list of accusers.

                  for(k1=1;k1<nos;k1++)
                  {n[i].accusers[k1][target]=0;
                  }

                  for(k1=1;k1<50;k1++)
                  {sum=0;
                  for(k2=1;k2<nos;k2++)
                  {if(n[i].accusers[k1][k2]!=0)
                   sum=sum+n[i].accusers[k1][k2] ;
                  }
       // if the number of accusers are >= threshold then the node is marked as convicted
         //otherwise it is marked as a suspect
                  if(sum>=thresh)

                              n[i].comments[k1]=1;
                      else
                              n[i].comments[k1]=0;
                  }
```

```
            //the accusation is flooded in the network
                  accusation(i,target);

       }
     }

}
/*****************************Function accusation()************************************/
//In this function when all nodes receive the flooded message they make a note of it in
the certificate revocation list

   void accusation(int accuser,int badnode)
        { int i,row,col,sum=0,k1;

         for(i=1;i<nos;i++)
         {

    if(i!=accuser&&i!=badnode)
         {
                row=badnode;
                col=accuser;
                n[i].accusers[row][col]=1;
                if(n[i].comments[badnode]!=1)
                {
          for(k1=1;k1<50;k1++)
                 {if(n[i].accusers[row][k1]!=0)
                    sum=sum+n[i].accusers[row][k1];
                 }
             if(sum>=thresh)
              n[i].comments[badnode]=1;
             else
                n[i].comments[badnode]=0;
                }

         }
         sum=0;
         }
        }

/**********************************Function revocation()******************************/
//Now each node will answer to the requests that it receives(request for certificate
renewal)
// each node will decide whether to reply or not based on whether it is convicted or not
// if the node is convicted the certificate is not renewed

void revocation()
{  //now the request for the certificates is flooded in the network, certificate is
granted
   // one by one to each node.

int i,j,n2,n3,n11,n22,member=0;

for(i=1;i<nos;i++)

{
  if (certreq[i]>0)

  {
          n2=certreq[i];

   for(j=1;j<nos;j++)
   {
    n3=n[n2].neighbors[j];
        if (n[n3].live==0||n[n3].mode==4)//since a node under dos attack cannot reply
        {
         n[n2].neighbors[j]=0;
         n3=0;
        }
        if(n3>0)
        {   member++;
               n[n3].request[i]=n2;
        }
   }

   fprintf(fp,"\n\nThe Node %d has %d neighbours",n2,member);
   if(member<thresh)
   {
fprintf(fp,"\n\n The certificate for %d cannot be revoked due to insufficient
neighbours",n2);
   certreq[i]=0;
   }
   member=0;

  }
}
//now each nodes decide whether it has to revoke the certificate or not

for(i=1;i<nos;i++)

{
        n[i].reply=0;
}

    for(i=1;i<nos;i++)
    {
        for(j=1;j<50;j++)
        { if(n[i].request[j]>0)
        {
            n11= n[i].request[j];
        if (n[i].comments[n11]!=1)

            n[n11].reply=n[n11].reply+1;

        }
    }
}


// now each node forms a certificate based on the no of replies it has got

        for(i=1;i<50;i++)

        {if( certreq[i]>0)

        { n22=certreq[i];

      if(n[n22].reply>=thresh)
          { n[n22].tcert=20;
           fprintf(fp,"\n\nCERTIFICATES ARE REVOKED");
           fprintf(fp,"\n\nThe Certificate of Node %d is revoked",n22);
          }
          else
          {
                 fprintf(fp,"\n\nThe node %d is kicked out of the network ",n22);
              n[n22].live=0;
                 if (n[n22].prevmode==2)
                        malacious++;//no of malicious nodes removed
                 else

                        innocent++;//no of innocent nodes removed due to malicious blames
on them.

          }
        }

        }
         }



   /****************************End of Simulation (scheme1) ***************************/
```

/***/
The following program is the simulation of the “Bayesian network Trust model
 (Scheme 2) ”for ad hoc wireless networks using random waypoint model.
  File: Scheme2.c
/*************************Simulation of scheme 2**************************/

```c
/*************************Declaration of Header files****************************/
#include <stdio.h>
#include<conio.h>
#include <stdlib.h>
#include <time.h>
#include<math.h>
#define nos 31
#define st 0.5
#define thresh 5
/*******************************Declaration of global variables*********/
int requests=10,timedelay=10,fflag=0,certreq[50];
float usefulcommu;
int cycle,i1,inode,destnode,tfile,tfile1,commureq=0,commudone=0,m2;
int dos=0,hijack=0,bad=0,innocent=0,unsatis=0,good=0;
FILE*fp;
/******************************Declaration of Functions***********/
void nodeinit(int);
void newposition(int);
void neighbourhood();
void communication(int);
void intruders(int);
void accusation(int,int);
void targetnode(int,int);
int recomendation(int,int);
void check(int,int,int);
void revocation();
/* Declaring a global structure for each node in the network*/
    struct node

        {
                int id;//Node's Id
                int tcert;//Certificate validity period
                int speed;//Speed at which the node travels
                int time1;// travel time
                int length;//The x coordinate of the nodes position
                int breadth;//The y coordinate if the nodes position
                int destlength;//The x coordinate of the destination
                int destbreadth;//The y coordinate of the destination
                int distcovered;//The total distance covered by the node
                int pausetime;
                //The total time the node pauses after reaching a destination
                int neighbors[50];//The list of neighbors for the node
                double neighbors1[50][15];//Trust table maintained by each node.
                int request[50];//List of nodes requesting certificates
                int reply;
                //The number of replies the node gets for its request for certificate
                int mode;//Defines the type of node(good,bad, Dos , hijacking)
                int replytime;//this is for handshaking during communication
                int prevmode;
                   //this mainly to used to restore the mode of nodes under dos attack
                  //for the next communication cycle.
                int filesize;//size of the file to be transferred
                int qos;//download speed;
                int live;// for marking the nodes that are kicked out of network
                int prevnode;//for marking the DOS attacking node
                float wa;//weight for availability
                float wq;//weight for quality of service
                int prevaccused;//The node that was accused in previous cycle
                float recoletters[50][4];//The list of recommendation letters
                int posrecos[50];//The list of nodes giving positive recommendations
                int negrecos[50];//The list of nodes giving negative recommendations
                long int timestamp[50];//The time stamp of experience
                int availability;// availability of the node
                long int rtimestamp[50];
                //The timestamp the recommendation letter was received
                int nrequest;//no of requests placed by the source node
                int splflag;//a flag to indicate the threshold of requests
                int inode;// for node in mode 5 & 6
                int cycle;//for node in mode 5
                int comments[50][50];
                 //reason for which the previous accused node was removed
                //1:mode2,2:dos,3:spoofing,4:info from trusted nodes

        };

        //defining a structure of 50 nodes
        struct node n[50];
/****************************************Function main()***************************/
// This function sequentially calls all the functions

 void main()

    {
        int ch=1,j,k1,k;
         fp=fopen("summary1.txt","w");//opening the summary file in write mode

        for(k1=1;k1<601;k1++)//Repeating the program for a number of cycles
        {
         fprintf(fp,"\n The cycle of communication is %d",k1);
         cycle=k1;
         i1=1,m2=1;

        for(ch=1;ch<6;ch++)
        {
          switch(ch)

        {case 1:    fprintf(fp,"\n\nINITIAL POSITIONS");
                    for(j=1;j<nos;j++)
                    {
                    nodeinit(j);
                              }
                     break;


        case 2:      fprintf(fp,"\n\nNEW POSITIONS OF THE NODES AFTER TRAVELLING");
                    for(j=1;j<nos;j++)
                       {
                       if(n[j].live==1)
                        newposition(j);
                       }
                       break;
        case 3 :      if(cycle==1)
                       neighbourhood();
                        break;

        case 4 :      fprintf(fp,"\n\nEACH NODE STARTS COMMUNICATING");
                        for(k=1;k<nos;k++)
                         {if(n[k].live==1)
                         communication(k);
                          }
                         break;
        case 5 :       revocation();
                        break;
                 }

           }

     }// end of major for loop

}//end of main



/******************************Function nodeinit()***********************************/
//This function initializes the various fields in each node
void nodeinit(int i)
         {int j;
           srand(50+i+cycle);//seeding the random generator
          double sa1,sa22=0,sum;
           int sa=0,sa2=0;

               for(j=0;j<50;j++)
                      certreq[j]=0;

                if (cycle==1)
                n[i].live=1;
                if(n[i].live==1)
                {

                fprintf(fp,"\n\nNode %d",i);
                n[i].id=i;//assigning Id to the node



                if(cycle==1)
                { /* modes: 1- good node,2-bad node(spoofing),3-dos attacking node,
               (4-node under dos attack),5-nodes which will hijack other nodes,
                   6-compromised node*/
                      if(i<=15)
                        n[i].mode=1;
                 if(i>15&&i<=20)
                        n[i].mode=2;
                 if(i>20&&i<=25)
                        n[i].mode=3;
                 if(i>25&&i<=30)
                        n[i].mode=5;
                         n[i].prevmode=n[i].mode;

                         //determine wa&wq

               while(sa==0)
               sa=rand()%100;
               sa1=(float)sa/100;

                         sum=sa1+sa22;//wa+wq should be equal to 1

            while(sa2==0||sum!=1.0000)
            {sa2=rand()%100;
             sa22=(float)sa2/100;
             sum=sa1+sa22;
             }
               n[i].wa=sa1;
                      n[i].wq=sa22;
                      fprintf(fp,"\n\n weight for availability is %f",sa1);
                      fprintf(fp,"\n\n weight for qos is %f",sa22);


                               }


          //randomly initializing the reply time
                       if (n[i].mode==1)
                       n[i].replytime=rand()%10;
                    if(n[i].mode==2||n[i].mode==5)
                       n[i].replytime=rand()%20;
                    if(n[i].mode==3||n[i].mode==6)
                       n[i].replytime=rand()%5;

          // Determining the initial position of the node

              if(cycle==1)
               {
               n[i].length=rand()%600;
               n[i].breadth=rand()%600;
                  }


                 fprintf(fp,"\n\nPosition :%dX%d",n[i].length,n[i].breadth);

            if (cycle==1)
         n[i].tcert=20;//initalizing the certificate time

               if(n[i].live==1)
               {
               fprintf(fp,"\n\nTcert=%ld seconds ",n[i].tcert);
         //initializing speed and time
           n[i].speed=rand()%11;
           n[i].time1=rand()%21;

        fprintf(fp,"\n\nSpeed =%dm/sec,Traveltime=%ldseconds",n[i].speed,n[i].time1);
        // initializing the destination
       if(((n[i].length==n[i].destlength)&& (n[i].breadth==n[i].destbreadth))||cycle==1)
               { n[i].destlength=rand()%600;
             n[i].destbreadth=rand()%600;
               }
               //initializing the file size to be transferred and the download speed
               n[i].filesize=rand()%500;
               n[i].qos=rand()%30;

           fprintf(fp,"\n\nDestination:%dX%d",n[i].destlength,n[i].destbreadth);
           }
        fprintf(fp,"\n\nMode =%d",n[i].mode);
    }
                }
/**********************************Function newposition()*******************/
//The main aim of this function is to find the new positions of the nodes after they
reach a destination

void newposition(int i)

{

       double theta,x,y,totaldistance;
       int distancecovered;

        fprintf(fp,"\n\nNode%d",i);
     //calculating the distance between the nodes initial position and destination
  y=(n[i].destlength-n[i].length)*(n[i].destlength-n[i].length) + (n[i].destbreadth-
n[i].breadth)*(n[i].destbreadth-n[i].breadth);
       totaldistance=sqrt(y);
       fprintf(fp,"\n\nDestination:%dX%d",n[i].destlength,n[i].destbreadth);
       fprintf(fp,"\n\nTotal Distance to travel =%g",totaldistance);
    // calculating the distance covered
       distancecovered=n[i].speed *n[i].time1;
       fprintf(fp,"\n\nDistance Covered:%d",distancecovered);
       fprintf(fp,"\n\nOld position:%dX%d",n[i].length,n[i].breadth);
       n[i].distcovered=distancecovered;
       if(distancecovered >= totaldistance) //rounding off the distance
       {
               n[i].length=n[i].destlength;
               n[i].breadth=n[i].destbreadth;

       }
```

```
        // finding the nodes new position
        else
        {
                //1st quadrant
                if((n[i].destlength >=n[i].length) && (n[i].destbreadth >=n[i].breadth))
                {
                        if((n[i].destlength-n[i].length)==0)
                                theta=3.142/2.0;
        else
        {
x=(double)(n[i].destbreadth-n[i].breadth)/(double)(n[i].destlength- n[i].length);
                 theta=atan(x);
                        }
                        n[i].length=n[i].length + (distancecovered * cos(theta));
                        n[i].breadth=n[i].breadth + (distancecovered*sin(theta));
                }
                //2nd quadrant
        else if((n[i].destlength <= n[i].length) && (n[i].destbreadth >=n[i].breadth))
                {
                        if((n[i].destlength-n[i].length)==0)
                                theta=3.142/2.0;
        else
{x=(double)(n[i].destbreadth-n[i].breadth)/(double)((-1)*(n[i].destlength-n[i].length));
            theta=atan(x);
                        }
                        n[i].length=n[i].length - (distancecovered * cos(theta));
                        n[i].breadth=n[i].breadth + (distancecovered*sin(theta));
        }
                //3rd quadrant
        else if((n[i].destlength <= n[i].length) && (n[i].destbreadth <= n[i].breadth))
                {
                        if((n[i].destlength-n[i].length)==0)
                                theta=3.142/2.0;
      else
{x=(double)((-1)*(n[i].destbreadth-n[i].breadth))/(double)((-1)*(n[i].destlength-
n[i].length));
            theta=atan(x);
                        }
                        n[i].length=n[i].length - (distancecovered * cos(theta));
                        n[i].breadth=n[i].breadth - (distancecovered*sin(theta));
        }
                //4th quadrant
        else if((n[i].destlength >= n[i].length) && (n[i].destbreadth <=n[i].breadth))
                {
                        if((n[i].destlength-n[i].length)==0)
                                theta=3.142/2.0;
                else
{x=(double)((-1)*(n[i].destbreadth-n[i].breadth))/(double)(n[i].destlength-n[i].length);
            theta=atan(x);
                        }
                        n[i].length=n[i].length + (distancecovered * cos(theta));
                        n[i].breadth=n[i].breadth - (distancecovered*sin(theta));
                }

        }

        fprintf(fp,"\n\nNew position:%dX%d",n[i].length,n[i].breadth);
        if (n[i].time1>n[i].tcert)
                n[i].tcert=0;
        else
        n[i].tcert=n[i].tcert-n[i].time1;
        //finding out the certificate validity period
        fprintf(fp,"\n\nThe time remaining is %d",n[i].tcert);
        n[i].pausetime=rand()%20;
        //initalizing the pause time for each node
        while(n[i].pausetime==0)
                n[i].pausetime=rand()%20;
        fprintf(fp,"\n\nQos :%d",n[i].qos);
        fprintf(fp,"\n\nPause time :%d",n[i].pausetime);
        fprintf(fp,"\n\nReply time :%d",n[i].replytime);
```

```
// this is done in this function as each node can have diff pause times
//     based on the destination it has stopped.

}



/*************************Function neighbourhood()***********************/
//The aim of this function is to fill in the start up values for trust
//all the nodes in the network scan their respective neighbors& get some
//startup values of trust.
void neighbourhood()


{
  int i,j;
  int refnode,l=0;
  int reflength,refbreadth;
  float sa,sq,si;

  for(i=1;i<nos;i++)
//scanning the neighbors,transmission range is the nodes x,y position+100
  { printf("\n\nThe neighbors of node %d are",i);
       refnode=i;
       l=0;
       reflength=n[i].length+100;
       refbreadth=n[i].breadth+100;

   for(j=1;j<nos;j++)

   {  if (refnode==j&&j!=(nos-1))
       j=j+1;
          if(refnode==j&&j==(nos-1))
                 break;
   //calculating the trust values
  /*description of neighbors1 [][] array, rows represent the nodes id,
   cols:0-pausetime,1-QOS,2:no of interactions in which availability was good
   3:no of interactions in which QOS was good,4:no of satisfying interactions
   5:total no of interactions,6:satisfaction as a service provider
   7: satisfaction as a recomender, 8: comments(0:suspect,1:convicted)
   9:Trust*/
          if ((n[j].length<=reflength)&&(n[j].breadth<=refbreadth))
             {
                      if(n[j].live==1)
                        { printf("\n\n%d",j);
                          n[i].neighbors[l]=j;
                                 n[i].neighbors1[j][0]=n[j].pausetime;
                                 n[i].neighbors1[j][1]=n[j].qos;
                                 //calulating satisfaction form availability
                                 sa=(float)n[j].pausetime/n[i].pausetime;
                                 if (sa>=n[i].wa)
                                        n[i].neighbors1[j][2]=1;
                                 else
                                n[i].neighbors1[j][2]=0;
                                 //calculating staisfaction from QOS
                                 sq=(float)n[j].qos/n[i].qos;
                                 if(sq>=n[i].wq)
                                        n[i].neighbors1[j][3]=1;
                                 else

                            n[i].neighbors1[j][3]=1;

                         if( (n[i].neighbors1[j][2]==1) && (n[i].neighbors1[j][3]==1))
                                 n[i].neighbors1[j][4]=1;
                                 else
                               n[i].neighbors1[j][4]=0;

                                        n[i].neighbors1[j][5]=1;

                                       si=sa*n[i].wa+sq*n[i].wq;
```

```
                                        n[i].neighbors1[j][6]=si;
                                        if(si>=st)
                                        n[i].neighbors1[j][9]=100;

                                        n[i].neighbors1[j][7]=0.5;


                                        __time64_t ltime;
                      _time64( &ltime );
                        //recording the timestamp of experience
                         n[i].timestamp[j]=ltime;
                      l++;
                         }
              }


       }
    }

 }


/**********************************Function communication ()******************/
// This function describes the nodes communication in the network
void communication(int k)
{
        //declaration of local variables
        int l=1,reflength,refbreadth,refnode,j,i,target,flag1=0,flag2=0;
        int choice,request=0,reply=0,commutime,splflag=0,ID=0;
        double si,sq,sa;
        int index=0,inode,ans,node;


        //intially a check is made if there is a valid certificate or not
        fprintf(fp,"\n\nNODE %d",k);
               for(i=0;i<50;i++)
                       n[k].neighbors[i]=0;


    //since there is a valid certificate it has to determine the node with which
         //it wants to communicate

         refnode=k;
         reflength=n[k].length+100;
         refbreadth=n[k].breadth+100;


for(j=1;j<nos;j++)

   {
           if (refnode==j&&j!=(nos-1))
        j=j+1;
           if(refnode==j&&j==(nos-1))
                  break;

           if ((n[j].length<=reflength)&&(n[j].breadth<=refbreadth))
         {
                 if(n[j].live==1)
                 {   n[k].neighbors[l]=j;
                        n[k].neighbors1[j][0]=n[j].pausetime;
                        n[k].neighbors1[j][1]=n[j].qos;
                 l++;
                 }
         }

   }
   if (l==1)
   {fprintf(fp,"\n\nThe node%d has no neighbors to communicate",k);
        flag1=1;
   }
   if(n[k].tcert>0&&n[k].mode!=4&&flag1!=1)
   {
   //each node scans its neighborhood for any observable malicious activity
   //it floods the accusation in the network.
  intruders(k);

  //now the source nodes picks up a node for communication, based upon weight given for 
the
  //availability& qos the choice is made.
  fflag=0;

  if(n[k].wa>=n[k].wq)
  { choice=1;
    targetnode(choice,k);
  }

  else
  {  choice=2;
      targetnode(choice,k);
  }





   fprintf(fp,"\n\nThe destnode of Node %d is %d",k,destnode);
        if(destnode==0)
               fprintf(fp,"\n\n The node %d has no neighbor to communicate",k);


//once a choice for the destnode is made, both the nodes exchange their
//      qos, availability, pausetime information

 //pausetime,ie availability is calculated
        if(destnode!=0)

        {
               if (n[k].mode==2)//if a node is in mode 2 it will do a spoofing attack by 
changing
                                    //its ID
                  do
                  {
           ID=rand()%30;
                  }while(ID==0&&ID==k&&ID==destnode);
               else
                       ID=k;
    //The source node calculates the availability of the target node
    // based on its reply time
               n[destnode].availability=n[destnode].pausetime-n[destnode].replytime;
               if(n[destnode].availability<0)
                       n[destnode].availability=0;

               if(n[destnode].availability>=n[destnode].tcert)
                       n[destnode].availability=n[destnode].tcert;


               if(n[k].pausetime>=n[k].tcert)
                       n[k].pausetime=n[k].tcert;
               tfile1=n[k].qos*n[k].pausetime;
               if(tfile1>n[k].filesize)
                       tfile1=n[k].filesize;

                n[k].neighbors1[destnode][0]=n[destnode].availability;
                n[k].neighbors1[destnode][1]=n[destnode].qos;

                n[destnode].neighbors1[k][0]=n[k].pausetime;
                n[destnode].neighbors1[k][1]=n[k].qos;
                n[destnode].nrequest=1;

//the nodes can communicate for the time the source is available
```

```
//The availability of the destination is also important
//The useful communication is calculated: This is the file size transferred
//based on the Tcert of source, destination, reply time of destination, pausetimes

                commutime=n[k].pausetime;
                if(n[destnode].availability<n[k].pausetime)
                        commutime=n[destnode].availability;
                //now the file is transfered

                tfile=n[destnode].qos*commutime;

                if(tfile>tfile1)

                        tfile=tfile1;
//commureq is the actual file size that the source can transfer based on its
//Tcert without taking the factors into account
                commureq=commureq+tfile1;
//commudone is the file transferred after taking all the factors into account
                commudone=commudone+tfile;

                usefulcommu=(double)commudone/(double)commureq *100;
        fprintf(fp,"\n\nThe percentage of useful communication is %f",usefulcommu);

//now the satisfactions are calculated on both sides, timestamps are recorded

                //source side

                sa=(float)n[destnode].availability/n[k].pausetime;
                if(sa>=n[k].wa)
                {
                        n[k].neighbors1[destnode][2]=n[k].neighbors1[destnode][2]+1;
                     index=index+1;
                }
                sq=(float)n[destnode].qos/n[k].qos;

                if(sq>=n[k].wq)

                { n[k].neighbors1[destnode][3]=n[k].neighbors1[destnode][3]+1;
                   index=index+1;
                       }

                if(index==2)

                        n[k].neighbors1[destnode][4]=n[k].neighbors1[destnode][4]+1;
                    n[k].neighbors1[destnode][5]=n[k].neighbors1[destnode][5]+1;

                        si=sa*n[k].wa+sq*n[k].wq;
                        n[k].neighbors1[destnode][6]=si;

                      if(si>=st)
                        {if(n[destnode].replytime>=timedelay)
                               n[k].neighbors1[destnode][9]=90;
                        else
                         n[k].neighbors1[destnode][9]=100;
                        }
                        else
                          n[k].neighbors1[destnode][9]= n[k].neighbors1[destnode][9]-10;
                        if(n[destnode].replytime>=timedelay)
                         n[k].neighbors1[destnode][9]= n[k].neighbors1[destnode][9]-10;



                      //recording timestamp
                        __time64_t ltime;
                      _time64( &ltime );

                       n[k].timestamp[destnode]=ltime;
                       n[k].tcert=n[k].tcert-n[k].pausetime;

                       fprintf(fp,"\n\nThe tcert of node is %d",n[destnode].tcert);
                       index=0;
                    //destination
                       sa=(float)n[ID].pausetime/n[destnode].pausetime;
                the(sa>=n[destnode].wa)
                {
                        n[destnode].neighbors1[ID][2]=n[destnode].neighbors1[ID][2]+1;
                     index=index+1;
                }
                sq=(float)n[ID].qos/n[destnode].qos;

                if(sq>=n[destnode].wq)

                { n[destnode].neighbors1[ID][3]=n[destnode].neighbors1[ID][3]+1;
                   index=index+1;
                       }

                if(index==2)

                        n[destnode].neighbors1[ID][4]=n[destnode].neighbors1[ID][4]+1;
                    n[destnode].neighbors1[ID][5]=n[destnode].neighbors1[ID][5]+1;

                        si=sa*n[destnode].wa+sq*n[destnode].wq;
                        n[destnode].neighbors1[ID][6]=si;
                       if(si>=st)
                        n[destnode].neighbors1[ID][9]=100;
                        else
                          {if(n[destnode].neighbors1[ID][5]==1)
                         n[destnode].neighbors1[ID][9]=90;
                          else
                        n[destnode].neighbors1[ID][9]= n[destnode].neighbors1[ID][9]-10;
                          }

             //recording timestamp

                      _time64( &ltime );

                       n[destnode].timestamp[ID]=ltime;

//handling DOS attacks
//if the no of requests exceed the threshold of requests, the destination node
//floods a message that a source node is a Dos attacker.
                       while(n[k].mode==3)
                       {n[destnode].nrequest=n[destnode].nrequest+1;
                              if(n[destnode].nrequest>=requests)
                              { splflag=1;
                               n[destnode].neighbors1[k][8]=1;
                               n[destnode].prevaccused=k;
                               n[destnode].comments[k][0]=2;
                               accusation(destnode,k);
                              break;
                              }
                       }


    //now the two nodes exchange recommendation letters
                       if(splflag==1)
                       n[destnode].neighbors1[k][9]=0;
                       n[destnode].recoletters[k][0]=1;
                       n[destnode].recoletters[k][1]=n[k].neighbors1[destnode][9];
                       n[k].recoletters[destnode][0]=1;
                       //detecting spoofing attack
                       if(ID!=k)
                       {
            n[k].recoletters[destnode][1]=0;
                       accusation(destnode,k);
                       }
                       else
            n[k].recoletters[destnode][1]=n[destnode].neighbors1[k][9];
                       //recording the timestamp the recos were exchanged
                        _time64( &ltime );
                       n[k].rtimestamp[destnode]=ltime;
                       n[destnode].rtimestamp[k]=ltime;
                       n[destnode].tcert=n[destnode].tcert-n[destnode].availability;

                       fprintf(fp,"\n\nThe tcert of destnode is %d",n[destnode].tcert);
            //the following block is executed if a node got recommendations for
                       //the destination node
                       //trust evolution for the nodes that recommended
              //when the performance was as expected
                       if(n[k].splflag==1)
                       {n[k].splflag=0;
                       //positive recomenders
                       if(n[k].neighbors1[destnode][6]>=st)
                         {for(i=0;i<nos;i++)
                       { target=n[k].posrecos[i];
                          if(target>0)
                          n[k].neighbors1[target][7]=0.6*n[k].neighbors1[target][7];
                          }
              //negative recommenders
               for(i=0;i<nos;i++)
                          { target=n[k].negrecos[i];
                          if(target>0)
                          n[k].neighbors1[target][7]=0.6*n[k].neighbors1[target][7]-0.4;
                          }

                         }
                         //when the performance was not as expected
                         //positive recommenders
                         if(n[k].neighbors1[destnode][6]<st)
                        {for(i=0;i<nos;i++)
                       { target=n[k].posrecos[i];
                          if(target>0)
                          n[k].neighbors1[target][7]=0.6*n[k].neighbors1[target][7]-0.4;
                          }
                 //negative recommenders
               for(i=0;i<nos;i++)
                          { target=n[k].negrecos[i];
                          if(target>0)
                          n[k].neighbors1[target][7]=0.6*n[k].neighbors1[target][7];
                          }

                         }
                       }


           //if their tcerts are 0 a request for revocation is sent
                       if(n[k].tcert==0)
                       {
                        certreq[m2]=k;
                        m2++;
                       }
                // if the node is in mode 5 the following is done
                // initially scan all the neighbors
                     if(n[k].mode==5)
                       {
                       n[k].cycle=n[k].cycle+1;
                       if(n[k].cycle==thresh+1)
                       {n[k].cycle=1;
                       }
                       if(n[k].cycle==1)
                       {
                 do
                    {
                     inode=rand()%nos;
                     }while(inode==0&&inode==k);

                     while(n[inode].live==0)

                     {inode=rand()%nos;
```

```
                     }
                n[k].inode=inode;
                 }
                      inode=n[k].inode;
                      n[destnode].mode=6;
                      n[destnode].inode=inode;
                      n[k].neighbors1[inode][8]=1;
                      accusation(k,inode);
                      n[k].prevaccused=inode;
                      n[k].comments[inode][0]=1;
                      fprintf(fp,"\n\nInnocent node  is %d",inode);
                      }



  if(n[k].mode==6)

  {inode=n[k].inode;
  n[k].prevaccused=n[k].inode;
  n[k].comments[inode][0]=-1;
  n[k].comments[inode][1]=n[k].prevnode;
  accusation(k,inode);
  n[k].neighbors1[inode][8]=1;
  n[k].mode=n[k].prevmode;
    }


  //here each node checks the comments array to make sure that it was not compromised

             for(i=1;i<50;i++)
                       {
                              ans=n[k].comments[i][0];
                            if(ans==-1)
                                { node=n[k].comments[i][1];

                   accusation(k,node);
                                }

                       }

                       n[k].prevnode=destnode;
                       n[destnode].prevnode=k;

}
}
else
{
if(n[k].tcert==0)
{
       certreq[m2]=k;
       m2++;
       fprintf(fp,"\n\nThe node doesnt have a valid certificate to communicate");
       flag2=1;
}

if(n[k].mode==4)
{
fprintf(fp,"\n\nThe node is under Dos attack");
n[k].tcert=0;
if(flag2==0)
{
certreq[m2]=k;
       m2++;
}

      tfile=n[k].qos*n[k].pausetime;
         if(n[k].tcert<n[k].pausetime)
                n[k].pausetime=n[k].tcert;
          if(tfile>n[k].filesize)
                 tfile=n[k].filesize;
                commureq=commureq+tfile;
}
                usefulcommu=(double)commudone/(double)commureq *100;
                fprintf(fp,"\n\nThe percentage of useful communication is
%f",usefulcommu);

}
index=0;
flag1=0;
flag2=0;

}


/*****************************Function intruders()*****************************/
//In this function each node looks around its neighborhood for
//malicious activity and floods messages
void intruders(int k)
{
        int i=k,target,j;

         for(j=1;j<nos;j++)
         {
                target=n[i].neighbors[j];
            if(target!=0&&n[target].mode==2)
           {
                  n[i].neighbors1[target][8]=1;
                  //the node previously accused
                  n[i].prevaccused=target;
                  //the reason for accusing
                  n[i].comments[target][0]=1;
                  //it floods the accusation in the network
                  accusation(i,target);

       }
     }

}
/*****************************Function accusation()***********************/

        void accusation(int accuser,int badnode)
        { int i,row,col,sum=0;

         for(i=1;i<nos;i++)
         {

          if(i!=accuser&&i!=badnode)
         {

                row=badnode;
                col=accuser;
                //Each node accepts the accusation only if it has
                //trust on the accuser
                if(n[i].neighbors1[col][8]!=1)
                {
                        n[i].neighbors1[row][6]=0;
                        n[i].neighbors1[row][8]=1;
                        n[i].prevaccused=row;
                        n[i].comments[row][0]=4;
                }
                else
                        n[i].neighbors1[row][8]=0;

        }
         }
        }

/*******************************Function targetnode()*****************************/
//each node calls this function to find an appropriate node to communicate
        //based on its weight given to availability&Qos
        void targetnode(int choice,int k)
        {int i,max,target;

     if(choice==1)


  {  destnode=n[k].neighbors[1];
          max= n[k].neighbors1[destnode][0];

    for(i=1;i<nos;i++)
        {
          target=n[k].neighbors[i];
            if(target>0)
               {
               if(n[k].neighbors1[target][0]>=max)
               {destnode=target;
                max=n[k].neighbors1[destnode][0];
               }
               }
     }


  }
  if(choice==2)
          {  destnode=n[k].neighbors[1];
          max= n[k].neighbors1[destnode][1];

    for(i=1;i<nos;i++)
        {
          target=n[k].neighbors[i];
          if(target>0)
          {
               if(n[k].neighbors1[target][1]>=max)
               {destnode=target;
                max=n[k].neighbors1[destnode][1];
               }
          }
     }

         }
          //once the destination node is picked, a check is made if there
          //is a direct experience with the node, satisfaction from it,
          //if its convicted etc.,if the results are not satisfactory the
          //next best node is picked and the process is repeated
     if(destnode!=0)
     check(choice,k,destnode);
         if(fflag==1)
         {fflag=0;
         targetnode(choice,k);

         }

        }
/********************************Function check()*************************/
void check(int choice,int k,int destnode)
{long int timediff;
 int i,reco;
  __time64_t utime;


  if(
n[destnode].live==0||n[destnode].mode==2||n[destnode].mode==6||n[destnode].tcert==0)
  {  //that node is not chosen for communication
          for(i=1;i<nos;i++)
          {
                 if(n[k].neighbors[i]==destnode)
             n[k].neighbors[i]=0;
          }
      fflag=1;


  }
  //a check is made if the node is convicted
  if(n[k].neighbors1[destnode][8]==1&&fflag==0)

  {
           for(i=1;i<nos;i++)
         {
                 if(n[k].neighbors[i]==destnode)
            n[k].neighbors[i]=0;
         }
      fflag=1;


  }


    _time64( &utime );
       //a check is made if there is direct experience with the target node

    if(n[k].neighbors1[destnode][6]!=0&&fflag==0)
       {timediff=utime- n[k].timestamp[destnode];
     //if there is direct experience it should not be older than 20sec
       if(timediff<20)
      { if(n[k].neighbors1[destnode][6]<st)
              { for(i=1;i<nos;i++)
                        {if (n[k].neighbors[i]==destnode)
                           n[k].neighbors[i]=0;
                          }
             fflag=1;

          }
         }
       }
       //if there is a outdated experience or no direct experience
       //the node seeks recommendation
        timediff=utime- n[k].timestamp[destnode];
       if(timediff>=20&&fflag==0||n[k].neighbors1[destnode][5]==0&&fflag==0)

      { reco=recomendation(destnode,k);
      if(reco==0)
         {for(i=1;i<nos;i++)
         {if(n[k].neighbors[i]==destnode)
          n[k].neighbors[i]=0;
         }//if the recomendations are not satisfactory then another node
          //is picked up for communication.
       fflag=1;

         }
         else
                 n[k].splflag=1;

   }

}
/******************************Function recomendation()*******************/
int recomendation(int destnode,int k)

{ float
priorprob,postprob,denom,hyp,evidence,trustref1=0,trustref2=0,untrust1=0,untrust2=0;
  float
untrust3=0,trustref3=0,untrust21=0,trustref21=0,untrust31=0,trustref31=0,ra,rq,sr;
  float w1=0.4,w2=0.2,w3=0.3,w4=0.1,priorprob1,postprob1,hyp1;
  long int timediff;
  int i,l=0,m=0,respond=0,flag11=0,reply;
  __time64_t ltime;


       for(i=1;i<nos;i++)
   { //nodes could be selfish they may not send recommendations
          reply=rand()%2;
      if(n[i].neighbors1[destnode][5]>=1&&i!=k&&i!=destnode&&reply==1)

//first the p(e/h) is calculated

    //for availability
   {   respond++;//counting the no of nodes replying

   denom=(float)n[i].neighbors1[destnode][4]/(float)n[i].neighbors1[destnode][5];

    priorprob=(float)n[i].neighbors1[destnode][2]/(float)denom;
    //fpr QOS
    priorprob1=(float)n[i].neighbors1[destnode][3]/(float)denom;

    hyp=(float)n[i].neighbors1[destnode][2]/(float)n[i].neighbors1[destnode][5];
    hyp1=(float)n[i].neighbors1[destnode][3]/(float)n[i].neighbors1[destnode][5];
    evidence=(float)n[i].neighbors1[destnode][4]/(float)n[i].neighbors1[destnode][5];
 //calculating post probability using Bayes theorem
postprob=(float)(priorprob*hyp)/(float)evidence;
    postprob1=(float)(priorprob1*hyp1)/(float)evidence;
    //calculating satisfaction
    sr=n[k].wa*postprob+n[k].wq*postprob1;
    //positive recommenders
     if(sr>=st)
     {
            n[k].posrecos[l]=i;
     l++;
     }
     else
     { //negative recomenders
            n[k].negrecos[m]=i;
     m++;
     }
     _time64( &ltime );


 timediff=ltime-n[i].timestamp[destnode];
     trustref1=trustref1+n[k].neighbors1[i][7];
     if(n[k].neighbors1[i][5]>=1)
     {//if the experience is latest
     if(timediff<=20)
        {
            postprob=postprob*n[k].neighbors1[i][7];
             postprob1=postprob1*n[k].neighbors1[i][7];
            trustref2=trustref2+postprob;
            trustref21=trustref21+postprob1;
           }


     else
        {//if the experice is outdated it is given less weight
            postprob=postprob*n[k].neighbors1[i][7];
            postprob1=postprob1*n[k].neighbors1[i][7];
            trustref3=trustref3+postprob;
            trustref31=trustref31+postprob1;
         }

     }
     else
       { //if the recommenders are unknown
                  untrust1++;
                  _time64( &ltime );
        timediff=ltime-n[i].timestamp[destnode];
     if(timediff<=20)
     {
            untrust2=untrust2+postprob;
        untrust21=untrust21+postprob1;
     }
     else
     {
            untrust3=untrust3+postprob;
        untrust31=untrust31+postprob1;
```

```
          }

        }


   }
   }
   //calculating recommendation for availability & QOS

ra=w1*(float)(trustref2/trustref1)+w2*(float)(trustref3/trustref1)+w3*(float)(untrust2/un
trust1)+w4*(float)(untrust3/untrust1);

rq=w1*(float)(trustref21/trustref1)+w2*(float)(trustref31/trustref1)+w3*(float)(untrust21
/untrust1)+w4*(float)(untrust31/untrust1);
        //Calculating satisfaction
 sr =n[k].wa*ra+n[k].wq*rq;
 if(sr>=st)
         return(1);
 else
         if(respond<thresh)//if the no of recommenders is less than threshold
         {
      for(i=1;i<nos;i++)
          {    _time64( &ltime );

         //the target nodes recommendation letters are verified
      timediff=ltime-n[destnode].rtimestamp[i];
                  if( n[destnode].recoletters[i][0]>0&&timediff<=20)
                  {
                         if(n[k].neighbors1[i][8]!=1)
                         { if(n[destnode].recoletters[i][1]>=50)
                            flag11=1;
                         }
                  }
          }

          if(flag11==0)
           return(0);
          else
                  return(1);

         }

}


/*********************************Function revocation()*****************/
void revocation()
{ //now the request for the certificates is flooded in the network, certificate is
granted

// one by one to each node.

int i,j,n2,n3,n11,n22,member=0,refnode,l=1,k,reflength,refbreadth,recos=0,precos=0;
 __time64_t utime;
 long int timediff;
 int flag3=0;
//the neighbors for each node are scanned again
// since the positions may have turned out 0 in the routine check
for(i=1;i<nos;i++)
{
k=i;
 refnode=k;
         reflength=n[k].length+100;
         refbreadth=n[k].breadth+100;


for(j=1;j<nos;j++)

   {
           if (refnode==j&&j!=(nos-1))
        j=j+1;
```

```
          if(refnode==j&&j==(nos-1))
                  break;

          if ((n[j].length<=reflength)&&(n[j].breadth<=refbreadth))
        {
                if(n[j].live==1)
                {   n[k].neighbors[l]=j;
                        l++;
                }
        }

   }

   l=1;
}

for(i=1;i<nos;i++)

{
  if (certreq[i]>0)

  {
         n2=certreq[i];

   for(j=1;j<nos;j++)
   {
    n3=n[n2].neighbors[j];
       if (n[n3].live==0||n[n3].mode==4)
               //since a node under dos attack cannot grant certificate
       {
        n[n2].neighbors[j]=0;
        n3=0;
       }
       if(n3>0)
       {   member++;
               n[n3].request[i]=n2;
       }
   }

   fprintf(fp,"\n\nThe Node %d has %d neighbours",n2,member);
   if(member<thresh)
   {
   fprintf(fp,"\n\n The certificate for %d cannot be revoked due to insufficient
neighbours",n2);
   certreq[i]=0;
   }
   member=0;

  }
}
//now each nodes decide whether it has to revoke the certificate or not

for(i=1;i<nos;i++)

{
       n[i].reply=0;
}



for(i=1;i<nos;i++)
{
       for(j=1;j<nos;j++)
       { if(n[i].request[j]>0)
       {
           n11= n[i].request[j];
               if(n[i].neighbors1[n11][5]>=1)

               { if(n[i].neighbors1[n11][8]!=1)
                  {
                    _time64( &utime );
           timediff=utime-n[i].timestamp[n11];

                 if(timediff<=20)
                    {
                               if(n[i].neighbors1[n11][6]>=st)
                  n[n11].reply=n[n11].reply+1;
                // this is because a genuine node may suffer a reject due to mis-
satisfaction, hence
                 //it is given a chance for 5 times if it has a remark that
               // its behavior is bad then it will not get a certificate

                               if((n[i].neighbors1[n11][6]<st)&&(n[n11].mode!=5))
                               {n[n11].cycle=n[n11].cycle+1;
                                 if(n[n11].cycle<=5)
                                n[n11].reply=n[n11].reply+1;
                               }

                        }
                 // if the experience is outdated, the target nodes recommendation
          //letter are reviewed
               else
                       {for(k=1;k<nos;k++)
                       {if(n[n11].recoletters[k][0]==1)
                         {
                            recos++;

                              if(n[n11].recoletters[k][1]>=50)
                    precos++;
                        }
                              }

                              if(recos==precos)

                 n[n11].reply=n[n11].reply+1;

                              else
                              {
                              if(n[n11].mode!=5)
                               {n[n11].cycle=n[n11].cycle+1;
                                 if(n[n11].cycle<=5)
                                n[n11].reply=n[n11].reply+1;
                               }
                              }

                       }
                 }
               }
               //if there is no direct experience
             recos=0,precos=0;
               if(n[i].neighbors1[n11][5]==0)
               {

                  if(n[i].neighbors1[n11][8]!=1)
                  {
                          for(k=1;k<nos;k++)
                       {if(n[n11].recoletters[k][0]==1)
                          {    recos++;

                              if(n[n11].recoletters[k][1]>=50)
                        precos++;
                        }
                              }

                              if(recos==precos)

                 n[n11].reply=n[n11].reply+1;
                              else
                              {

                  if(n[n11].mode!=5)
                               {n[n11].cycle=n[n11].cycle+1;
```

```
                                  if(n[n11].cycle<=5)
                                 n[n11].reply=n[n11].reply+1;
                                }
                               }

                  }
               }

        }

}
}



// now each node forms a certificate based on the no of replies it has got

        for(i=1;i<nos;i++)

        {if( certreq[i]>0)

        { n22=certreq[i];

      if(n[n22].reply>=thresh)
          { n[n22].tcert=20;
           fprintf(fp,"\n\nCERTIFICATES ARE REVOKED");
           fprintf(fp,"\n\nThe Certificate of Node %d is revoked",n22);
          }
          else
          {
                 fprintf(fp,"\n\nThe node %d is kicked out of the network ",n22);
              n[n22].live=0;
                 if(n[n22].mode==5)
                         hijack++;
                 if(n[n22].mode==1||n[n22].mode==6)
                 {

                         for(i=26;i<=30;i++)
                       {
                               if(n[i].prevaccused==n22||n[i].inode==n22)
                                flag3=1;
                       }
               if(flag3==1)



                                  innocent++;
                          else
                                  unsatis++;

                 }
                 flag3=0;

                 if(n[n22].mode==2)
                         bad++;
                 if(n[n22].mode==3)
                         dos++;


                 }
        }

        }
        fprintf(fp,"\n\n innocent :%d",innocent);
       fprintf(fp,"\n\n unsatisfying :%d",unsatis);
       fprintf(fp,"\n\n Dos:%d",dos);
       fprintf(fp,"\n\n Hijack :%d",hijack);
       fprintf(fp,"\n\n Bad :%d",bad);
        good=15-innocent-unsatis;
       fprintf(fp,"\n\n Good nodes remaining :%d",good);


 }

/*******************End Of Simulation(Scheme 2)*****************************/
```

VITA

Sudha Chinni

Candidate for the Degree of

Master of Science

Thesis: BAYESIAN NETWORK TRUST MODEL FOR CERTIFICATE REVOCATION IN ADHOC WIRELESS NETWORKS.

Major Field: Computer Science

Biographical:

Personal Data: Born in Eluru, Andhra Pradesh, India on October 28, 1980, the daughter of Mr. C.Anjaneyulu and Mrs. C.Nirmala.

Education: Received Bachelor of Engineering in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad, India In April, 2002.Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December, 2004.

Professional Experience: January 2004 – Present: Computer Programmer, Dept of Plant & Soil Sciences, Oklahoma State University, Stillwater.