uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Namita Singh

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Secure Communication Using Elliptic Curve Cryptosystem in Ad Hoc Network

TITRE DE LA THÈSE / TITLE OF THESIS

Amiya Nayak

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Ali Miri

Chung-Horng Lung

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

SECURE COMMUNICATION USING

ELLIPTIC CURVE CRYPTOSYSTEM IN AD HOC NETWORK

Namita Singh

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the MSc degree in Computer Science

School of Information Technology and Engineering

University of Ottawa

Ottawa, Ontario

© Namita Singh, Ottawa, Canada, 2008

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-48508-8
Our file Notre référence
ISBN: 978-0-494-48508-8

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Ad hoc networks are standalone networks supporting "communication anytime and anywhere" using portable devices like PDAs, cell phones, laptops etc. which require no predefined organization of available links but offer constraints such as battery life, bandwidth, memory, computational ability, security, quality of service, reliability, range of the device and speed. Security framework is essential and relies on certificates to communicate with each other but requires higher battery life, bandwidth and memory space.

Researchers have been using keys as an alternative. However, no protocol is complete solution due to the presence of large key lengths and high bandwidth usage. Therefore, an efficient key management system is proposed using Elliptic curve cryptosystem (ECC) aiming at secure communication among the nodes concentrating mainly on key generation, agreement and encryption/decryption with an assumption that the nodes have capabilities for efficient key storage and key security. In other words, the goal is to enable sender nodes to transmit data from sender to the destination without threats.

Acknowledgements

I would like to thank **Prof. Amiya Nayak** of School of Information Technology & Engineering, University of Ottawa and **Mr. Achin Jain**, Norleaf Networks, Gatineau, for their guidance and support during my thesis work.

I am sincerely grateful to **Dr. Nishith Goel**, Cistel Technologies and **Dr. Venkat Narasimhan**, Visiting Professor, University of Ottawa, for their support and advice.

I would also like to thank my family and friends for giving me moral support during difficult times.

TABLE OF CONTENTS

Abstract ii
Acknowledgements iii
List of Figures vii
List of Tables ix
List of Algorithms ix
List of Abbreviations x
List of Standard Notations xii
Chapter 1 1
Introduction 1
1.1. Overview 1
1.2. Motivation 3
1.3. Main Contributions 4
1.4. Thesis Outline 5
Chapter 2 6
Ad hoc Security and Existing Solutions 6
2.1. Security - Overview 6
2.2. Security Threats 7
2.3. Security Attacks 7
2.4. Security Goals 8
2.5. Security Techniques 9
2.6. Existing Solutions 10
2.6.1. Certificates vs. Keys 10
2.6.2. Key Management System 11
2.6.3. Encryption Schemes 13
2.6.4. Ad Hoc Routing and Secure Routing Techniques 16
Chapter 3 21
Elliptic Curve Ad Hoc Security Scheme 21
3.1. Encryption-Overview 21
3.2. ECC – Overview 23
3.2.1. Key Agreement and Exchange Protocols 24
3.2.2. Elliptic Curve Integrated Encryption Scheme (ECIES) 27
3.3. ECC Based Encryption/Decryption Primitives 29
3.3.1. Elliptic Curve Domain Parameter Generation 30
3.3.2. Elliptic Curve Domain Parameters over F_q Validation Primitive 31
3.3.3. Elliptic Curve Key Pair Generation 32

3.3.4. Validation of Elliptic Curve Public keys 33
3.3.5. ECDH-KAS Primitives 34
3.3.6. ECIES-Encryption/Decryption Primitives 35
Chapter 4 **37**
Implementation of Proposed Ad Hoc Security Scheme **37**
4.1. Designing Proposed Scheme 37
4.2. The Proposed Scheme 39
4.3. Implementation Details 42
4.3.1. Scheme-Inputs and Outputs 43
4.3.2. Scheme Information Flowchart 44
4.4. Implementation 45
4.4.1. Initial parameter Generation Module 46
4.4.2. Key Generation/Validation Module 47
4.4.3. Key Agreement Module 47
4.4.4. Encryption/Decryption Module 48
Chapter 5 **52**
Development of the Testbed **52**
5.1. Testbed – An Overview 52
5.2. Design of the Testbed 53
5.3. Factors Affecting the Performance of the Scheme 58
5.3.1. Number of Nodes and Their Abilities 59
5.3.2. Nodes and Traffic Patterns 60
5.3.3. Systems and their Abilities 61
5.4. Working of the Testbed 61
5.5. Simulation Parameters 61
5.6. Simulations and its Setup 63
5.6.1. Network Simulator (NS-2) – Overview 64
5.6.2. Graphical User Interface - Overview 66
5.7. Evaluation Parameters 68
5.8. Configuring the Testbed 69
5.9. Interacting with the Testbed 69
5.10. Using Testbed to Read and Visualize Results 70
Chapter 6 **72**
RSA versus ECC – An Experimental Comparison **72**
6.1. Overview 72
6.2. Experimental Setup 73
6.3. Experimental Results 74
6.4. Experimental Data Analysis 74
6.5. Results based on Computing Power of the Computer Systems 84
6.6. Conclusions 89

Chapter 7 91
Experiments with Elliptic Curve Cryptosystem on the Testbed 91
7.1. Experimental Setup 91
7.2. Experimental Results 91
7.3. Experimental Data Analysis 93
7.3.1. AODV with Variable Network Size– Results 93
7.3.2. OLSR with Variable Network Size- Results 97
7.3.3. LAR with Variable Network Size – Results 102
7.4. Experimental Results for Different Routing Protocols- Comparison 106
7.5. Experimental Comparison between SAODV-RSA and SAODV-ECC 110
7.6. Comparative summary 112
7.7. Conclusion 113
Chapter 8 115
Conclusions and Future Work 115
8.1. Conclusions 115
8.2. Future Work 116
Glossary 118
References 121

List of Figures

Figure 1-1: Wireless Infrastructure based Ad Hoc Network 2
Figure 2-1: Key Management System 11
Figure 2-2: Key Lifetime 12
Figure 2-3: General Encryption Overview 13
Figure 3-1: Diffie-Hellman Key Agreement Protocol 25
Figure 3-2: Elliptic Curve Diffie-Hellman Key Agreement Protocol 26
Figure 3-3: Elliptic Curve Integrated Encryption Scheme 27
Figure 3-4: ECC Security Evaluation against RSA (Source: [CER00]) 28
Figure 4-1: Proposed Security Scheme 40
Figure 4-2: Proposed Encryption Scheme's Step-by-Step Flowchart 41
Figure 4-3: Implementation Modules 42
Figure 4-4: Sender-Receiver Information Flowchart 44
Figure 5-1: Testbed Design 54
Figure 5-2: GUI User Module 56
Figure 5-3: GUI System Module 57
Figure 5-4: NS-2 Simulation Model 65
Figure 5-5: GUI Menu 66
Figure 5-6: GUI Main Parameter Selector Window 67
Figure 5-7: Using Nam from the Testbed (Source: [FAL00]) 70
Figure 6-1: Block Diagram of Encryption Engine 73
Figure 6-2: Size of the Encrypted file for Different Algorithms 75
Figure 6-3: Size of the Encrypted file for ECC 256 and RSA 2048 Algorithms 75
Figure 6-4: Size of the Encrypted file for ECC 160 and RSA 1024 Algorithms 76
Figure 6-5: Size of the Encrypted file for ECC 112 and RSA 512 Algorithms 77
Figure 6-6: Total time taken for Encryption/Decryption for Different Algorithms 78
Figure 6-7: Total time taken by ECC 256 and RSA 2048 Algorithms 78
Figure 6-8: Total time taken by ECC 160 and RSA 1024 Algorithms 79
Figure 6-9: Total time taken by ECC 112 and RSA 512 Algorithms 79
Figure 6-10: Percentage Increase in Data Size for Different Algorithms 80

Figure 6-11: Percentage Increase in Data Size for ECC 256 and RSA 2048 Algorithms 81
Figure 6-12: Percentage Increase in Data Size for ECC 160 and RSA 1024 Algorithms 82
Figure 6-13: Percentage Increase in Data Size for ECC 122 and RSA 512 Algorithms 82
Figure 6-14: Average Encryption and Decryption time taken by RSA and ECC 83
Figure 6-15: Total time taken for Encryption/Decryption for RSA 512 on Different Machines 84
Figure 6-16: Total time taken for Encryption/Decryption for RSA 1024 on Different Machines 85
Figure 6-16: Total time taken for Encryption/Decryption for RSA 1024 on Different Machines 86
Figure 6-17: Total time taken for Encryption/Decryption for RSA 2048 on Different Machines 87
Figure 6-18: Total time taken for Encryption/Decryption for ECC 112 on Different Machines 87
Figure 6-19: Total time taken for Encryption/Decryption for ECC 160 on Different Machines 88
Figure 6-20: Total time taken for Encryption/Decryption for ECC 256 on Different Machines 89
Figure 7-1: AODV-Delay for Different Encryption Algorithms 94
Figure 7-2: AODV-NRL for Different Encryption Algorithms 94
Figure 7-3: AODV-PDF for Different Encryption Algorithms 95
Figure 7-4: AODV-Routing Overhead for Different Encryption Algorithms 96
Figure 7-5: AODV-Average Hop Counts for Different Encryption Algorithms 97
Figure 7-6: OLSR-Delay for Different Encryption Algorithms 98
Figure 7-7: OLSR-NRL for Different Encryption Algorithms 99
Figure 7-8: OLSR-PDF for Different Encryption Algorithms 100
Figure 7-9: OLSR-Routing Overhead for Different Encryption Algorithms 101
Figure 7-10: OLSR-Average Hop Counts for Different Encryption Algorithms 102
Figure 7-11: LAR-Delay for Different Encryption Algorithms 103
Figure 7-12: LAR-NRL for Different Encryption Algorithms 103
Figure 7-13: LAR-PDF for Different Encryption Algorithms 104

Figure 7-14: LAR-Routing Overhead for Different Encryption Algorithms 105
Figure 7-15: LAR-Average Hop Counts for Different Encryption Algorithms 106
Figure 7-16: Delay for Different Encryption Algorithms 107
Figure 7-17: NRL for Different Encryption Algorithms 107
Figure 7-18: PDF for Different Encryption Algorithms 108
Figure 7-19: Routing Overhead for Different Encryption Algorithms 109
Figure 7-20: Average Hop Counts for Different Encryption Algorithms 110
Figure 7-21: Packet Delivery Fraction for Different Routing Protocols 111
Figure 7-22: Routing Overhead for Different Routing Protocols 112

List of Tables

Table 3-1: Public Key Sizes (bits) Comparison (Source: [CER00]) 28
Table 4-1: Techniques Input and Output 43
Table 7-1: Performance Scenario 93
Table 7-2: SAODV-ECC – Performance Scenario 111

List of Algorithms

Algorithm *Domparamgen* (Source: [HAN04]) 31
Algorithm *Domparamval* (Source: [HAN04]) 31
Algorithm *Keygen* (Source: [HAN04]) 32
Algorithm *PubKeyval* (Source: [HAN04]) 33
Algorithm *Keyagree* (Source: [HAN04]) 34
Algorithm *Keyencrypt* (Source: [HAN04]) 35
Algorithm *Keydecrypt* (Source: [HAN04]) 35

List of Abbreviations

AODV	Ad Hoc Distance Vector Routing
ARAN	Authenticated Routing for Ad-Hoc Networks
CBR	Constant Bit Rate
D-H	Diffie-Hellman
DHP	Diffie-Hellman Problem
DHKAS	Diffie-Hellman Key Agreement Scheme
DoS	Denial of Service
DP	Domain Parameter
DSA	Digital Signature Algorithm
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
ECAES	Elliptic Curve Augmented Encryption Scheme
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDH-KAS	Elliptic Curve Diffie-Hellman Key Agreement Scheme
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDP	Elliptic Curve Domain Parameters
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ECMQV	Elliptic Curve Menezes-Qu-Vanstone Protocol
GPS	Global Positioning System
GUI	Graphical User Interface
HMAC	Hashed Message Authentication Code
IDE	Identity Based Encryption
IDS	Intrusion Detection System

KDF	Key Derivation Function
KMS	Key Management System
LAR	Location Aided Routing
MAC	Message Authentication Code
MANET	Mobile Ad Hoc Network
Nam	Network Animator
NRL	Normalized Routing Load
NS-2	Network Simulator -2
OLSR	Optimized Link State Routing
O-TcL	Object Tool Command Language
PDF	Packet Delivery Fraction
PKC	Public Key Cryptography
PKG	Private Key Generator
PKI	Public Key Infrastructure
RSA	Rivest, Shamir, Adleman Public Key Algorithm
SAODV	Secure Ad Hoc Distance Vector Routing
SEAD	Secure Efficient Ad Hoc Distance Vector Routing Protocol
SLSR	Secure Link State Routing
SOLSR	Secure Optimized Link State Routing
SRP	Secure Routing Protocol
TCL	Tool Command Language

List of Standard Notations

a,b	Coefficient that defines equation of Elliptic curve
C	Cipher text
D	Elliptic Curve Domain parameters
d	Private key
DEC	Decryption
E	An Elliptic curve over field F_q defined by a and b.
E(Fq)	The set of all elliptic curve E points *over* F_q and including infinity point 0.
# E(Fq)	The number of points on the curve E defined over F_q.
EK	Encryption key
EM	Some octets of C as Octet string
ENC	Encryption
enckeylen	Length of symmetric key in octets
F_q *or* F_p	The finite field containing q elements where q is an odd prime number (p).
FR	Field representation used for elements of F_q
G	Base point or generating point on an elliptic curve
h	cofactor
H	Hash function
K	Keying data
KDF	Key derivation function
keydatalen	Number of octets of keying data required

(k', R)	One particular instance ephemeral key pair
L	Security level
Log 2 q	The logarithm of q to the base 2
m	Plain text message
MAC	Message authentication code
mackeylen	Length of MAC key in octets
maclen	Length of MAC tags in octets
MK	MAC key
mod	Modulo
N	Elliptic curve order
n	Order of base point or generating point G
N_w	Network
p	An odd prime number
P	EC point
Q	Public key
q	The number of elements in the field F_q
R	An EC public key specific to one instance of a cryptographic scheme
R'	An EC public key converted into octet string
S	Seed returned
Sharedinfo	Some data shared by communicating nodes
x□y	Concatenation of two strings x and y(bit strings or both octet strings).
Z	A shared secret value.

Chapter 1

Introduction

1.1. Overview

Ad hoc network (Figure 1-1) a 30 year old technology, is a standalone network organization which supports "communication anytime and anywhere" using portable devices like PDAs, cell phones and laptops. It is formed by autonomous collection of mobile users, with no fixed architecture or pre-determined organization of available links, who wish to communicate directly or through one another. These networks, being relatively inexpensive, prove to be an attractive option for commercial and government users, varying diversely from small limited power static networks to large, mobile, highly dynamic networks. However, constraints like battery life, bandwidth, memory, computational ability, security, quality of service, reliability, range of the device and speed, are constant source of threats [RAM02]. Due to these inherent shortcomings and to provide security, most cryptographers develop algorithms based on certificates [ZHO99]. Alternative for certificates, which require higher battery life, bandwidth and memory space, is keys which offer conservation of bandwidth, battery life and memory space. Key generation, key agreement, key distribution, key exchange, key storage and key security are together known as Key Management System (KMS) [WU07].

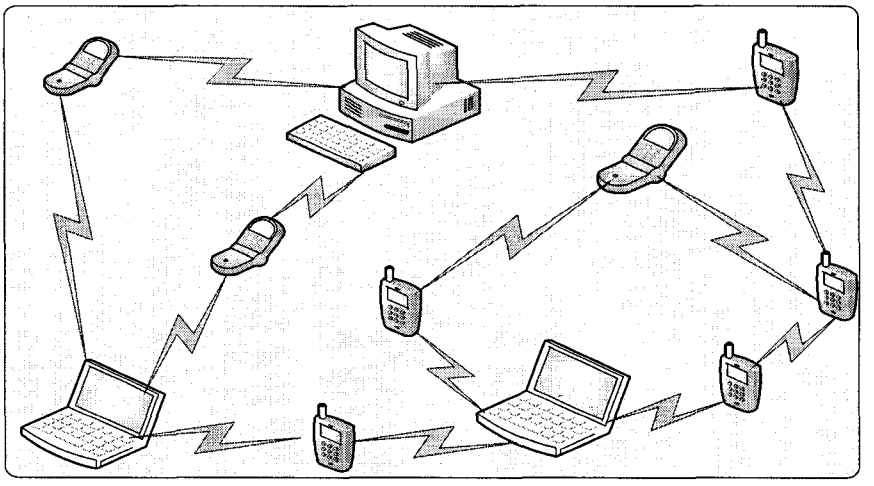

Figure 1-1: Wireless Infrastructure based Ad Hoc Network

The goal of this research is to develop a system to maintain secure communication among the nodes when the link between them is not secure. This could be achieved using Elliptic Curve Cryptography (ECC) in the ad hoc network focusing mainly on key generation, agreement and encryption/decryption with an assumption that the nodes have capabilities for well-organized key storage and key security. Each node in the network must be able to send data to the destination without threats using intermediate nodes but these links from the sender to the receiver are still risky and precarious. These intermediate nodes can be an attacker; therefore, encryption of the message being sent is necessary while maintaining individual security and privacy for the sender-receiver pair. The decryption of this encrypted message must be unfeasible by the intermediate nodes as the actual message can only be decrypted by the secret key attained through key agreement.

Key agreement protocol allows two or more parties to agree on a common secret key in such a way that both influence the outcome [ISL05]. This scheme helps prevent attacks such as eavesdropping which allows intermediate nodes and attacker nodes to listen secretly to the private conversation of others and thus use it to corrupt network. Among many others, Elliptic Curve Diffie-Hellman (ECDH) [HAN04] is the

scheme chosen that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. Once a session key is established, encrypted messages could be sent over.

Key encryption [ISL05] relies heavily on the credentials of the nodes. Once the nodes are identified based on its credentials, security issues are very easy to handle. Using the nodes' individual identities and credentials, keys are generated and used to encrypt the message. Messages can be encrypted using key encryption algorithms such as ECC.

1.2. Motivation

Ad hoc networks are employed where infrastructure support of wireless access or wired backbone is not feasible. Nodes within range and with access to common radio link can easily participate to set up ad hoc alliance. These ad hoc scenarios require physical security and transmission security. Also, with lack of infrastructure and physical security, the inherent security weaknesses are the biggest threats to the network. The most obvious shortcomings [HUB01] are: 1) Lack of predefined infrastructure and physical security, 2) Susceptibility to wireless communication link attacks, 3) Highly dynamic network topology, 4) Limited communication bandwidth and battery power, 5) Size, processing power and storage capacity of mobile devices, 6) Information, communication and security overhead and 7) Scalability in the case of large number of nodes.

Secure communication amongst the network nodes requires safe communication link to transmit messages in the network. This requires network formation based on associated credentials and unique identification of every node. To send confidential information over insecure network link is risky and also the nodes are hesitant to share such sensitive information [VIN02]. This also raises privacy issues since any

compromised information leads attacker to create privacy threat to user device. Unfortunately, current mobile standards grant minimum or no protection. However, if delivered, identity and credentials are to be authenticated and protected so that identity and credentials of the sender node cannot be questioned by the receiver node. Also, every node wants to be sure that delivered identity and credentials are not compromised. These identification and privacy problem needs to be standardized.

Moreover, there is no single scheme which would guarantee security against all the existing problems at once because with the increase in problems, the solution of those problem increases the processing, storage and security overhead considerably. In addition, this type of power cannot be expected from handheld devices as yet. Therefore, this research considers a security scheme which handles security threats by message-level encryption with minimal overhead.

1.3. Main Contributions

The goal of this research is to provide secure communication between two nodes operating under ad hoc mode. As explained earlier, identification plays an important role in securing a network. The following summarizes the present work:

- An encryption scheme for the ad hoc network providing better data security and user security is proposed. It is simulated using Network Simulator 2 (NS-2) [FAL00] which provides real world environment for the ad hoc network.
- To further validate the efficient working of ECC, some experiments are performed to compare ECC against a widely used RSA. Experiments are performed based on three criteria: no encryption, encryption using a standard scheme and encryption using the proposed scheme. The results from these experiments are further analyzed to validate the use of new encryption scheme.

- The comparison of the performance of an insecure network i.e. using no security protocol and secure i.e. using ECDH and Elliptic Curve Integrated Encryption Scheme (ECIES) [HAN04] security protocol is provided. The proposed scheme is tested for secure communication even with the dynamic environment of the ad hoc network and variable data loads. Experiments are also performed on different machines to test the effect of the individual performance of the computer system. The effect of different routing protocols on the performance of the proposed scheme is also evaluated. These results are analyzed and observations are discussed.
- This scheme is tested against other existing schemes such as Secure ad hoc distance vector (SAODV) routing protocol in a predefined environment. Experiments are done to validate the improvement of this scheme in comparison to the other standard schemes currently used widely for encryption purposes.

1.4. Thesis Outline

In Chapter 2, the security threats, goals, security techniques and existing solutions are explained in detail while highlighting the problems associated with them. Discussion of encryption, ECC, its schemes, the basic primitives and threat analysis are presented in Chapter 3. In Chapter 4, the proposed security scheme and its implementation are discussed in detail. Chapter 5 covers the design and development of testbed used for testing and running simulations while Chapter 6 covers testbed configuration for experimentation, evaluation parameters and also discusses factors that affect performance. The results of experiment to compare ECC and RSA are given in Chapter 6. The experimental results of the proposed scheme and their comparison with the existing scheme are discussed in Chapter 7. In Chapter 8, concluding remarks and direction for future work are presented.

Chapter 2

Ad hoc Security and Existing Solutions

In this chapter, an overview of the ad hoc network security, security threats and security attacks are provided. Security techniques and available solution which could be used to provide security environment to all network nodes are also discussed briefly. Discussion on various security threats that are propagating through the network, the security features that are required, various techniques that are being used to implement those features and how they can be used to improve the secure transmission of the data from the sender to the receiver node, are presented.

2.1. Security - Overview

Security has various definitions based on the requirements of the user. In this work, security is defined as the reliable transmission of information across an insecure network link while the routing protocol determines the route needed to direct packets between the various devices in the mobile ad hoc network [VIN02, WU05, PFL97]. Routing packets securely from one node to the other is difficult to achieve due to several factors such as network environment, number of nodes, information transmission capacity of each node etc. Security performance evaluation can enhance the reliability and dependability of the network by detecting, slowly isolating and removing attacker nodes from the network before it interferes with network performance [SUB99, VIN02].

2.2. Security Threats

Wireless nodes are prone to various active and passive attacks. Securely routing messages from one node to another is difficult to achieve. This is due to the non-cooperative behaviour of network nodes and several other attacks to destroy the network, such as eavesdropping [ZHO03, HUB01], which is done by the nodes in the network as the message transmits through air. This intentional non-cooperative behaviour is mainly caused by two types of nodes:

- **Selfish Nodes:** act greedily by saving battery life and power for their own communication rather than cooperating and distributing data for other nodes [WU05]. They do not want to damage the network or any other nodes directly. However, some nodes do not forward messages for other destination nodes while some other nodes never execute the packet forwarding phase or use energy thresholds to determine its execution policy.
- **Malicious Nodes:** act in order to harm other nodes and disrupt network by causing network to fail. They do not aim at saving battery life for themselves [WU05].

2.3. Security Attacks

Ad hoc attacks [WU05, DES07] can be generally classified as:

- **Passive Attacks:** in which attackers try to collect valuable information extracted by listening to traffic but they do not interrupt the procedure of the protocol.
- **Active Attacks:** in which attackers draw all packets, corrupt them and then retransmit them to disrupt the operation of the network. This helps them to achieve authentication, attract more packets and also hinder availability.

- **Modification Attacks:** in which numeric value changes or alterations in the information of the network are made.
- **Fabrication of Information Attack:** in which false route error messages, disturbing routing state or route table overflow are done.
- **Insider Attacks:** in which the node within the network makes attacks. Route disruption attacks disrupt existing route or prevents new route formation; Route invasion attacks disrupts network communication channel and masquerading attacks violate privacy by masquerading as other nodes and gain sensitive information of the network.

2.4. Security Goals

Achievement of security goals facilitates network nodes to send sensitive information to another node. These goals help network nodes not only to prioritize responsibilities according to the level of security required but also to prevent, detect and respond to security attacks. It also helps to identify messages to a certain degree. Conventional goals [ZHO99] are discussed below:

- **Confidentiality:** ensures that certain classified information is not disclosed to any unintended or unauthorized entities.
- **Availability:** ensures that all the parties get intended network services they require at anytime.
- **Integrity:** ensures that the message remains unaltered and uncorrupted during transmission.
- **Authentication:** ensures the communicating parties are genuine.
- **Non-repudiation:** ensures that the parties can prove the transmission or reception of information by another party i.e. the party cannot misleadingly deny having received or sent certain data.

- **Dependability and Reliability:** ensures message delivery even in the case of node failure or power failure which could affect message delivery actions that impact security. Their performance is measured as failure rate or mean time between failures.
- **Accountability:** ensures that all the security activities are documented as logs and protected which allows users to realize that they are held responsible for all their actions. This also helps to allow appropriate reaction against attacks.

2.5. Security Techniques

Data security can be defined as reliability of information and privacy, lack of unauthorized disclosure of information and data availability for authorized actions. It is a very active and well established area [WU05]. In this work, user and data security are both counted as a part of the service. All these security measures provide security but due to the overhead might also increase system complexity and can sometimes result in system failure. Some approaches [KEM02, WU05] to improve these network conditions are suggested below:

- **Prevention:** bars malicious attackers are barred from actively initiating attacks. It requires encryption techniques like using symmetric/asymmetric cryptography or by using mesh chains.
- **Intrusion Detection:** (IDS) involves the detection of intrusion and recognising proof of intrusion either while they are in progress or after the effect. For accurate intrusion detection, reliable and complete data about the target system's activities is collected which is crucial, expensive and important. Therefore, IDS is required to be local and independent so that neighbouring nodes can form association and collaboratively investigate in a broader range. The response of IDS varies according to the type of intrusion, type of network protocol and the confidence in the veracity of the trace data. Its response may

vary from generating an alert, sounding a siren or even mounting a counter attack.

- **Anomaly Detection:** aims to detect abnormal behaviour using models of the intended behaviour of uses and applications interpreting deviations from this normal behaviour as a problem. These systems, however, are not very reliable in highly dynamic environments.
- **Misuse Detection:** serves as the fundamental definitions of attacks and looks for evidence of known attacks in the data stream. This technique is good when signature behaviour for attack have been defined.

2.6. Existing Solutions

This section contains a brief description of the schemes currently used. Most of the existing security schemes depend on certificates but this trend is changing since key based encryption/decryption of data is easy and as reliable as certificates. A brief comparison of the two schemes is given below:

2.6.1. Certificates vs. Keys

Traditional public key cryptography (PKC) uses certificates issued by certification authority to bind users with public keys. KMS is the best alternative to certificates since they suffer from following drawbacks [KUM05]:

- Usage of certificates requires sending messages with signatures which uses more memory and strains the bandwidth.
- Receiver nodes need to validate certificates and verify signatures which are time consuming and affect the performance of the system.
- Certificate management is tricky and also cumbersome.
- A queue of certificates being issued must be verified by the verifier which is a problem and is time consuming.

2.6.2. Key Management System

Cryptography is based on keys for its functioning. The encrypted information can be revealed if the key is discovered. Managing the keys used in the system is called KMS (Figure 2-1) [ISL05] which includes several algorithms related to the functionality of keys. In this work, the emphasis is mainly on key generation, key agreement and key encryption/decryption. It constitutes of three protocols described below:

- **Key Distribution:** protocol requires an authority to create or otherwise obtain secret values and securely distribute to other node in the network.
- **Key Agreement:** requires each node to establish a common session and a shared secret key without a trusted third party involvement using successive pair-wise message exchange. They are fully distributed and self organized.
- **Key Pre-distribution:** requires the existence of an offline trusted third party who pre-initializes each node in a set with some secret information like a set of long lived keys.

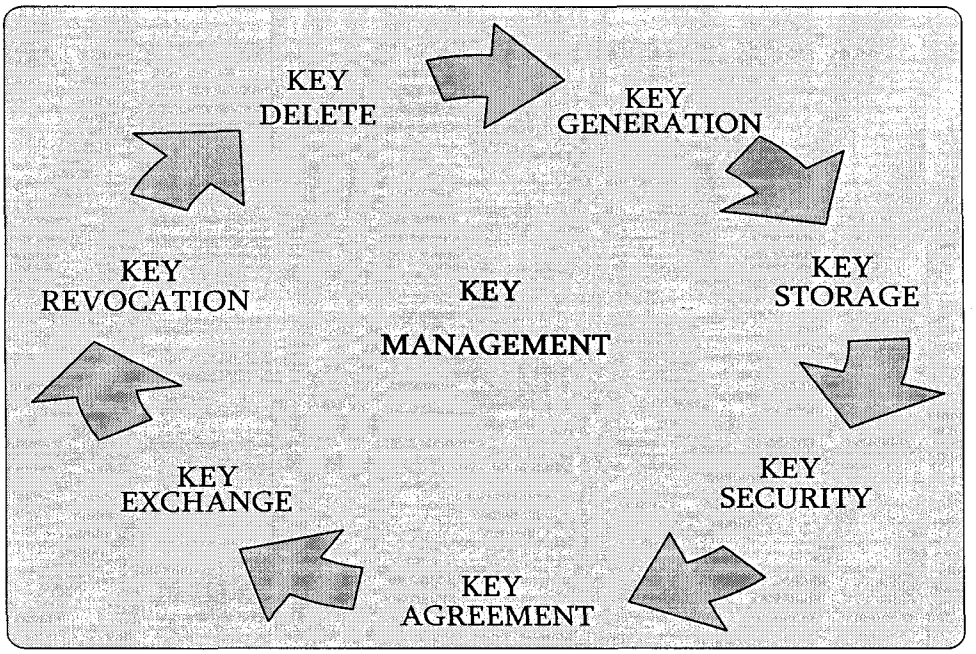

Figure 2-1: Key Management System

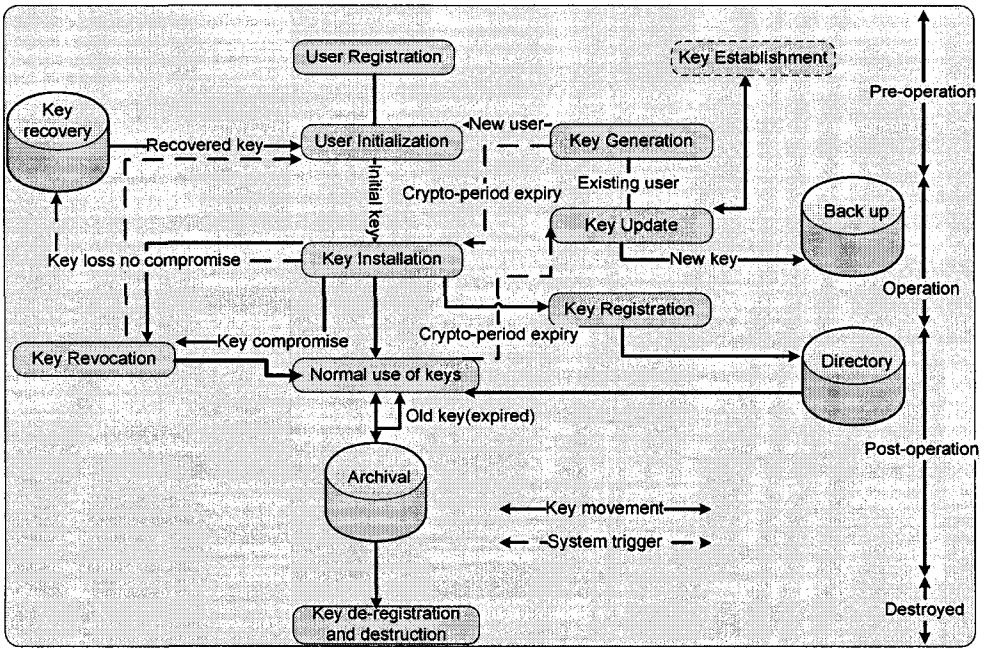

Figure 2-2: Key Lifetime

KMS involves keys at different stages and with each stage, keys have different functionalities. Therefore, the lifetime of a key (Figure 2-2) [BAR07] is categorized in four different phases which are described below:

- **Pre-operational phase:** The keying material is not yet available for normal cryptographic operations. Keys are yet to be generated or may be are in the pre-activation state.
- **Operational phase:** The keying material is available and in normal use. Keys are in the active state.
- **Post-operational phase:** The keying material is no longer in normal use but access to the keying material is possible. Keys are in the deactivated or compromised states.
- **Destroyed phase:** The keying material is destroyed and all records of their existence may have been deleted while the key attributes (e.g. key name, type) may be retained. Keys are in the destroyed or compromised states.

2.6.3. Encryption Schemes

Encryption system (Figure 2-3) [MEN96] generally involves two parties trying to send confidential data over insecure channel and the receiver must be able to retrieve the confidential message. For such schemes, procedure would include generation of key, encryption operation and decryption operation.

The sender and the receiver should first setup the initial parameters and generate individual key pairs. The sender node should obtain the receiver's public key to control the encryption procedure while the receiver's private key is used to control the decryption operation. Then each time the sender wants to send a message to the receiver, it should apply the encryption operation under the receiver's public key to compute ciphertext. Upon receiving ciphertext, the receiver should apply the decryption operation under its key pair to recover the message.

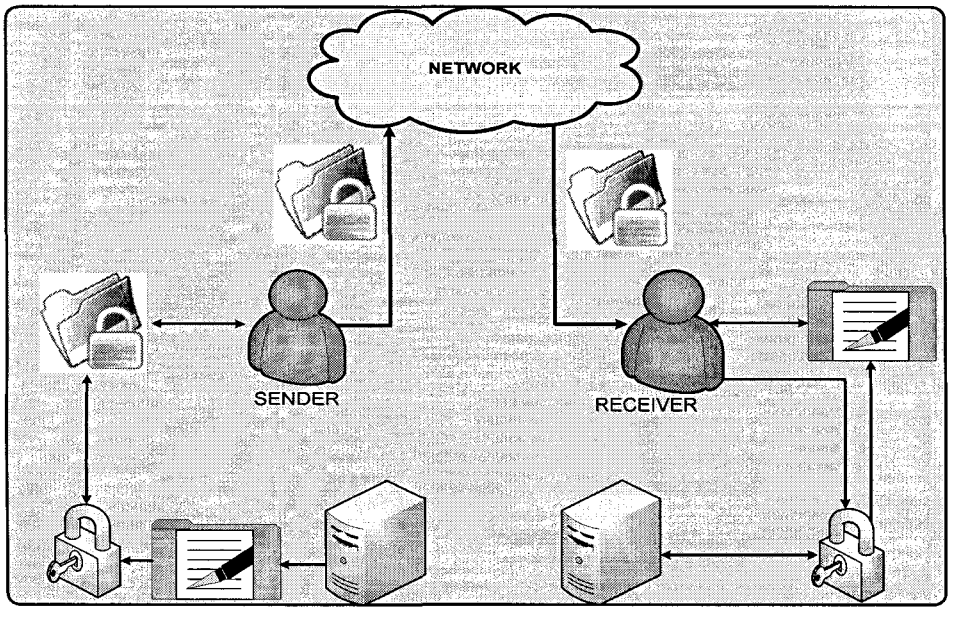

Figure 2-3: General Encryption Overview

Such encryption schemes are deliberately designed in such a way that it should be hard for any evil node to recover the message from cipher text if it does not possess

receiver's private key. This scheme provides data confidentiality while encrypting messages and transporting them from receiver to sender. This elasticity makes the scheme applicable to a broad range of cryptographic systems. This is the traditional usage for PKC schemes. There are several encryption schemes developed specifically for the ad hoc networks due to their specialized needs. Some of those schemes are described below:

2.6.3.1. Identity Based Encryption (IDE) Scheme

IDE cryptosystem [SHA84], introduced by Adi Shamir in 1984, overcomes the overhead and privacy problems by avoiding the use of certificates. It uses user's identity as his public key. The private keys of the users are issued by a private key generator (PKG) through a secure channel after verifying the user's credential. The scheme first establishes a mutually acceptable set of security parameters at the formation of the network. A sender can then encrypt a message for a recipient with identity 'ID' using only the master public key and string 'ID'. In particular, the recipient ID does not have to establish his own public key and propagate it throughout the network. In order to decrypt the message encrypted under a particular identity ID, the user ID must obtain the secret key corresponding to his identity from the PKG. Only the PKG can generate private keys for various identities. There are several encryption schemes available besides various modified versions of Shamir's approach developed by cryptographers.

2.6.3.2. Elliptic Curve (ECC) Encryption Scheme

ECC [WAS03], introduced by Neal Koblitz and Victor S. Miller in 1985, is a public key cryptosystem (PKC) designed to use the algebraic structures of elliptic curves over finite fields and thus overcomes problems introduced by the use of certificates. Public key cryptographic schemes are based on the creation of mathematical puzzles that are

difficult to solve without certain knowledge about how they were created. As the name suggests, elliptic curve is based on an equation which involve real or complex numbers which are really difficult to solve and are known ad discrete logarithm problems. The difficulty of solving this equation determines the security level of elliptic curve. ECC could be used as a key agreement scheme, encryption scheme or as a signature scheme. Some variants of ECC [HAN04, CER00] are given below:

- The ECDH is based on the D-H scheme,
- The Elliptic Curve Digital Signature Algorithm (ECDSA) is based on the Digital Signature Algorithm (DSA),
- The Elliptic Curve Menezes-Qu-Vanstone (ECMQV) key agreement scheme is based on the MQV key agreement scheme.

2.6.3.3. Threshold Based Encryption Scheme

Threshold cryptography [ZHO99] is designed to protect information or computation by fault-tolerantly distributing it among a cluster of cooperating computers. The fundamental problem of threshold cryptography is secure sharing of a secret which allows one to distribute a piece of secret information among several servers in a way that meets the following requirements: 1) no group of corrupt servers (smaller than a given threshold) can figure out what the secret is, even if they cooperate and 2) when it becomes necessary that the secret information be reconstructed, a large enough number of servers (a number larger than the above threshold) can always do it. Another problem of secret sharing is function sharing in which a highly sensitive operation, such as decryption or signing, can be performed by a group of cooperating servers in such a way that no minority of servers are able to perform this operation by themselves nor would they be able to prevent the other servers from performing the operation when it is required.

2.6.3.4. Elgamal Encryption Scheme

The Elgamal encryption scheme [MEN96], developed by Taher Elgamal in 1984, is asymmetric PKC scheme based on the Diffie-Hellman Key Agreement Scheme (DHKAS). It can be used for both digital signatures and encryption and its security results from the difficulty of calculating discrete logarithms in a finite field. It usually works in a multiplicative group of *GF(p)* and in this case, the progress in the discrete logarithm problem forces the users of such a basic Elgamal PKC to permanently increase a prime mod *p* in order to ensure the desired security. However, the task of finding a multiplicative group of *GF(p)* is unfeasible for an ordinary user.

2.6.3.5. Rivest, Shamir and Adleman (RSA) Encryption Scheme

RSA [DIF76, MEN96], developed by Ron Rivest, Adi Shamir and Leonard Adleman at MIT in 1977, is the first asymmetric PKC scheme which had the capability not only for signing but also for encryption. Being the major advancement in public key cryptography, it is among the most widely used algorithms. It involves usage of two different types of key, public key and private key. Public keys are open to public viewing whereas private keys are kept secret. Messages are encrypted using the public key and then decrypted at the receiver's end using the private key. It provides security with sufficiently long key and is a credible algorithm used in most commercial protocols. The security of RSA, believed to be lower than other symmetric algorithms, is based on two mathematical problems, namely, factoring large number problem and RSA problem. RSA is a compatible standard for several applications.

2.6.4. Ad Hoc Routing and Secure Routing Techniques

The basic idea of routing protocol is to inform all the other nodes close by about its presence in the network and then try transmitting messages. Based on the threat

assessment and the potential capabilities of attacks, several secure versions of routing protocols have been developed such as Authenticated routing for ad hoc network (ARAN), Secure routing protocol (SRP), Secure efficient ad hoc distance vector routing (SEAD), Ariadne, Secure AODV (SAODV), Secure link state routing protocol (SLSR). There are various types of existing routing protocol and some of them used in the implementation of the scheme are described below:

2.6.4.1. Dynamic Source Routing (DSR)

DSR [JOH94] is an on-demand source routing based protocol for the ad hoc network. DSR depends on forwarding packets to only the known path and accumulating these paths is a difficult job. Since this protocol is beacon-less, it does not require periodic hello packet (beacon) transmissions, which are used by a node to inform its neighbors of its presence. This protocol is designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach. The intermediate nodes also utilize the route cache information efficiently to reduce the control overhead. The disadvantage of this protocol is that the route maintenance mechanism does not locally repair a broken link.

2.6.4.2. Destination Sequenced Distance Vector (DSDV) Routing

DSDV [PER94] is a table-driven ad hoc routing scheme based on the Bellman-Ford algorithm that was developed by C. Perkins and P. Bhagwat in 1994. This scheme resolves the routing problem. It is quite suitable for creating the ad hoc networks with small number of nodes. Routing messages are exchanged between neighbouring nodes and these updates may be triggered or routine. These updates also consume battery power and a small amount of bandwidth even when the network is idle. DSDV is not suitable for highly dynamic networks.

2.6.4.3. Ad Hoc On-Demand Distance Vector (AODV) Routing

AODV routing protocol [PER99], based on DSDV and DSR, is capable of both unicast and multicast routing. It is a on-demand protocol that generates routes between source and destination only when there is a requirement and would maintain those routes as long as they are needed by the sources. Additionally, AODV is loop-free, self-initiating and scalable to large dense network. It uses periodic beaconing and sequence numbering procedure of DSDV while it differs from DSR in two main properties:

- In DSR, each packet carries full routing information whereas AODV carries only the destination address.
- In DSR, route replies carry the address of each node along the route whereas in AODV route replies carry only the destination IP address and the sequence number.

2.6.4.4. Secure Ad Hoc On-Demand Distance Vector (SAODV) Routing

Since AODV is vulnerable to attacks, SAODV [ZAP02], a secure version, was developed to prevent them. SAODV, an extension of the AODV routing protocol, is a public key infrastructure (PKI) based approach that relies on RSA signatures and hash chains. The main objective of SAODV is integrity, authentication and non-repudiation of AODV routing information. This scheme assumes that each node has certified public keys of all network nodes so that intermediate nodes can validate all in transit routing packets. As the message traverses the network, the intermediate nodes cryptographically validate the signature and the hash value. Since AODV and SAODV are similar, they have the same problems. In fact, with SAODV the problems are more severe. SAODV increases the packet overhead with increasing mobility due

to the extra processing time. This may cause Denial of Service (DoS) of low computational resource nodes even in the absence of malicious nodes.

2.6.4.5. Optimized Link State Routing (OLSR)

OLSR [CLA03a] is a point-to-point routing protocol based on the traditional link state algorithm. Each node maintains topology information about the network and updates this information by periodically exchanging link state messages. It minimizes the size of the control messages and the number of rebroadcasting nodes during each route update by employing multipoint replaying strategy.

2.6.4.6. Secure Optimized Link State Routing (SOLSR)

SOLSR [CLA03b] is an extension of OLSR which uses digital signatures for improving the security in routing messages and provides integrity by securing only the routing messages and not the user traffic. This solution is extendable and can be modified to include confidentiality. The purpose is to provide authentication to OLSR signalling packets and to prevent replay attacks. This is achieved by using Hashed Message Authentication Code (HMAC) in every hop and timestamp exchange. All traffic is signed by every hop node which ensures that nodes trust the source of the message in the previous hop. A shared secret key is used for signature creation and verification which is accessible to all intended nodes in the network. A node that does not have access to the shared secret key cannot produce a verifiable digest. This approach adds more packet overhead to OLSR protocol with higher mobility networks. This overhead is relative to the size of the OLSR packets since the signature size is static. With larger OLSR packets, the secure plug-in will have less effect on overhead.

2.6.4.7. Location Aided Routing (LAR)

LAR [KO98], as the name suggests, is based on the location of the destination node. It uses flooding to reduce the routing overhead present in the traditional flooding algorithm. However, it is mandatory that every node in the network must have and provide the location information of every other node. Nodes can use Global Positioning System (GPS) to collect location information. It could even define a boundary for the traversal of the route request packets to the required destination. Route formation could also be based on the relative distance to the destination. The protocol tries to conserve bandwidth and may behave similar to flooding in a highly dense network.

Chapter 3

Elliptic Curve Ad Hoc Security Scheme

In this chapter, ECC security scheme essentials which support secure message transmission between two nodes in the network are described. Some encryption basics, ECC basics and the scheme which use ECC as the fundamental technique are described here. Algorithms containing primitives of ECC which are used in implementation are also provided in this chapter.

3.1. Encryption-Overview

Encryption is used to prevent attacks on confidential data. Decisions about what information should be encrypted and placement of the encryption function must be made. Attacks could be made on end devices or on the link between them [STA02].

- **Link Encryption:** Although the user has no control over the links between end points in a public network, link encryption [STA02] could be used to prevent attacks on the communication links. This link encryption allows user to generate a secret shared key generated from the individual user data portions. Any attacker or intermediate node will route the packet after reading the header which is not encrypted but if it tries to decrypt the packet, it will need user data portions which are not available publicly. Thus, it could be concluded that user data is secure over any communication link but the traffic headers are not, since packet headers are transmitted without encryption. The

second potential location of an attacker is the end devices. End devices could be made secure by using end-to-end encryption which also provides some degree of authentication to prevent attacks on packet header. Packet header is important because it contains sensitive information about the end devices and the routing information which could be used against the network. Such authentication is not provided by link encryption.

- **End-to-End Encryption:** End-to-end encryption [STA02] stands for secure, confidential and reliable transmission of information which could be encrypted at source and decrypted at the destination while maintaining data integrity. It also provides a degree of authentication and works with two end systems: source and destination. The encrypted message is transferred unaltered over the network to the destination end system. Since the destination system shares a key with the sender node, destination node is able to decrypt the message. This scheme ensures security against the network link attacks but since the packet header is not encrypted, there are still some possibilities for threats.

To ensure the safety of the network and network links, both link encryption and end-to-end encryption must be used together [STA02]. When both encryptions techniques are used together, the sender can encrypt the data portion of the packet using the end-to-end encryption key. This entire packet is then encrypted using the link encryption key. If end-to-end encryption is used with link encryption, encryption function must be performed at the low level of the communication hierarchy and link encryption should be performed at high level of the communication hierarchy. Link encryption can occur at either the physical layer or the link layer while several choices exist for end-to-end encryption. End-to-end encryption is based on encryption option and has several options for placing this encryption function such as on the network layer or the transport layer. When

encryption function is implemented on the network layer, users use encrypted messages for communication and also share a secret key. All the users will also employ similar processes and applications to provide encryption/decryption capabilities. Such network layer framework provides end-to-end security for all traffic within the network.

Another important aspect that should be considered is the traffic confidentiality which is defined as the protection provided to the traffic carrying sensitive information such as identities of the nodes, frequency of the node's communication, message length, message pattern and also the keys. When link encryption is used, network header is also encrypted which provides security against traffic analysis. This property also benefits proposed scheme. The next step is to analyze the techniques used in order to develop this scheme.

3.2. ECC – Overview

The proposed security scheme is based on ECC [CER00, KOB00] which is designed to provide a variety of security goals. ECC is fundamentally more difficult to explain than RSA or any other scheme. ECC systems are based on elliptic curve which could be defined in the form of a cubic equation $y^2 = x^3 + ax + b$. ECC can make use of these elliptic curve in which the variables and coefficients are all restricted to the elements in finite field. Prime curves are best for software implementation since they provide bit fiddling operations. Cyclic group for any point G is defined as $(O, G, G+G, G+G+G, G+G+G+G)$. Scalar multiplication for any integer k is defined as $k * G$. Multiplication group of powers in integer g mod prime p is defined by $(g^0, g, g^2, g^3, g^4, ...)$. Since cyclic groups are similar to multiplicative groups, the problem of finding k given points kG and G is called the elliptic curve discrete logarithm problem (ECDLP). ECC is based on binary operators and require

conversion of bit-string to a point in a curve. ECC curves are based on certain group laws using binary operation (denoted by $\bullet$):

- Closure- If $a, b \in G$ then $a \bullet b \in G$.
- Associative- $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all a, b and c in G.
- Identity element- $a \bullet e = e \bullet a = a$ hold true for an element e and all a in G.
- Inverse element- $a \bullet a' = a' \bullet a = e$ hold true for an element a' and all a in G.
- Commutative- $a \bullet b = b \bullet a$ for all a and b in G.

Several schemes have been developed based on ECC and some of them are described below:

3.2.1. Key Agreement and Exchange Protocols

PKC keys have two main problems: 1) key escrow and 2) insecure communication channel. Key agreement protocol [WIL97] is a good solution for both the problems since it allows two parties to generate a secure communication session and a secure session key which is used for further correspondence. This protocol is secure since it hides sensitive information from any eavesdropping user [RUE94]. Some of the protocols are described below:

3.2.1.1. Diffie-Hellman Key Agreement Scheme (DHKAS)

DHKAS [RUE94] (Figure 3-1) was first developed in 1976 by Diffie and Hellman. It provides a perfect forward secrecy allowing two parties with no prior knowledge of each other to jointly establish a shared secret key over an insecure communication channel. This agreement could serve as the basis for several authentication protocols. Two users generate and exchange two global public parameters prime number q and primitive root of q α. Parameter q is a prime number and parameter α (usually called a generator) is an integer less than q, with the following property: for every

number n between *1* and $q-1$ inclusive, there is a power k of g such that $n = g^k \bmod p$. This value is then exchanged to derive the session secret key.

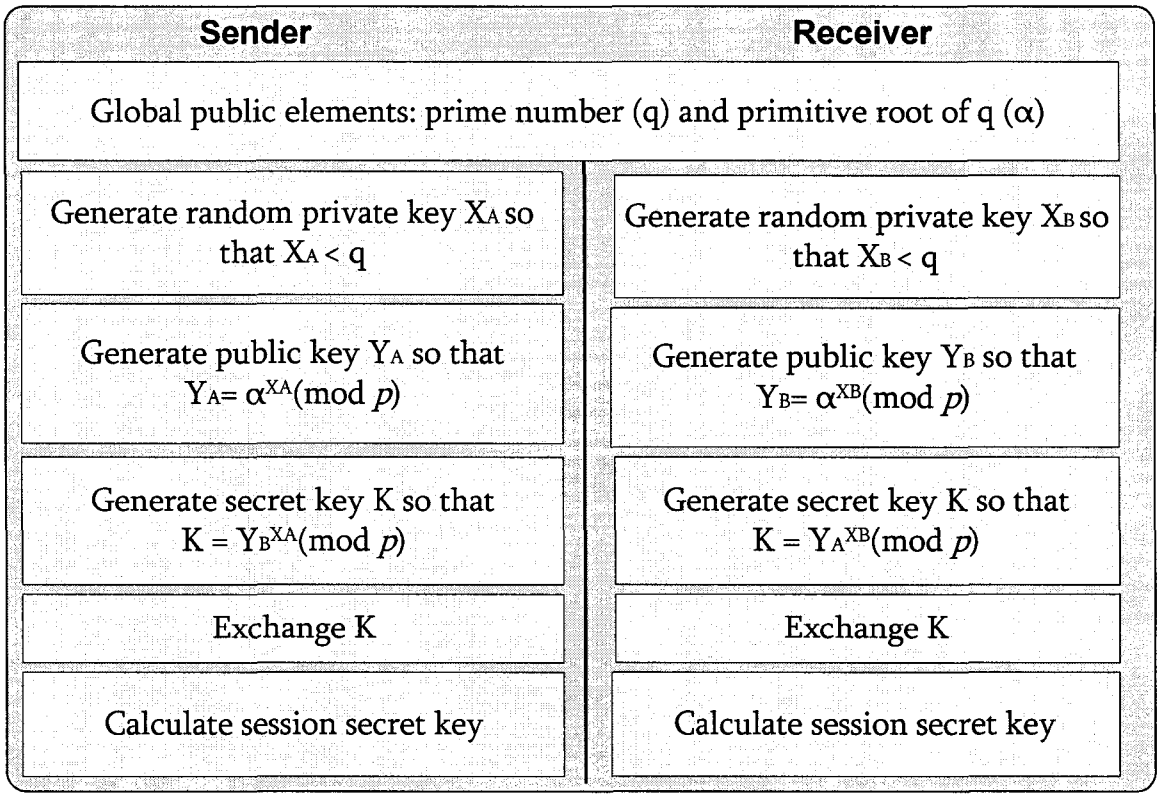

Figure 3-1: Diffie-Hellman Key Agreement Protocol

3.2.1.2. Elliptic Curve Diffie-Hellman Key Agreement Scheme (ECDH-KAS)

ECDH-KAS [CER00] (Figure 3-2) is a key agreement asymmetric cryptographic protocol used to generate a secure shared session key which could be used for communication over an insecure channel. ECDH uses D-H protocol over elliptic curve groups and is secure since any other node, which does not have access to the private information of the devices involved, cannot calculate the shared secret from the publicly available information. Another benefit of using this protocol is that it provides perfect secrecy using smaller key lengths. It is standardized in ANSI X9.63 [ANI99], IEEEP1363 [IEE00] and ISO 15946-3 [ISO98]. Apart from the ECDH, ECIES is also used to provide the basic encryption/decryption operations.

Two users, each with a key pair, generate a private key X (a randomly selected integer less than n, where n is the order of the curve), an elliptic curve domain parameter (ECDP) and a public key $Y = X*G$ (G is the generator point, an elliptic curve parameter). Let (X_A, Y_A) be the private-public key pair of sender and (X_B, Y_B) be the private-public key pair of receiver. With these key pairs, a new secret key is generated using the public key of the other interacting node and then exchanged. The receiver node also does the same calculations and sends it to the sender in the network. Both calculations yield determines the same output and that point is selected as the shared secret session key.

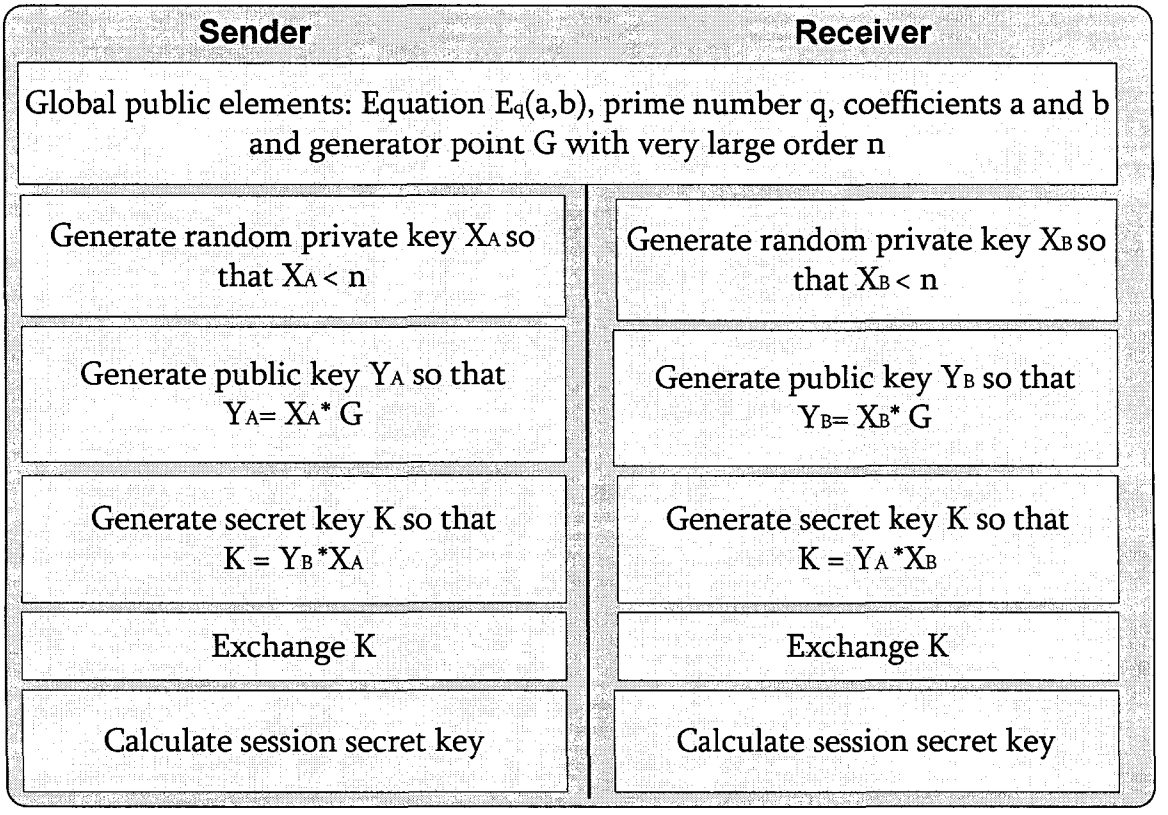

Figure 3-2: Elliptic Curve Diffie-Hellman Key Agreement Protocol

3.2.2. Elliptic Curve Integrated Encryption Scheme (ECIES)

ECIES (Figure 3-3), initially proposed by Neal Koblitz and Victor S. Miller in 1985, is a PKC scheme based on ECC [CER00, KOB87]. It is, also known as Elliptic Curve Augmented Encryption Scheme (ECAES) or Elliptic Curve Encryption Scheme, a combination of ephemeral-static DHKAS and a technique for symmetrically encrypting plaintext data. In the presence of an adversary, this scheme is semantically secure, plaintext-aware and secure even when the adversary is capable of launching chosen-plaintext, chosen-cipher text attacks and eavesdropping. It provides integrity over the resulting ciphertext and was proven to be secure under specific assumptions.

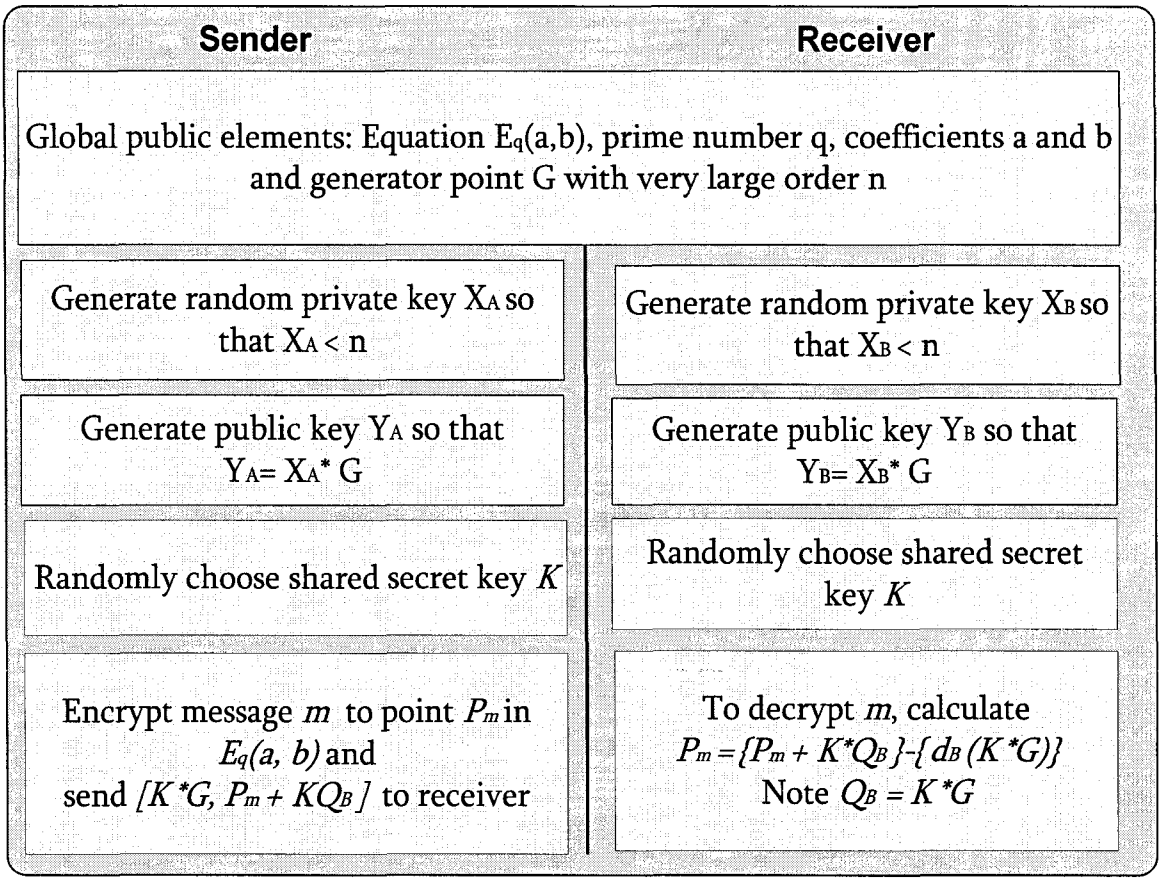

Figure 3-3: Elliptic Curve Integrated Encryption Scheme

This scheme owes its working to the development of the initial domain parameters (DP) [ISO08] which specify the use of a prime order group generated by the base

point *G*. The implementation requires both nodes to understand and agree on the initial setting for further correspondence. The main calculations used by ECIES while encrypting is a two point scalar multiplication and while decrypting is a single point scalar multiplication. ECIES is also being standardized in ANSI X9.63 [ANS99]. It is proven secure based on a variant of the Diffie-Hellman problem (DHP) in [BAO03].

Table 3-1: Public Key Sizes (bits) Comparison (Source: [CER00])

AES key size	ECC key size	RSA key size	Ratio
80	163	1024	1:6
128	256	3072	1:12
192	384	7680	1:20
256	512	15360	1:30

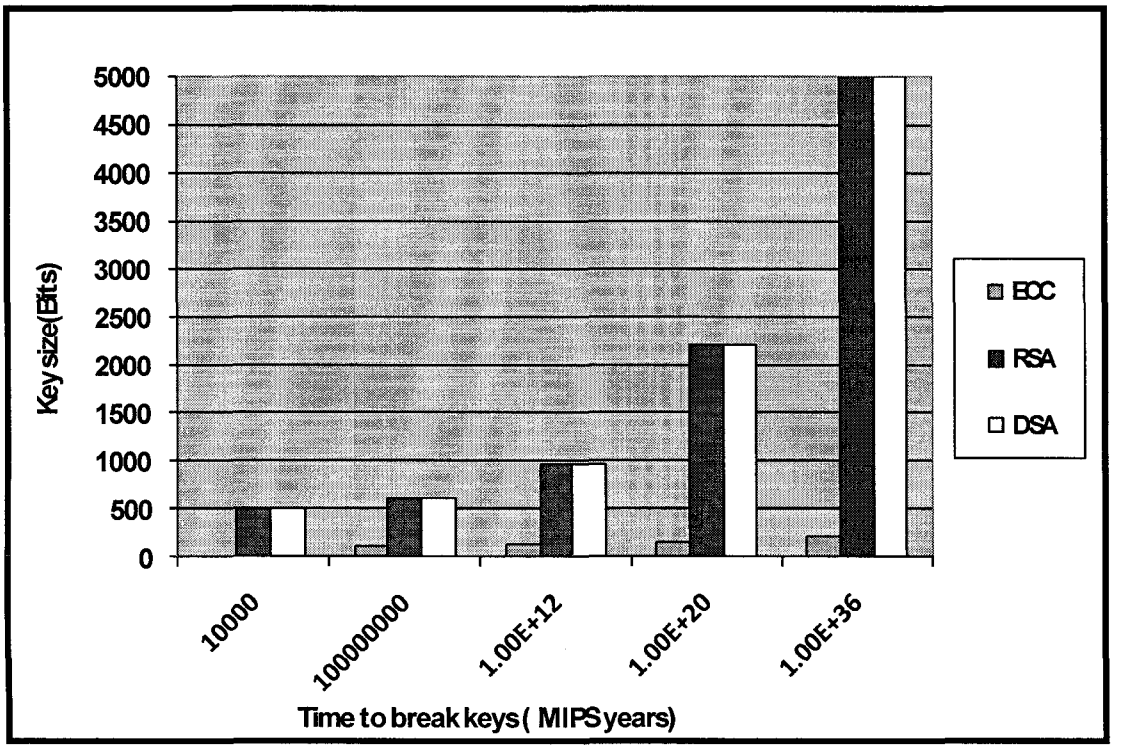

Figure 3-4: ECC Security Evaluation against RSA (Source: [CER00])

ECIES is chosen for this research since it offers an attractive mix of provable security and efficiency. Its performance shows that under similar circumstances, ECIES

performs better than or as good as other schemes [LOP00]. ECC is better than RSA due to shorter key lengths [CER00] of ECC, reduced overhead and less bandwidth requirement due to shorter keys (Table 3-1). It also ensures confidentiality, mutual authentication, timing modification and content modification even while encrypting long messages.

ECC is more efficient than RSA due to the lack of additional code that is required by RSA to provide additional security [CER00] and error checking (Figure 3-4). ECC scales well to the limited space, memory and processing power of small devices. It does not require any additional crypto processors in order to function more efficiently. With ease of use, interoperability, flexibility, association to standards, market value and return of investment, ECC becomes an advantageous choice.

3.3. ECC Based Encryption/Decryption Primitives

All the security schemes can be divided into two categories: technique primitives used and usage of those primitives in the implementation of those techniques. Existing methods fail to provide protection against eavesdropping and ECDH scheme offers relief to a certain extent. However, the proposed scheme depends on certain assumptions to solve this problem. ECC is based completely on elliptic curve theory over finite fields. Earlier version of cryptosystems tried to unscramble the message using numerous possible combinations but with the latest advancement in discrete logarithms [ODL95, SMA99], finding solution has become even more difficult, thus affording greater security.

An elliptic curve [WAS03, BLA99] is a plane over an equation E of the form $y^2 = x^3 + ax + b$ or $y^2 + xy = x^3 + ax + b$ with x and y coordinates chosen from two finite fields: fields of odd characteristic (F_p, where $p > 3$ is a large prime number) or

fields of characteristic two (F_2m) respectively [KOB87]. In cryptography, an adversary capable of calculating all discrete logarithms in the group can break the cryptosystem. Discrete logarithmic problem [BLA99] for such elliptic curve systems are much more difficult to resolve than other systems. This complexity and difficulty works for benefit by offering equivalent level of security.

In the following section, the primitive used by ECC is explained. First, the ECDP generation and their validation are demonstrated which every node must agree to. Second, keys are generated, validated and then used to implement ECDH protocol which generates a shared secret key. Using this key ECIES protocol is implemented. Both of these protocols help in secure transmission. The former scheme is used to initiate a session establishment where the man-in-the-middle is unable to interpret the information being sent and the latter is used to encrypt the data. This data encryption takes place at two levels. First, the location of the message is obtained in elliptic curve and then this message is encrypted using the shared session key of the receiver node. Once the message is received it can easily be decrypted by the private key of the receiver node and message can be retrieved from point location in the elliptic curve. This two-way secure system allows secrecy of the message even with the presence of an intermediate node or man-in-the-middle.

3.3.1. Elliptic Curve Domain Parameter Generation

This primitive provides the freedom to use any of the known curve selection methods. It is strongly recommended to use one of the ECDP over F_q to promote interoperability. ECDP over F_q should be generated as follows:

Algorithm *Domparamgen* (Source: [HAN04])

Input: A field order q; a field representation FR for F_q; security level L satisfying $160 \le L \le \lceil \log_2 q \rceil$ and $2^L \ge 4\sqrt{q}$.

Output: Elliptic curve domain parameters D $(q, FR, S, a, b, P, n, h)$ where q is the field order, *FR* is the field representation for F_q, S is the seed, (a,b) are the coefficients of the equation $E(F_q)$, P is the point in $E(F_q)$, n is the order of P and h is the cofactor.

1. *Select* $a, b \in F_q$ *verifiably at random.*
2. *Compute elliptic curve order* $N = \#E(F_q)$.
3. *Verify that* N *is divisible by a large prime* n *satisfying* $n \rangle 2^L$. *Else, go to step 1.*
4. *Verify that* n *does not divide* $q^k - 1$ *for* $1 \le k \le 20$. *If not then go to step 1.*
5. *Verify that* $n \ne q$. *If not then go to step 1.*
6. *Set* $h \leftarrow N/n$.
7. *Select an arbitrary point* $P' \in E(F_q)$ *and set* $P = hP'$. *Repeat until* $P \ne \infty$.
8. *Return* $(q, FR, S, a, b, P, n, h)$.

3.3.2. Elliptic Curve Domain Parameters over F_q Validation Primitive

ECDP over F_q validation primitive should be used to check the validity of ECDP over F_q. This is necessary because all the nodes who are communicating needs to do calculations on the same value and on the same network. Any disagreement in these values could result in a disaster since every node will be doing calculation based on different value and none of them will be able to get their messages across. There are various steps to check the validity however, any of these steps could be followed in order to validate the DP. The procedure for checking validation is as follows:

Algorithm *Domparamval* (Source: [HAN04])

Input: Elliptic curve domain parameters D $(q, FR, S, a, b, P, n, h)$

Output: Acceptance or rejection of the validity of D.

For each node in the network N_w:

1. *Verify that q is a prime power ($q = p^m$ where p is prime and $m \geq 1$).*
2. *If $p = 2$ then verify that m is prime.*
3. *Verify that FR is valid field representation.*
4. *Verify that a, b, xp, yp (where $p = (x, y)$) are elements of F_q.*
5. *Verify that a and b define an elliptic curve over F_q.*
6. *If the elliptic curve was randomly generated then verify that s is a bit string of length at least l bits where l is the bit length of hash function H.*
7. *Verify that $P \neq \infty$.*
8. *Verify that P satisfies that elliptic curve equation defined by a, b.*
9. *Verify that n is prime, that $n > 2^{160}$ and that $n > 4\sqrt{q}$.*
10. *Verify that $nP = \infty$.*
11. *Compute $h' = \left\lfloor (\sqrt{q} + 1)^2 / n \right\rfloor$ and verify that $h = h'$.*
12. *Verify that $n \neq q$.*
13. *If verification fails then return ('Invalid'); else return ('Valid').*

3.3.3. Elliptic Curve Key Pair Generation

Given ECDP D $(q, FR, S, a, b, P, n, h)$, an elliptic curve key pair (Q, d) associated with *DP* consists of an elliptic curve secret key d which is an integer in the interval [1, $n-1$] and an elliptic curve public key Q (x_Q, y_Q) which is the point $Q = d * G$. Elliptic curve key pairs should be generated as follows:

Algorithm *Keygen* (Source: [HAN04])

Input: Elliptic curve domain parameters D $(q, FR, S, a, b, P, n, h)$.

Output: Elliptic curve Public key Q and Private Key d.

For each node in a network N_w:

1. *Select $d \in_R [, n-1]$.*
2. *Compute $Q = dp$.*

3. *Return* (Q,d).

3.3.4. Validation of Elliptic Curve Public keys

There are four acceptable methods for the sender to receive an assurance that an elliptic curve public key is valid. Only one of the methods must be applied, although in many cases greater security may be obtained by applying more than one of the methods. The four acceptable methods are:

1. The sender performs validation of the elliptic curve public key itself using the public key validation primitive.
2. The sender generates the elliptic curve public key itself using a trusted system.
3. The sender receives assurance in an authentic manner that a party trusted with respect to the sender's use of the elliptic curve public key has performed validation of the public key using the public key validation primitive.
4. The sender receives assurance in an authentic manner that a party trusted with respect to the sender's use of the elliptic curve public key generated the public key using a trusted system.

Algorithm *PubKeyval* (Source: [HAN04])

Input: Elliptic curve domain parameters D (q,FR,S,a,b,P,n,h); EC Public key Q.

Output: Acceptance or rejection of the validity of Q.

For each node in a network N_w:

1. *Verify that* $Q \neq \infty$.
2. *Verify that* x_Q *and* y_Q *are properly represented elements of* F_q.
3. *Verify that* Q *satisfies the elliptic curve equation defined by* (a,b).
4. *Verify that* $nQ = \infty$.
5. *If any verification fails then return ('Invalid'); else return ('valid').*

3.3.5. ECDH-KAS Primitives

The ECDH scheme is a key agreement scheme based on ECC. It is designed to provide a variety of security goals depending on its application. The goals it can provide include unilateral implicit key authentication, mutual implicit key authentication, known-key security and forward secrecy, depending on issues like whether or not public keys are exchanged in an authentic manner and whether key pairs are ephemeral or static. ECDH primitives [SEC00] can be issued only when the selection has been made from the following options: the standard ECDH primitive or the cofactor ECDH primitive. The basic concept of both primitives is the same, which is to generate a shared secret key from the keys owned by the sender and the receiver so that when both nodes complete the ECDH primitive procedure they simultaneously establish the same shared secret value. However, the main difference between the two primitives is that the standard ECDH primitive is the basic analogue of the well-known DHKAS whereas the cofactor ECDH primitive incorporates efficient resistance to attacks like small subgroup attacks. The sender should utilize the following ECDH algorithm to calculate a shared secret value with receiver:

Algorithm *Keyagree* (Source: [HAN04])

Input: Users' keypair $[(Q_A, d_A);(Q_B, d_B)]$; Elliptic curve domain parameters D $(q, FR, S, a, b, P, n, h)$

Output: Elliptic curve shared secret key Z.

For each two nodes A and B communicating in network N_w:

1. *Compute the elliptic curve point* $P = (x_p, y_p) = d_u Q_v$.
2. *Check that* $P \neq 0$. *If* $P = 0$, *output "invalid" and stop.*
3. *Output* $Z = x_p$ *as the shared secret field element.*

3.3.6. ECIES-Encryption/Decryption Primitives

The ECIES is a public-key encryption scheme based on ECC. It involves usage of symmetric encryption scheme, message authentication scheme and hash function. This combinational encryption scheme acts in two steps; first the encryption of the message using the **Algorithm** *Keyencrypt* at the sender's end and then the decryption by using the **Algorithm** *Keydecrypt* at the receiver's end. The setup procedure for ECIES, the encryption and the decryption operation are explained below:

Algorithm *Keyencrypt* (Source: [HAN04])

Input: Elliptic curve domain parameters $D(q, FR, S, a, b, P, n, h)$; EC Public key Q; plain text m.

Output: Ciphertext $C = (R, c, t)$.

For each node sending information to user B in network N_w:

1. *Select* $K \in (1, n-1)$.
2. *Compute* $R = kP$ *and* $Z = hkQ$ *and verify* $Z \neq \infty$.
3. *If* $Z = \infty$ *then go to step 1.*
4. $(k1, k2) \leftarrow KDF(x_z)$, *where* x_z *is the coordinate of* Z.
5. *Compute* $c = ENC_{k_1}(m)$ *and* $t = MAC_{k_2}(c)$.
6. *Return* (R, c, t).

Algorithm *Keydecrypt* (Source: [HAN04])

Input: Elliptic curve domain parameters D $(q, FR, S, a, b, P, n, h)$; EC Private Key d; Ciphertext $C = (R, c, t)$.

Output: Plain text m or rejection of Ciphertext $C = (R, c, t)$.

For each node receiving information from user A in network N_w:

1. *B validate public key of* R_A. *If fails then return ('Reject Ciphertext').*

2. *Compute* $Z = hkQ$ *and verify* $Z \neq \infty$.
3. *If* $Z = \infty$ *then return ('Reject Ciphertext').*
4. $(k1, k2) \leftarrow KDF(x_z)$ *, where* x_z *is the coordinate of* Z.
5. *Compute* $t' = MAC_{k_2}(c)$. *If* $t \neq t'$ *then return ('Reject Ciphertext').*
6. *Compute* $m = DEC_{k_1}(c)$
7. *Return* m.

Chapter 4

Implementation of Proposed Ad Hoc Security Scheme

In this chapter, a security scheme is proposed using ECC and its primitives. The only way to validate efficiency of this scheme is via implementation which involves a two-level system namely, formation of the session key and encryption/decryption of the message. Security scheme essentials, implementation modules, use of primitives and step-by-step procedure for the experimentation are provided.

4.1. Designing Proposed Scheme

After analyzing the problem related to ad hoc security scheme, the design requirement of the proposed scheme were short-listed. The design of the proposed scheme needs to:

- minimize the dependency, number and complexity of trusted mechanisms involved;
- minimize physical activity and attain more physical security;
- achieve maximum flexibility and versatility in terms of routing protocols and
- achieve maximum robustness.

To minimize dependency and reduce complexity, KMS requires two users to develop confidence in one another. Therefore, ECDH scheme [NIS06], explained in section 3.2.1.2, is used to generate a common session key between communicating nodes by

exchanging some public information. The public information could be any well-known unique fact about the user. For any further correspondence, both parties use session key generated by the ECKAS scheme. This also prevents eavesdropping by an evil user due to the restricted access to the information. Even if the intermediate evil node hacks the message or decides not to forward it to the original recipient, the message is still sealed with a confidential code which could be opened only by the original recipient. Keys are never sent over insecure channel. Every node generates and stores its keys. This design also provides flexibility of using different routing protocols. These goals vary with application; but for any network, the main security concern will always be to achieve authentication, non-repudiation and confidentiality.

Key as the central unit is also susceptible to attacks especially old keys since key could be compromised or disposed after a certain period of usage [MEN96]. Therefore, timely renewal of the keys, depending on the application, is recommended. To prevent these attacks, they could be securely stored on the user system, thus minimizing the use of key in communication. Same approach is utilized for this scheme and therefore, renewal of the keys is no longer required. However, periodic renewal is still recommended for highly sensitive and physically insecure user systems. Key integrity and ownership can be protected from attacks by hashed message authentication code (HMAC) [KRA97]. Message authentication codes (MAC) [BEL96, KRA97] checks the integrity of the information transmitted over or stored in an unreliable medium by using secret key. Typically, these schemes are used to authenticate information transmitted between two parties when these communicating parties have agreed on a secret key. HMAC provides an ultimate integrity check for any information by using cryptographic hash functions along with secret key.

Ad hoc network can use different routing protocols and to avoid misrouting of traffic in the network, each node must be assigned with a unique identification name (ID) and the keys are generated using their ID. Transferring private key over an insecure channel is risky since it is a direct threat to user's system. In this scheme, the private keys are safely stored and are secure with the user itself. In order to check the credibility of the users within the network, key agreement service acts as capable authentication service keeping records of the nodes in the network even when the node was not available during the network initialization phase. The credibility of the user also increases the credibility of the message sent by the sender to the receiver who is trustworthy and appealing.

Packet forwarding becomes easier and more reliable assuming that each node knows its location and the location of its neighbors. Since this technique does not use memory space for saving explicit routes, it scales well for the highly dynamic networks. One of the biggest problems arises due to the privacy issue of nodes' location since it serves as the base of the packet forwarding techniques. Associating individual nodes to their identity seems to be a feasible solution to this problem. Another solution is to use encryption before sending information to the receiver node. However, this introduces another threat known as eavesdropping. Several protocols to handle eavesdropping exist but they fail to avoid the massive control overhead and others do not support the dynamic nature of the network as partitioning and merging situations. Some of them consider standalone ad hoc networks and others consider the connected scenario.

4.2. The Proposed Scheme

The proposed security scheme (Figure 4-1) for MANETs involving a two-level system namely, formation of the session key and encryption/decryption of the message is

proposed. This scheme is especially beneficial because encryption of data using the secret key, that communicating nodes agreed on, is done with minimum overhead both in terms of time and memory. The sender would first execute a key agreement phase using ECDH secret key and then perform message encryption/decryption using ECIES [CER00]. In order to decrypt the message, the receiver does not need to obtain private key from PKG as in the case of IDE cryptosystems [SHA84].

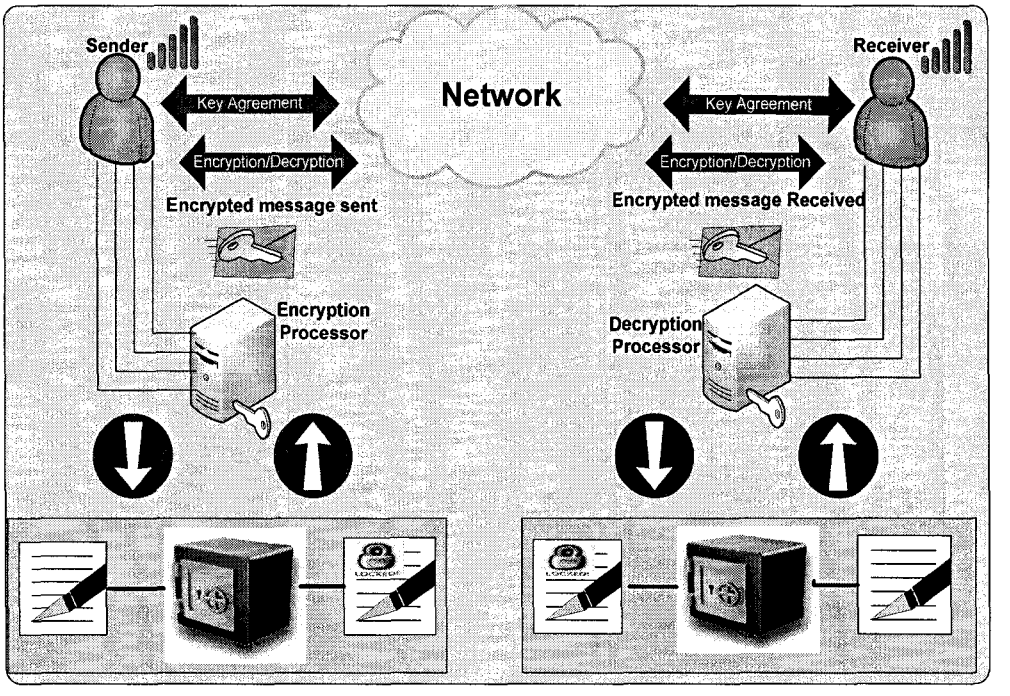

Figure 4-1: Proposed Security Scheme

If a sender wants to send a message to a receiver, they can communicate using their identities and keys generated from those unique identities. This scheme also proposes that during the initial contact between the sender and the receiver, they must agree on a session key using the ECDH scheme. It should be noted that any intermediate node just sees numbers being sent and received between the sender and the receiver and hence, is incapable of interpreting the encrypted message being sent to the receiver node. Once this session key is generated, all the information is encrypted using the shared session key and sent to the intermediate node which will not be able

to decrypt the information since it does not possess the session key and there is no way it can attain the private key of the receiver. This ensures secrecy of the private key and the session key. This scheme ensures secrecy against eavesdropping and is simple to implement.

This scheme can be represented as an envelope carrying encrypted information, receiver node and the encrypted message. The encrypted information cannot be decrypted by the intermediate nodes and the encrypted message can only be read by the receiver. Decryption of the message is not possible since it is encrypted by the session key which the sender and the receiver agreed upon and that session key is not known to anyone else in the network. It is also possible to choose the needed security level for a particular application.

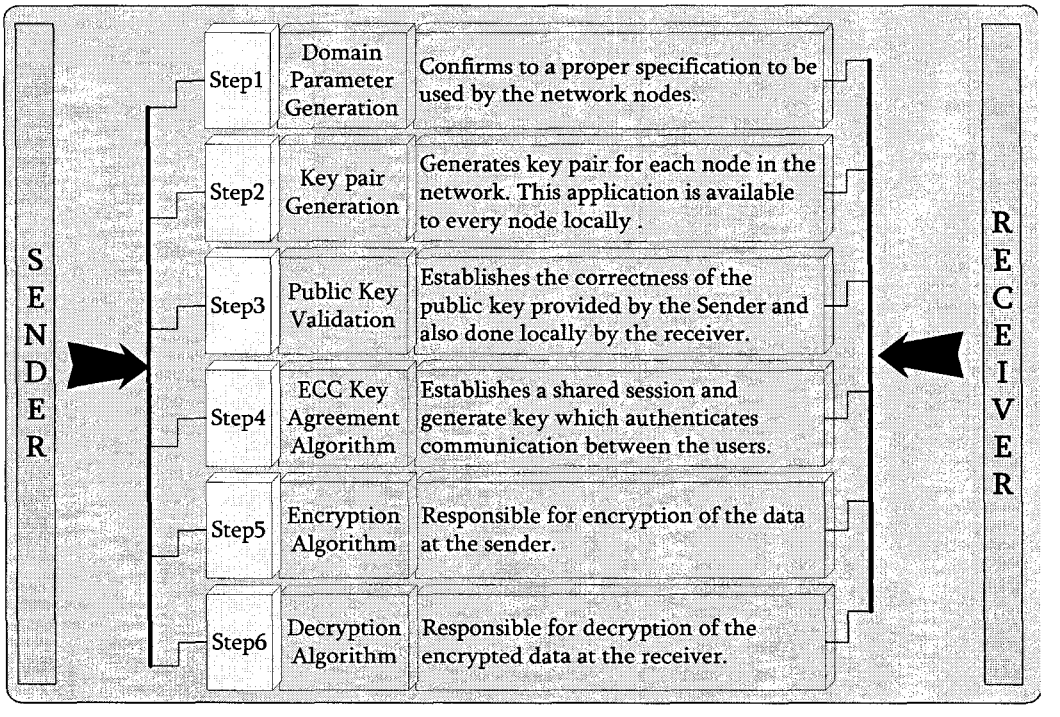

Figure 4-2: Proposed Encryption Scheme's Step-by-Step Flowchart

There are various steps (Figure 4-2) involved in the execution of the scheme. The first step is required to setup the curve and all its elements. It is expected that all the

communicating node pairs should agree on all of these elements. In the second step, they generate their own key using Algorithm *Keygen* in Chapter 3 and also validate the public keys using Algorithm *PubKeyval* in Chapter 3. The next step involves generating a shared secret key using Algorithm *Keyagree.* Once these steps are done, nodes can start encrypting messages and decrypting received messages using Algorithm *Keyencrypt* and Algorithm *Keydecrypt.*

4.3. Implementation Details

The scheme requires that the two parties must protect the keys from access to others especially from attackers. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if the key is compromised. In this section, the implementation of the security scheme is described which is divided into four modules (Figure 4-3):

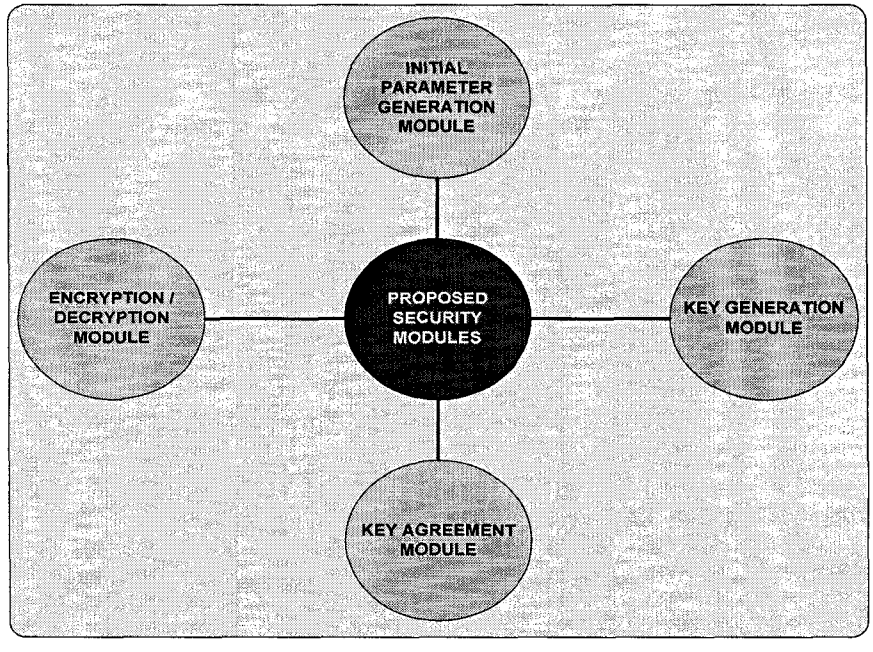

Figure 4-3: Implementation Modules

- Initial parameter generation module: which generates and assigns the DP to all the nodes within a network,

- **Key generation module:** which generates and assigns public and private keys to all the nodes in the network,
- **Key agreement module:** which ensures that a shared session key is established for any further correspondence and
- **Encryption/decryption module:** which allows users to encrypt and decrypt their messages to guarantee confidentiality and security of the message being sent to the receivers.

4.3.1. Scheme-Inputs and Outputs

In Table 4-1, the relationship between the parameters input taken by the nodes' calculation processor and the output generated from them are presented. For notation, List of Standard Notations could be referred to.

Table 4-1: Techniques Input and Output

Technique	Input	Output
DP Generation	Field order q; Security level L; Field representation FR for F_q	Domain parameters $(q, FR, S, a, b, P, n, h)$
DP Validation	Domain parameters $(q, FR, S, a, b, P, n, h)$	Valid/Invalid
Key Pair Generation	Domain parameters $(q, FR, S, a, b, P, n, h)$	Public key Q; Private key d
Public Key Validation	Domain parameters $(q, FR, S, a, b, P, n, h)$ Public key Q	Valid/Invalid
Encryption	Domain parameters $(q, FR, S, a, b, P, n, h)$; Public key Q; Plain text message m	Cipher text C
Decryption	Domain parameters $(q, FR, S, a, b, P, n, h)$; Cipher text; Private key d	Plain text message m

The first parameter that needs to be set is the DP which essentially defines the graph along which the elliptic curve is drawn. Once these parameters are set, they are tested and validated so that both communicating parties have the same values. The second step is to generate and validate the key pair of both communicating parties. With

these key pair validation, encryption and decryption of these data could be easily done.

4.3.2. Scheme Information Flowchart

In Figure 4-4, the flow of information is shown.

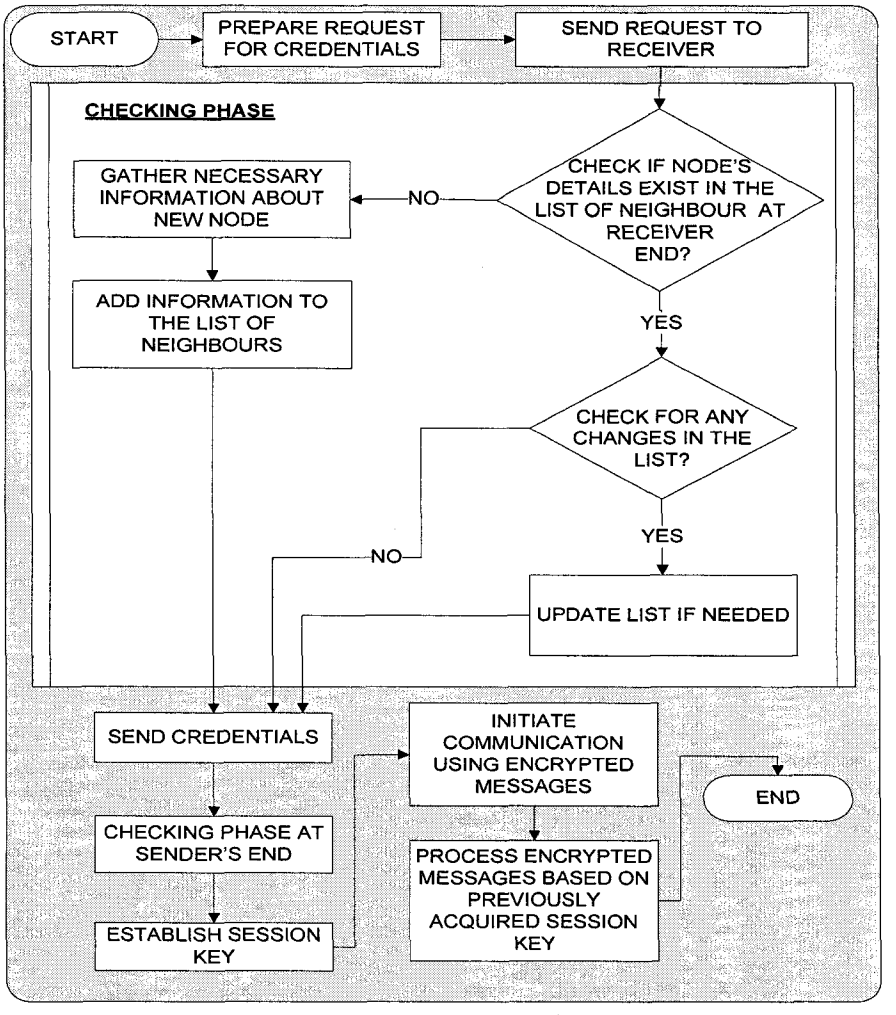

Figure 4-4: Sender-Receiver Information Flowchart

The first step includes initializing the session formation procedure which requires nodes to send their credentials. The receiver after checking and updating its records answers the sender's request for credentials. If the sender is a new node, then appropriate details are gathered and registered in the records. Once the processing at

the receiver's end is complete, the requested information is sent to the sender. The sender will then initiate its own checking phase. After establishing session key, encryption/decryption of messages can be done between the sender and the receiver.

4.4. Implementation

Implementation of this scheme needs selection of standard parameters and usable settings from a number of options [CER00]. These choices affect the outcomes of the encryption/decryption and must be chosen according to the application. The various parameters are: 1) Determining DP of elliptic curve, 2) Choosing key derivation function to derive secret key from known information, 3) Selection of a symmetric encryption scheme and MAC scheme, 4) Choice between standard and cofactor D-H primitives, 5) Selection of parameter and public key validation methods and 6) Selection of appropriate data to include in *SharedInfo*1 and *SharedInfo*2 which is optional. Each of this selection affects the course and outcome of the scheme.

To access an elliptic curve, it is mandatory that the curve is defined using DP and that both parties agree on all the elements. Symmetric encryption affects the length of plain text that could be encrypted and thus the amount of memory available. Using an appropriate MAC scheme is crucial since both security and bandwidth usage are equally important. Choices for MAC scheme [KRA97] are HMAC–SHA-1–160 scheme or the HMAC–SHA-1–80 scheme where the former offers more security and the latter offers more bandwidth savings. D-H primitive could either be standard or cofactor [JOH96]. Validation of parameters and public keys are very important since discrepancies in such parameters could lead to network failure. *SharedInfo*1 and *SharedInfo*2 are optional information which could be included depending on the particular application.

The scheme is implemented in four modules and the steps involved in implementing those modules are explained below.

4.4.1. Initial parameter Generation Module

This initial setup is required to prepare the communicating nodes to follow the same parameters which would be employed further in the scheme. The sender and the receiver in the ad hoc network should perform the following setup procedure as preparation for using the ECDH scheme [CER00]:

1. The sender and the receiver should choose the key derivation function (KDF) to use.
2. The sender and the receiver should choose ECDH primitive: 'standard' ECDH primitive or 'cofactor' ECDH primitive.
3. Communicating nodes need to setup the desired security level and based on the chosen setting, generate and validate ECDP $D = (q, FR, S, a, b, P, n, h)$ using the primitive described in **Algorithm** *Domparagen* and **Algorithm** *Domparaval* respectively.
4. The sender should establish the MAC schemes to use where *mackeylen* denotes the length in octets of the keys used by *MAC* and *maclen* denotes the length in octets of tags produced by *MAC.*
5. The sender should also select the symmetric encryption schemes (ENC) to use and denote the length in octets of the keys used by ENC by *enckeylen.*
6. The sender should obtain, in an authentic manner, the selections made by the receiver-the key derivation function KDF, the MAC scheme *MAC,* the symmetric encryption scheme ENC, the elliptic curve domain parameters *DP* and an indication whether to use the 'standard' ECDH primitive. The elliptic curve domain parameters $D = (q, FR, S, a, b, P, n, h)$ must be validated by the sender.

7. The sender should establish whether or not to represent elliptic curve points using point compression.

4.4.2. Key Generation/Validation Module

This setup is required to allow the communicating nodes to generate the public and private key pair which would be used in any further correspondence in the scheme. The sender and the receiver in the ad hoc network should perform the following key pair setup procedure [CER00]:

1. Once the DP is set, communicating nodes must generate their own elliptic curve key pair *(d, Q)* using the primitive described in Algorithm *Keygen* and then exchange the public keys *Q*.
2. Once these public keys are exchanged, both nodes must validate the received public key using the procedure described in Algorithm *PubKeyval*.

After establishing these initial settings, the communicating nodes must proceed to establish key pairs. Since these key pairs are the main parameters used for any further communication, the validation of the same is essential. Hence, the second step is the most important step until now. This validation confirms the usage of the same key on both the sender and the receiver ends. Due to the keys' lifetime and expiration time, keys might have expired or changed. It needs to be confirmed that the public keys exchanged are the same public keys used and stored in the databases of the generating nodes. Hence, this validation is crucial. The next step is to generate the shared secret session key.

4.4.3. Key Agreement Module

In order to generate a shared session key, the sender and the receiver must perform the following steps. For simplicity, while describing the step, it is assumed that the

sender is trying to send information to the receiver. To achieve communication from the receiver to the sender similar steps are followed but the roles of the sender and the receiver are reversed. The sender should establish keying data with the receiver using the keys and parameters established during the setup as follows [CER00]:

1. Using the chosen D-H primitives, derive the shared secret field element $z \in F_q$ from the sender's private key that was established previously and from the receiver's public key.
2. Convert $z \in F_q$ to an octet string Z using the conversion routine in [CER00].
3. Use the chosen key derivation function (KDF) to generate keying data K of length '*keydatalen*' octets from Z and (*SharedInfo*). '*keydatalen*' is the number of octets of keying data required. '*SharedInfo*' consists of some data shared by the sender and the receiver. If the key derivation function outputs 'invalid', output 'invalid' and stop.
4. Output K which is the keying data of length '*keydatalen*' octets or 'invalid'.

4.4.4. Encryption/Decryption Module

This module works in two steps: Encryption operation and Decryption operation. Encryption operation defines the steps involved in making the message unreadable to any user except the receiver. Decryption operation defines the steps involved in making the message readable only to the receiver node.

4.4.4.1. Encryption Operation for the Scheme

The node that wants to send information to any receiver should perform the encryption operation described in this section to communicate using encrypted data with the help of ECIES. For simplicity, while describing the step, it is assumed that the sender is trying to send information to the receiver. To achieve communication from the receiver to the sender similar steps are followed but the roles of the sender

and the receiver are reversed. The encryption procedure for ECIES is specified below using the keys and parameters established during the setup procedure [CER00]:

1. Using the keypair generation procedure described in **Algorithm** *Keygen* generate an ephemeral elliptic curve key pair (k', R) with $R=(X_R,Y_R)$ which is linked with the ECDP D.
2. Using conversion routing, calculate the octet string R' from R and also choose representation of R using point compression.
3. Derive shared secret field element $z \in F_q$ from the ephemeral secret key K and the receiver's public key using the D-H primitives. If the D-H primitive outputs 'invalid', output 'invalid' and stop.
4. Convert $z \in F_q$ to an octet string Z using the conversion routine described in [CER00]. If the conversion routine outputs 'invalid', output 'invalid' and stop.
5. Using *KDF,* generate keying data K of length *enckeylen* + *mackeylen* octets from Z and (*SharedInfo*1). If the key derivation function outputs 'invalid', output 'invalid' and stop.
6. Separate and analyze the left most *enckeylen* octets of K as an encryption key *EK* and the right most *mackeylen* octets of K as a MAC key *MK*.
7. Use the chosen symmetric encryption scheme ENC to encrypt the user provided message M under *EK* as cipher text *EM*. If the encryption scheme outputs 'invalid', output 'invalid' and stop.
8. Using MAC scheme compute the tag D on $EM \,||(SharedInfo2)$ under *MK*. If the MAC scheme outputs 'invalid', output 'invalid' and stop.
9. Output $C\ R'||EM||D||$.

4.4.4.2. Decryption Operation for the Scheme

The node that wants to decrypt the received information from any the sender should perform the decryption operation described in this section to understand the message

with the help of ECIES. For simplicity, while describing the step, it is assumed that the sender is trying to send information to the receiver and the receiver is trying to decrypt the encrypted message. The receiver should be able to decrypt cipher text using the keys and parameters established during the initial setup as follows [CER00]:

1. When the left most octet of cipher text $R' \| EM \| D \|$ is 02_{16} or 03_{16}, then parse the $\left\lceil \log_2 q \right\rceil + 1$ octets of cipher text as octet string R', the right most *maclen* octets of cipher text as octet string D and the remaining octets of cipher text as octet string *EM*. When the left most octet of C is 04_{16}, then parse the left most $2\left\lceil \log_2 q \right\rceil + 1$ octets of C as an octet string R, the right most *maclen* octets of C as an octet string D and the remaining octets of C as an octet string *EM*. If the left most octet of C is not 02_{16}, 03_{16} or 04_{16}, output 'invalid' and stop.
2. Use the conversion routing and convert the octet string R' to an elliptic curve point $R = (X_R, Y_R)$ associated with the ECDP D. If the conversion routine outputs 'invalid', output 'invalid' and stop.
3. Confirm that R is a valid elliptic curve public key. If an appropriate assurance is not obtained, output 'invalid' and stop.
4. Derive the shared secret field element $z \in F_q$ from the receiver's private key and the public key R. If the D-H primitive outputs 'invalid', output 'invalid' and stop.
5. Convert $z \in F_q$ to an octet string Z using the conversion routine.
6. Using *KDF*, generate keying data K of length *enckeylen* + *mackeylen* octets from Z and [*SharedInfo*1]. If the key derivation function outputs 'invalid', output 'invalid' and stop. Then parse the left most *enckeylen* octets of K as an encryption key *EK* and the right most *mackeylen* octets of K as a MAC key *MK*.
7. Using MAC scheme check that D is a tag on *EM* || (*SharedInfo*2) under *MK*. If the MAC scheme outputs 'invalid', output 'invalid' and stop.

8. Use the chosen symmetric decryption scheme ENC to decrypt the *EM* as message *M* under *EK*. If the encryption scheme outputs 'invalid', output 'invalid' and stop.

9. Output *M*.

Chapter 5

Development of the Testbed

The research scheme developed needs to be simulated under different sets of parameters and analyzed under pre-determined evaluation criteria used to study the performance. In this chapter the testbed configuration and its design, simulation setup and input parameters are discussed.

5.1. Testbed – An Overview

Testbed, by definition, is the platform used for experimentation that allows rigorous, transparent and replicable testing of scientific theories, computational tools and other new technologies. Testbed gives researchers a wide variety of environments in which development, debugging and evaluation of their systems could be done. The evaluation of the system is based on set of performance parameters discussed later in the chapter. Performance is a key criterion in the design which sets its goal to get the highest performance for a given cost. To achieve that goal, a performance evaluator needs, at least, a basic knowledge of performance evaluation terminology and techniques with the ability to state the performance requirements of their systems and should be able to compare different alternatives to find the one that best meets their requirements.

To support the dynamic nature of the system and reduce complexity, the design of the system is done on three levels. All these levels, in conjunction with each other, facilitate the user-selected process to be completed and the results displayed for evaluation. As the process continues, the monitoring scheme automatically collects data file and evaluates information pertaining to the defined events and recognizes event occurrences. By providing flexible structuring and dynamic configuration of the monitoring system, multiple time view protocols and the separation of event and monitoring specification, this testbed provides the capability of developing, testing and comparing various approaches.

5.2. Design of the Testbed

In this section, the architecture of the testbed is explained. Dynamic number of nodes present in a network can have a serious impact on the accuracy of the message delivery and also on the performance of the network. This is one of several parameters that could affect performance and they are described in section 5.3. The testbed system is developed with the following design objectives:

- Automation to monitor and process the data and the results.
- Accuracy of performance measurements
- Adaptability of the system to different users
- Reducing complexity and enabling ease of use
- Allowing coding for the techniques into software programs
- Retrieving performance data acquired under different conditions.

In this testbed, automatic monitoring is provided by Graphical User Interface (GUI) and NS-2, discussed later in the chapter. Any user with a minimum knowledge of the system can easily select parameters, run simulation and collect results. Since all the results are stored in a log file, referencing and analyzing results to a previous scenario

can also be done using this testbed. The three major components in the testbed system, as described in Figure 5-1 are:

- GUI: System module
- GUI: User module
- NS-2

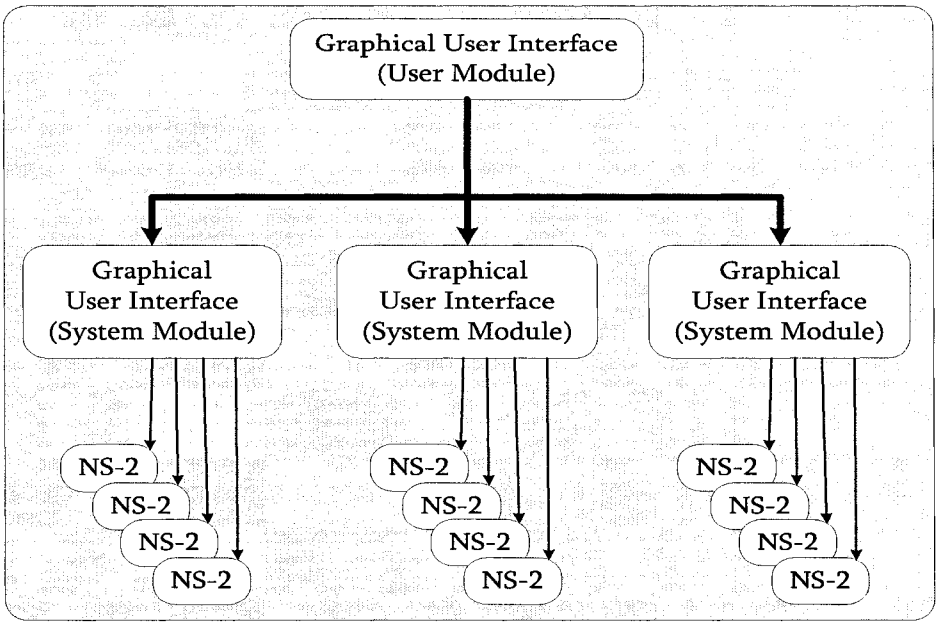

Figure 5-1: Testbed Design

As it can be seen from the diagram, the user controls all the experimentation using a GUI. This interface takes the user inputs provided in the main window and then transfers them to the system module onto NS-2 which handles the coding for the selected technique. A detailed description of all three components is given later in the chapter. Several NS-2 instances can be used to derive results while working in conjunction to provide the performance results. In practice, the user can only interact with the GUI by selecting parameters to run simulations. Several parameter issues must be taken care of such as number of nodes in the user-defined network area, which can seriously affect the performance of the system. When the network is dense, most of the schemes do not work well since the performance is also dependent

on the traffic patterns. Therefore, a robust scheme should be able to estimate accurately under these dynamic conditions. The focus is to test schemes that work well with high number of nodes, variable area of network and variable traffic. Along with the number of nodes, the other factor affecting the accuracy of the estimate is the degree of movement of the nodes and the rate at which the data is transferred.

5.2.1. User Module

The user module seen in Figure 5-2 is designed in JAVA. It provides the interface for the user to interact with the underlying code and execute simulations. It also gives the user the facility to select parameters run simulation and then derive results based on those parameters. The user module is made up of three main components:

- **User Interface:** acts as the main window that provides user access to the underlying code. This window allows the user to interact with the system. It contains functions such as file menu, run simulation, select parameters and gets results. These functions can be called in a particular order and will be highlighted only when they could be used. For example, get result can only be used after run simulations.
- **Parameter Selector:** provides a menu to select the parameters for the simulation. The value of these parameters could be fixed, variable or user defined. This interface also checks the validity of the selected parameters as per the rules provided.
- **Simulation Engine:** is the main interface that allows access to the system module installed on the different computers. It consists of two main sub-modules namely system module interface and result manager. As the name suggests system module interface provides the communication between user module and the system modules. It also controls the number of parallel executions depending upon the desired level. Second sub-module called the

result manager handles result files. It stores the simulation results in a file. This sub module uses system memory to store trace files obtained from the NS-2. It is also responsible for displaying the results from trace file as trace graphs.

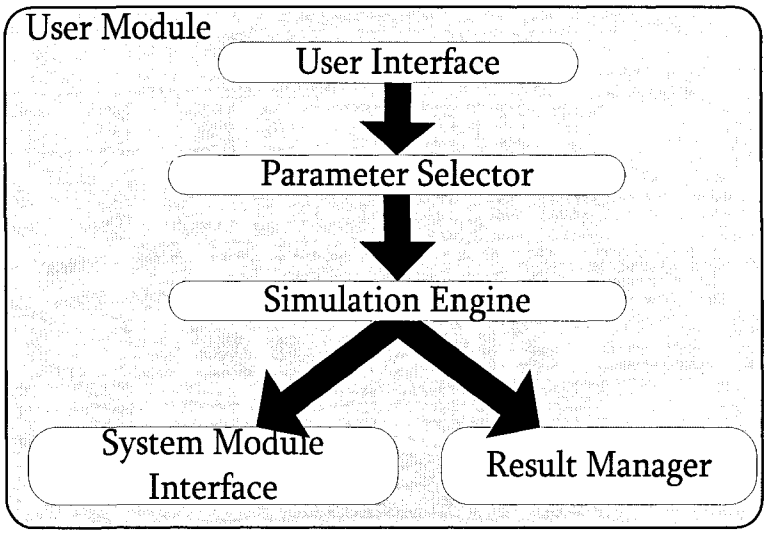

Figure 5-2: GUI User Module

5.2.2. System Module

The system module is the main module that provides the bridge for communication between the code and the user. It handles all factors based on which results could be derived. In Figure 5-3, the architecture of this module is described. It consists of a scenario generator, a traffic generator and a script generator which use input from the user obtained via user module's system module interface. The system module interface allows the user to input various parameter values to be used while running simulations.

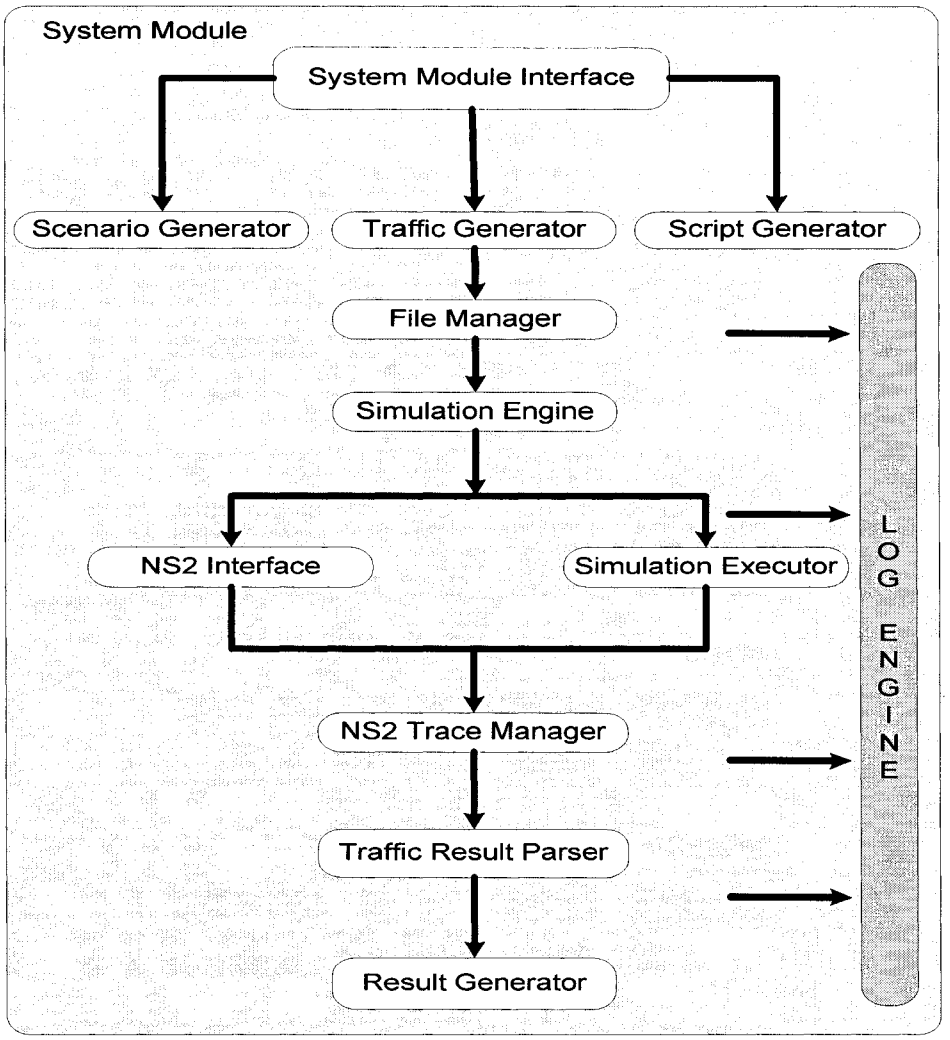

Figure 5-3: GUI System Module

The system module consists of eleven sub modules:

- System module Interface: basically communicates with the user to get the simulation parameters and commands and provides the results obtained from those simulations.
- Scenario Generator: generates the NS-2 scenario file based on the selector parameters.
- Traffic Generator: generates the NS-2 traffic file depending on the input parameters provided by the user through the system module interface window.

- **Script Generator:** generates the Tool Command Language (TCL) script file based on the user defined parameters.
- **File Manager:** handles the files generated as the result of parameter selection. It handles all the file generation, deletion and also storage function related to the files.
- **Simulation Engine:** provides the execution interface to NS-2 and is responsible for running the simulation. **NS2 Interface** and **Simulation Executor** are part of Simulation Engine.
- **NS2 Trace Manager:** is responsible for all NS-2 trace file generated as the result of simulation. It handles NS-2 files, their storage and also scanning functionality.
- **Traffic Result Parser:** is individually responsible for the scanning of trace file and then collecting all the necessary information in such a format that it could be used to derive system information.
- **Result Generator:** as the name suggests is in charge of all the result information received from the parser. It converts this valuable information into a format so that the user could clearly identify performance parameters such as delay, packet delivery fraction (PDF), number of hops etc. This sub-module serves as the analysis centre and presentation centre of the testbed.

5.3. Factors Affecting the Performance of the Scheme

Performance is a key criterion in the design which decides the fate of any new scheme since these experiments and evaluation decide how the scheme is performing under different environments. The goal of performance measurement is to test the scheme many times under different circumstances for several variable parameters. The comparison of the results ensures that the scheme gives the highest performance for a given cost. Since several factors might affect the performance, a performance

evaluator needs, at least, a basic knowledge of performance evaluation terminology and techniques.

There has not been a detailed study of the performance of secure routing protocols for such dynamic networks. There has been only a comparative study for some protocols under various loads, size of network and working environments. One of the main roadblocks in such networks is the very dynamic nature and temporally and spatial varying topology. As a result, the chances of link establishment and failure increases by leaps and bounds when there is an increase in the probability of frequent packet dropping and of multipath effect. Another roadblock is the limited memory on the network nodes.

To evaluate the performance of the security scheme, several factors must be considered namely, the number of nodes, individual node capacity, the delay in message delivery and traffic patterns.

5.3.1. Number of Nodes and Their Abilities

The first important factor to be kept in mind while applying the security scheme is the number of nodes in the network and the area they cover. Another factor that affects data delivery is the range of individual nodes. More nodes mean more complex network and therefore, the need for even complex security framework to be implemented at various levels. This can cause failure in message delivery which is expensive both in terms of time and battery power. Message could also be lost if they are transmitted on the periphery of the area covered by the node. So it is really beneficial that the nodes use intermediate nodes to transfer data along the network.

The individual abilities of the nodes are also an important factor. These abilities can be defined in terms of processing power, battery life and memory. Processing power

of mobile nodes could be really low. Such mobile nodes would require that there should be minimum computations while operating. More computations also mean that more battery power will be used and similarly more memory. Higher usage of memory space will also take its toll on the processing speed. Therefore, minimizing these effects or at least maintaining a certain threshold will greatly improve the performance of the scheme.

5.3.2. Nodes and Traffic Patterns

In this research, the impact of traffic on nodes is studied. Several experiments are performed to test the dependencies among the nodes and on traffic patterns. Since every node has limited capacity in terms of memory space and computation power, it becomes relatively very difficult for these small wireless nodes to rapidly transfer data and also to receive data. All the nodes have a definite battery life which again constraint their message transmission ability. The message transmission in the ad hoc network is supported mainly by hopping over intermediate nodes. If an intermediate node fails to transmit messages for other nodes, the trust value for that node drops and in future the value may becomes low enough to exclude that node from the network altogether.

These issues might complicate the network functioning. Therefore, it is initially assumed that the network can handle any traffic load and the nodes work smoothly. However, this assumption is not necessarily true all the time, since nodes work dependently and the delay caused in data transmission can further delay the next transmission. In this work, two levels of traffic pattern, heavy and moderate, are used for the simulations.

5.3.3. Systems and their Abilities

The first important factor affecting the performance of the testbed is the power of the individual computer systems being used. Each of these systems may have different configurations of their own and the results of simulation may vary with individual system performance. Some experiments are performed to check the effect of the computer system power configuration and their results are presented in section 6.5. A more powerful computer is expected to perform better than the others. Powerful computer systems are faster, have more computing capabilities and more memory.

5.4. Working of the Testbed

Usually testbeds are complicated to operate but this one is made simple by the use of GUI and NS-2. In order to analyze a network, it should first be defined using parameters from GUI which allows the user to configure any network, traffic and technique to be used in the network. Once these values are obtained, simulation engine uses its system module interface to send these values to the script generator. A new file is also created by the file manager to store the scenario details and the results obtained from these simulations. The script generator accesses NS-2 interface to run simulations and then creates a trace file for the same. The results from these simulations are collected and stored by result generator in a log file.

5.5. Simulation Parameters

Simulation setup and its performance are based on the parameter selection which define the setup used for the implementation and simulation of the scheme.

- **Area of the Network:** describes the area of the network using X and Y coordinates. The input for this parameter is the value of X and Y. The value of X-coordinate and Y-coordinate can be between 0 and 1000.

- **Pause Time:** defines the number of milliseconds to pause, allowing the user to see some information. The unit for this parameter can be either milliseconds or seconds.
- **Seed:** is generated by a random number generator object and determines the data flow. The value of this seed can be between 0 and 10.
- **Ad hoc Routing Protocol:** defines which routing protocol is being used. Although various other options could be added through NS-2, the choices for routing protocol for this GUI are DSR, DSDV, AODV, OLSR and LAR.
- **Number of Nodes:** defines the number of nodes being used in the network. The number of nodes indicates the density and complexity of the network. There could be 100 to 500 nodes in the network.
- **Simulation Time:** is necessary to control the time of simulation runs. The unit for this parameter is seconds. Usually the simulation time could be between 100 and 1000.
- **Mobility:** defines the degree of movement of the nodes in the network. The value for mobility can be any number other than zero, as zero means that the nodes are not moving at all.
- **Rate:** defines the rate at which the data is transferred from one node to the other throughout the network. The value for rate can be any number other than zero, as zero means that the nodes are not transferring at all.
- **Network Traffic Type:** defines the type of traffic in the network. The traffic could be using a Constant bit rate (CBR) network traffic generator or other traffic generator such as TCP/IP, UDP or ICMP. GUI uses CBR which gives a constant traffic with a steady bit rate for a given interval of time.
- **Propagation:** defines the type of transmission techniques being used in the network. This parameter could be any of the four schemes which are used to predict the received signal power of each packet. At physical layer there

is a receiving threshold which sorts receiving signal based on their strengths. If the receiving signal strength is less than the receiving threshold, the packet is marked as error and dropped. There are four models: Free space, Two-ray ground, Shadowing and Hybrid. Free space model assumes only one line of sight path between transmitter and receiver. Two-ray model considers both the direct path and a ground reflection path. The shadowing model extends the ideal circle model to a richer statistic model: nodes can only probabilistically communicate when near the edge of the communication range. Hybrid propagation model uses a bit of all the propagation discussed above.

- **Encryption Algorithm:** sets which type of encryption scheme are used.
- **Bits:** define the amount of data being sent from one node to other throughout the network.
- **Key Length:** is an important security parameter. Key size or key length is the size (usually measured in bits or bytes) of the key used in a cryptographic algorithm (such as a cipher).

5.6. Simulations and its Setup

The main use of a testbed is for improving the software or a product under development. Different scenarios could be generated from the testbed and then tested to analyze the result. Several computer systems may run the simulations at one time and each of these computers may have a different configurations. Even complex security framework can cause failure in message delivery and since each message delivery failure costs a lot both in terms of time and battery power, security framework has to be lightweight and simple. Every node has a definite capacity for message transmission such as node memory, battery life etc. In this section, the tools

used to carry out the simulations are described. For the simulation of proposed scheme, NS2 and GUI are the main tools.

5.6.1. Network Simulator (NS-2) – Overview

Network Simulator is the software which allows the user to imitate a real world network environment and is designed to provide diverse network, simulate and then analyze the effect of different parameters on the network performance. Any user can use this simulator as an inexpensive testbed with a capability to simulate rather difficult scenarios such as sudden increase in traffic or attack providing the equivalent capabilities of an entire testbed. The entire testbed contains multiple computers, networked together with routers and data links where performance usually depends on the performance of the individual computer. Simulators, however, are relatively high-speed and economical and thus provide the best environment for the development of a new protocol or implementation of any new security scheme in a controlled and reproducible environment. Among several others, Network Simulator version 2 (NS-2) is used for this work. NS-2 [FAL00], a variant of the REAL network simulator in 1989, is a network simulator to perform and analyze event occurring in network. It provides powerful features for simulation of TCP, routing and multicast protocols over wired and wireless (local and satellite) networks. Since 1989, several versions of NS-2 have been released by DARPA.

5.6.1.1. NS-2 – Setup

NS-2 (Figure 5-4) [FAL00] is based on two different languages, namely, object-oriented simulator written in C++ and an Object Tool Command Language (O-TcL) interpreter used to execute user's command script. It provides a library of networks and protocols to incorporate comparison capability. It also incorporates a visualization tool [FAL00] to allow graphical representation of the network scenario.

5.6.1.2. Selection of Parameters

Simulation setup and its performance are based on the parameter selection. These parameters define the setup used for the implementation and simulation of the scheme. The parameters used by NS-2 are 1) Area of the Network, 2) Pause Time, 3) Seed, 4) Routing Protocol, 5) Number of Nodes, 6) Simulation Time, 7) Mobility, 8) Rate, 9) Network Traffic and 10) Propagation.

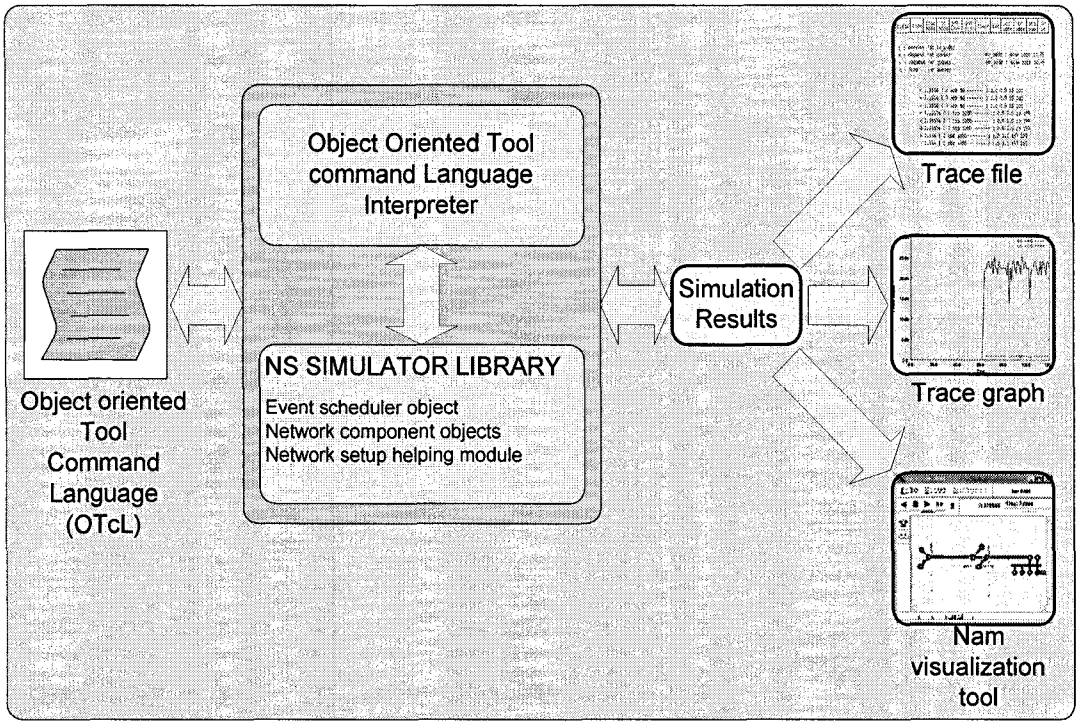

Figure 5-4: NS-2 Simulation Model

5.6.1.3. Study Points

The key points for this simulation are to set up NS-2 and configure it to display preferred parameters. To provide an easy access to parameters settings, a simple GUI has been generated as described in section 5.6.2. Once parameter settings are done and network is generated, NS-2 runs and outputs a result file. These results are analyzed based on the delay, normalized routing load, packet delivery fraction, overhead and average hop count.

5.6.2. Graphical User Interface - Overview

Graphical user interface (GUI) is a platform for user interaction, developed and customized by the user himself with a goal to enhance the efficiency and the ease of use for the underlying logical design of a stored program, a design discipline known as usability. It is a two dimensional display with graphical icons, visual indicators or special graphical elements called widgets and acts as a shortcut for computer functionality and computer-controlled devices. Text, labels or text navigation is used to permit information representation and perform user defined actions by accessing elements through GUI rather than through text menus or commands. The first GUI developed by Douglas Engelbart [ENG62] used text based hyperlinks manipulated by mouse. With the advancements and latest technology, most systems like Microsoft Windows, MAC OS X and X Windows system are using it. GUI uses a physical input device to control the position of the cursor, present information organized in windows and represented with icons. Based on the input, it compiles available commands and delivers requested information results.

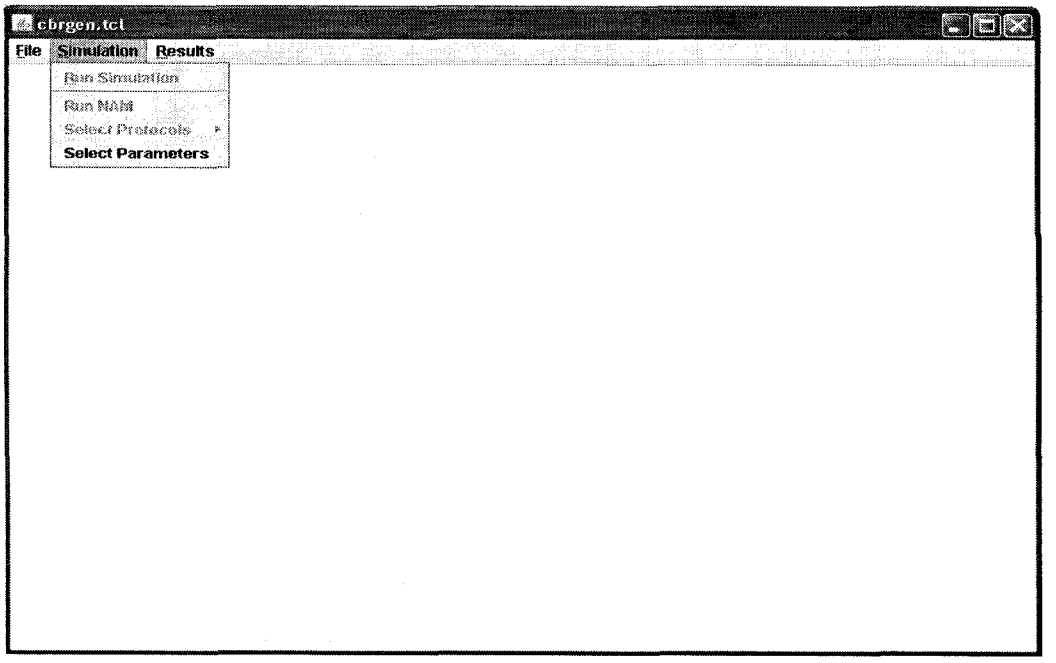

Figure 5-5: GUI Menu

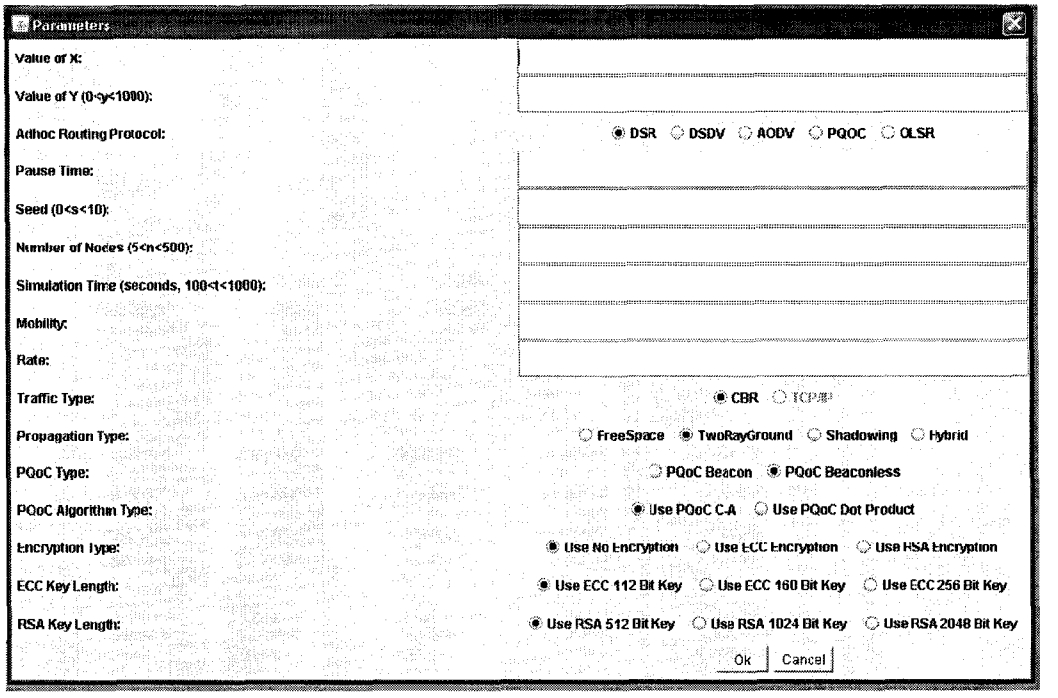

Figure 5-6: GUI Main Parameter Selector Window

For these simulations, a very simple GUI (Figure 5-5 and Figure 5-6) has been designed which allows the selection of various parameters. Once these parameters are selected GUI allows interoperability to the underlying code and processes code without the user having to run command on its own. This only makes it simpler to access various functions used in the code and thus reducing the complexity. As seen from the GUI, several selection parameters are available. Some inputs are done by check boxes while others need a numeric user input. This window is the front end of the whole processing being done with the code written using JAVA and C.

5.6.2.1. Selection of Parameters

Simulation setup and its performance are based on the parameter selection. These parameters define the setup used for the implementation and simulation of the scheme. For this work, performance of the system is evaluated based on the following parameters. The parameters for GUI are 1) Value of X-coordinate, 2) Value of Y-

coordinate, 3) Ad hoc routing protocol, 4) Pause time, 5) Seed, 6) Number of nodes, 7) Simulation time, 8) Mobility, 9) Rate, 10) Network traffic type, 11) Propagation type, 12) Encryption type and 13) Key lengths.

5.6.2.2. Study Points

The key points for this GUI to run simulation are selecting the parameters required to configure the network scenario. The main aim of developing this GUI is to provide an easy access to parameters settings without having to manage the code. Once parameter settings are done and the network is generated, NS2 runs and outputs a result file as a log. The proposed security scheme is tested using three different routing schemes namely Ad Hoc On-demand Distance Vector Routing Protocol (AODV) [PER99], Optimized Link State Routing Protocol (OLSR) [CLA03a] and Location Aided Routing Protocol (LAR) [KO98]. The results of each are shown and discussed in section 7.3.1, section 7.3.2 and 7.3.3 respectively.

5.7. Evaluation Parameters

The parameters that have been selected for performance evaluation of the scheme are described below. Each of these parameters has an effect on the performance of the scheme since the variation in the value of the parameter increases the time taken by the scheme for encryption and decryption. This value of time when compared against other proven schemes needs to be less, which will prove that the proposed scheme presents some improvement and that it performs better than other schemes. For the present work, performance of the system is evaluated based on the following parameters:

- Network Size: measures the total number of nodes in the network.
- Topological Rate of Change: measures the speed with which a network topology is changing.

- **Link Capacity:** measures the effective link speed measured in bits per second.
- **Control Overhead:** measures the number of control packets transmitted for every data packet.
- **End-to-end Delay:** measures the average time a packet takes from its source to its destination. It is given by $Delay_{end\text{-}to\text{-}end} = N\left[delay_{transmission} + delay_{propagation} + delay_{processing} \right]$.
- **Average Hop Count:** measures the number of hops taken by data packets.
- **Packet Delivery Fraction:** (PDF) measures the ratio of number of data packets delivered to the receiver and the number of packets transmitted by the sender.
- **Routing Overhead:** measures the algorithm's internal efficiency and is calculated as the total number of control packets sent divided by the number of data packets delivered successfully.
- **Normalized Routing Load:** (NRL) measures the ratio of the number of control packets propagated by every node in the network and the number of data packets received by the destination nodes.

5.8. Configuring the Testbed

The next step is to configure the testbed so that experiments using the testbed, GUI and NS-2 could be performed. In order to use the testbed, the setup files on the system chosen for experiments must be installed first. It should be noted that this system should have enough memory capacity to complete installation and store results because the log file could be big. Since NS-2 works in the Linux environment, it should be easily installable on the Linux environment; but for Windows, Cygwin will be required.

5.9. Interacting with the Testbed

To allow ease of use for the testbed, a front end GUI is designed. This GUI as shown in Figure 5-5 and Figure 5-6 allows easy access to underlying parameters. This GUI

opens up as a dialog box with option menus such as file, simulations and results. Each of these options is clickable drop down menus. In order to run simulations first the parameters are selected. This can be done by clicking on the simulation button and selecting "select parameters". This will open a new dialog box as shown in Figure 5-6 where the user can insert parameters values.

Once these parameters are selected, run simulation tab is pressed to run simulation based on the user selected parameter values. Upon finishing, the results could be obtained in the form of a log file, presented as trace graphs or displayed in a visualization tool like Network Animator (Nam) (Figure 5-7) [FAL00].

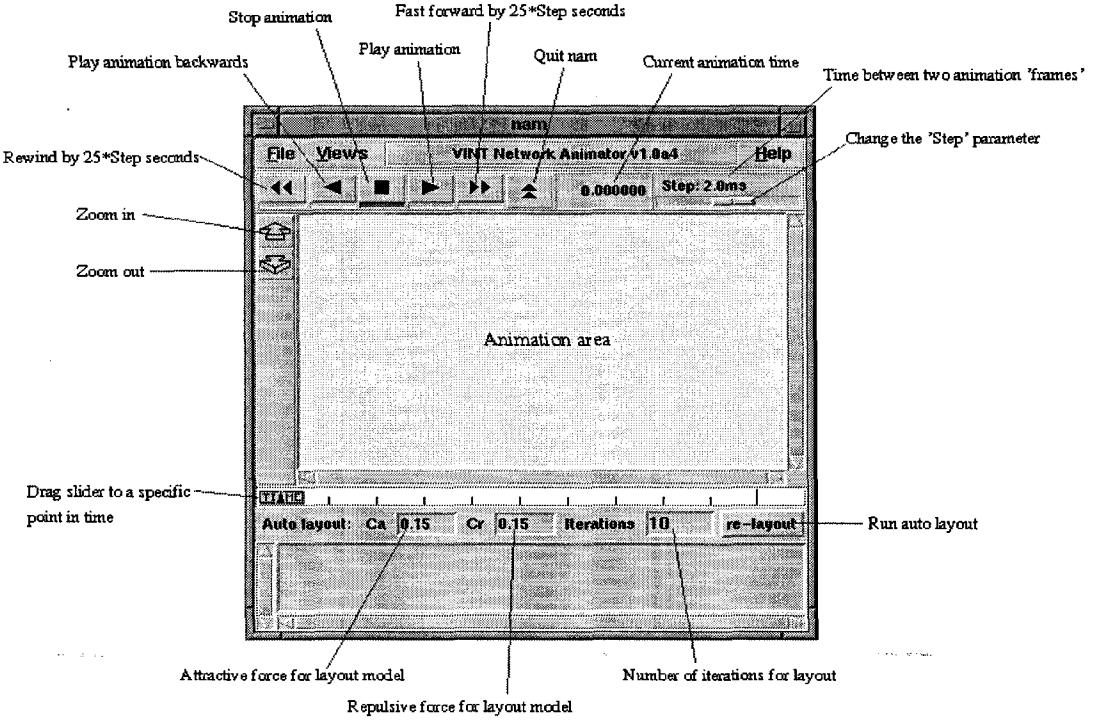

Figure 5-7: Using Nam from the Testbed (Source: [FAL00])

5.10. Using Testbed to Read and Visualize Results

Simulation model (Figure 5-4) in NS-2 produces one or more text based results and contains all the details about the parameters set and the output data. These files are

referred to as trace files and could be used for analysis. However, for graphical visualization of the data these trace files are used as input and displayed using a graphical simulation display tool called Network Animator that is developed as a part of VINT project.

Using GUI, the results could be viewed as log file. The results could be modified to show pattern changes in particular parameters. These pattern changes could also be identified using the trace graphs. A comparative study of two different scenarios could also be done by comparing the evaluation parameters results obtained after simulations. These results could be ported to MS Excel also. Each simulation scenario is run for at least 30 times and then the results are analyzed.

Chapter 6

RSA versus ECC – An Experimental Comparison

Experiments to check the performance of RSA and ECC in the presence of the variable factors are performed. The experimental results are presented in terms of size data, time data, percentage increase in size data and average encryption/decryption time taken based on various network sizes and data load. Since simulations are machine dependent, their effect is also discussed later in the chapter.

6.1. Overview

A simulation model is, in general, used to study real life systems which are either in the process of being developed or being tested for an optimal performance. In particular, these simulations are performed mainly to quantify the performance of the system for variable value of the input parameters. These measures test if the current system is performing according to the expected behaviour.

RSA and ECC are both encryption schemes used mainly for encryption and decryption of data. Since RSA is a very well-known standard and ECC is not a popular choice among cryptographers, it was best to compare these two schemes. To prove the efficiency of ECC, it must be compared to an existing standard scheme such as RSA. These comparative simulations are executed to study the performance of RSA and

ECC when several variable parameters influence the scenarios. The experimentation setup is described in section 6.2.

6.2. Experimental Setup

An illustration of the experiment setup is provided in Figure 6-1.

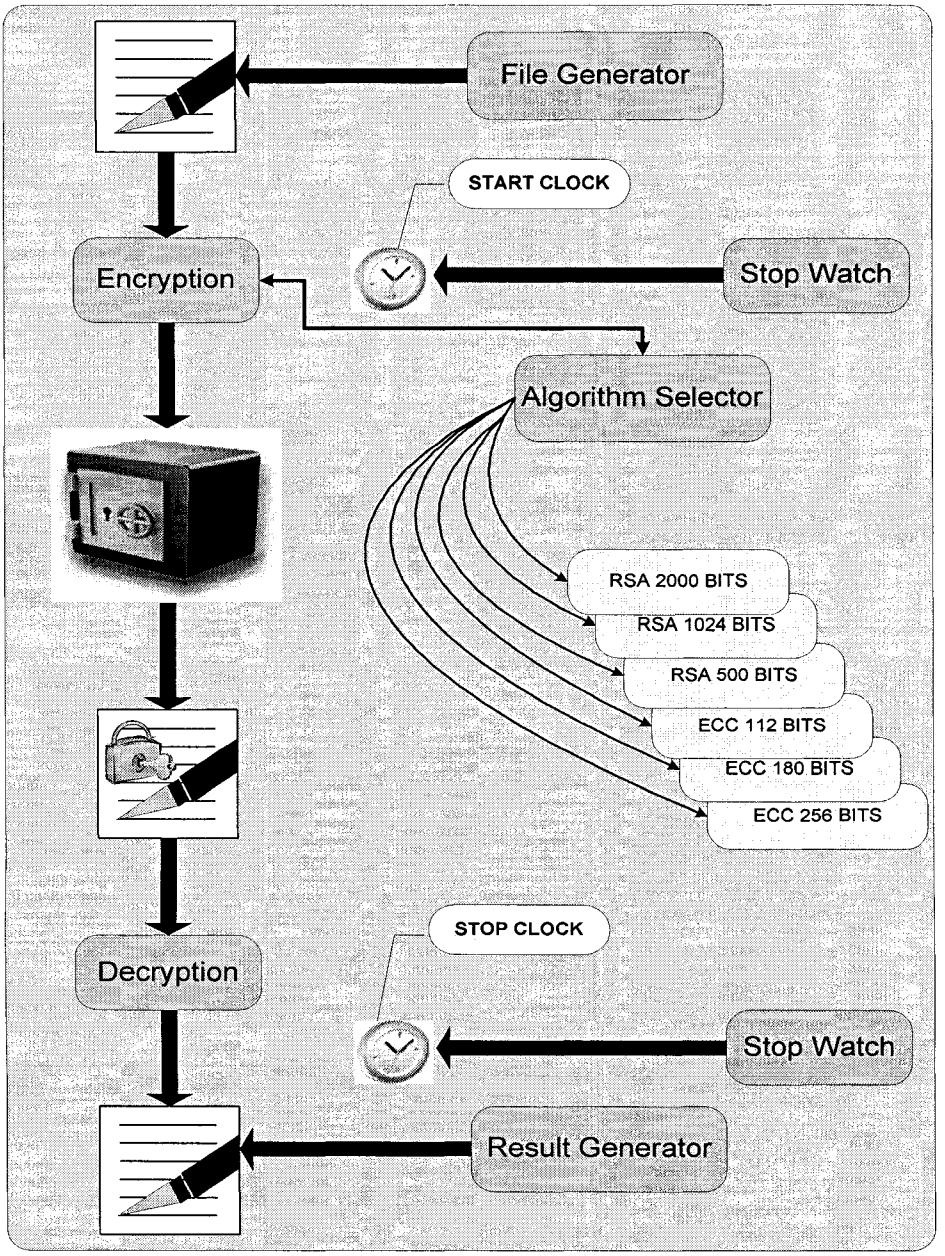

Figure 6-1: Block Diagram of Encryption Engine

The main variables in these simulations are the number of nodes, data load, time taken to encrypt/decrypt and the size of encrypted file. The setup for this experiment contains a file generator, an encryptor/decryptor, an algorithm selector and two stop watches. The encryption is performed over the input file and decryption is performed on the scrambled input file. Stop watches are placed to calculate the total encryption/decryption time for different types of encryption schemes. The result generator collects the results for these different scenarios and stores it locally at the pre-specified location.

6.3. Experimental Results

A comparison of performance of various ECC and RSA key lengths is presented. The graphs include the variable encrypted file sizes in bytes on the y-axis and variable original file sizes in bytes on the x-axis. Size data indicates the size of the encrypted files for different algorithms. Time data indicates the time taken to encrypt and decrypt files for different algorithms. Size data percentage mainly indicates the percentage increase in the data size of the encrypted files for different algorithms. These experiments are done using RSA with key lengths of 2048, 1024, 512 bits and ECC with key lengths of 256, 160, 112 bits.

6.4. Experimental Data Analysis

The results obtained from the trace file of NS2 are further illustrated graphically. The graphs (Figure 6-2, Figure 6-3, Figure 6-4 and Figure 6-5) show encrypted file sizes for different original file sizes for different algorithms.

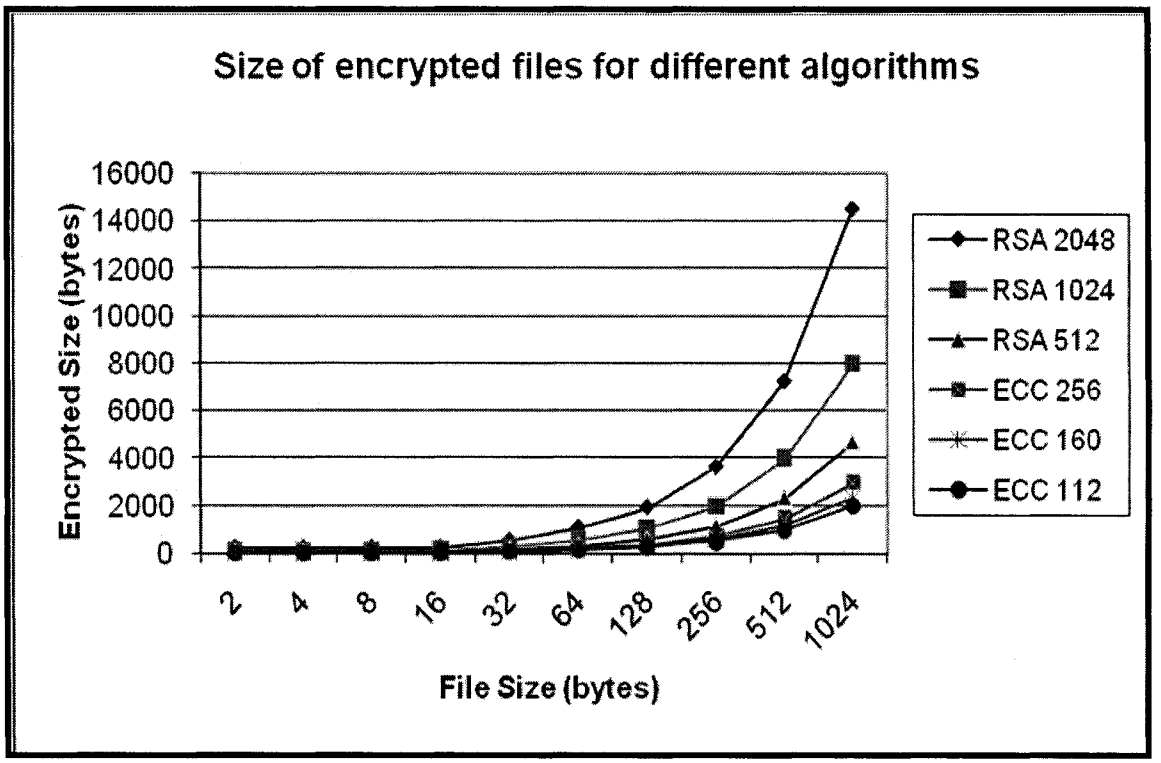

Figure 6-2: Size of the Encrypted file for Different Algorithms

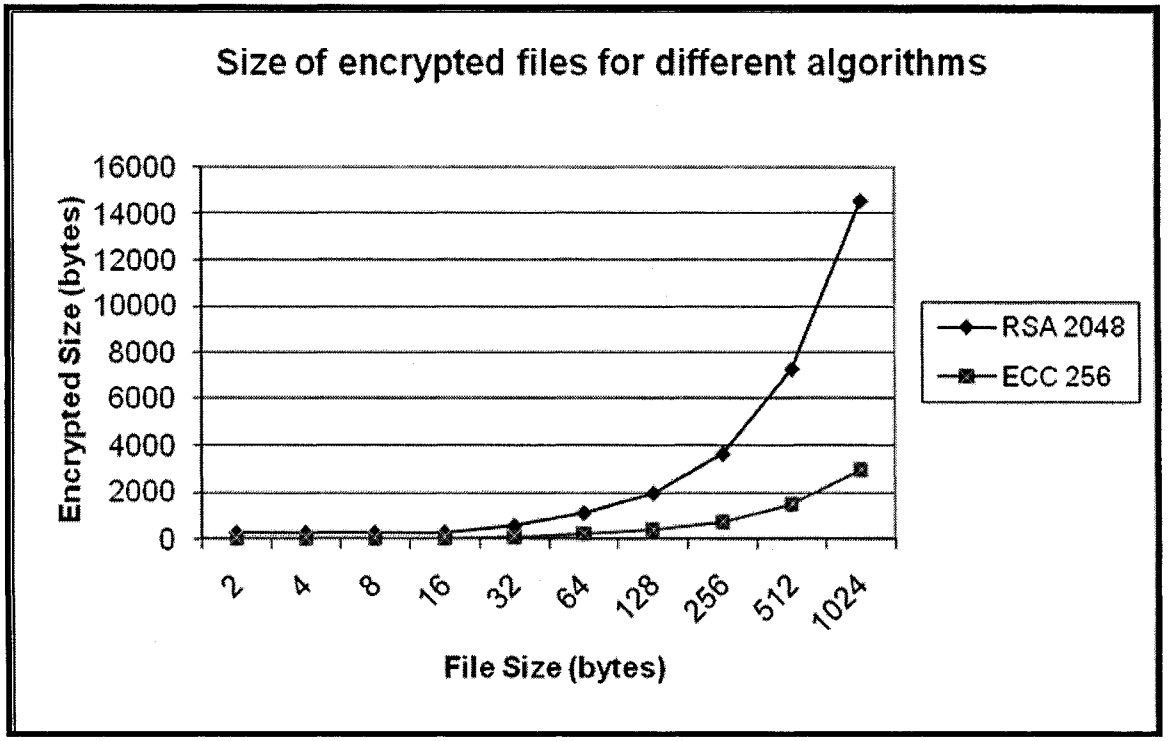

Figure 6-3: Size of the Encrypted file for ECC 256 and RSA 2048 Algorithms

From Figure 6-2, it is observed that the encrypted file size increases manifolds for RSA since ECC has comparatively shorter keys than RSA. Another observation was that the increasing file size of data to be encrypted also increases the time consumed. ECC, however, does not increase the file size so much after applying encryption. For RSA 2048, the size of encrypted files increases rapidly. This proves the complexity of RSA. For ECC, the files size increase is gradual with the key length increase. If equivalent key lengths are compared, the difference in file size is almost 6000 bytes. ECC gives the best performance for shorter key lengths. In Figure 6-3, Figure 6-4 and Figure 6-5, the different key sizes are compared. It can be easily seen from the graphs that the size of encrypted file increases rapidly for RSA. For ECC, the increase in size of encrypted file is still less which is an advantage for the proposed scheme. The greater the size of file, the more time it will require to be transferred to the destination. Another advantage of ECC is that it has comparatively shorter keys than RSA for corresponding original file sizes.

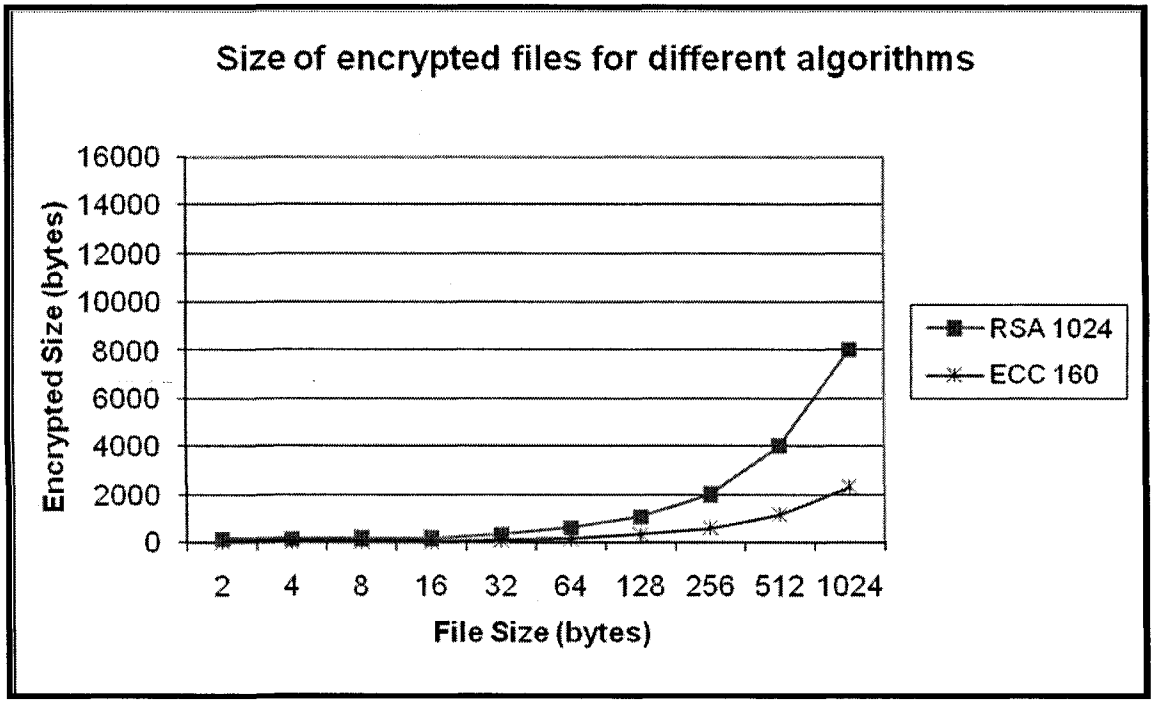

Figure 6-4: Size of the Encrypted file for ECC 160 and RSA 1024 Algorithms

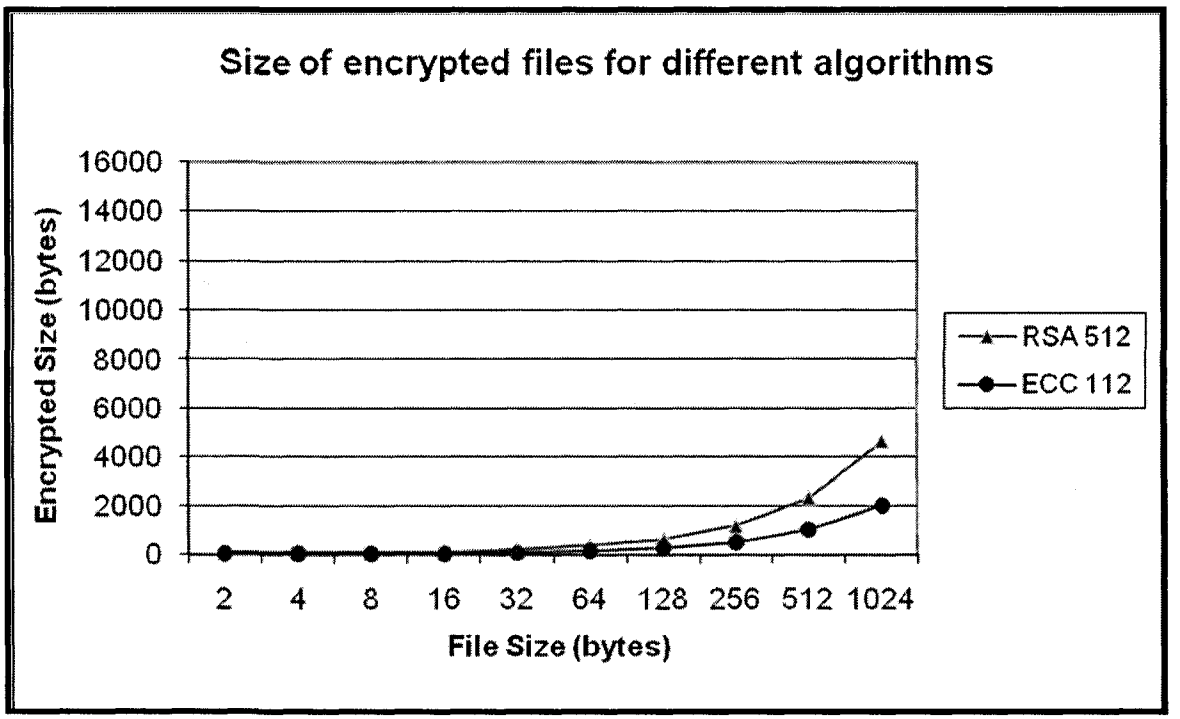

Figure 6-5: Size of the Encrypted file for ECC 112 and RSA 512 Algorithms

In the graphs (Figure 6-6, Figure 6-7, Figure 6-8 and Figure 6-9), the time taken by encryption schemes for a variable bit length is shown. As seen earlier, the encrypted file size increases with increasing bit length and this demonstrates the advantage of shorter key lengths.

Similarly, it is seen from Figure 6-6 that there is less time consumption with ECC in comparison to RSA, projecting the benefits of shorter key length with similar level of security. As the file size of the data to be encrypted increases, the time taken for the file to be encrypted also increases very rapidly for RSA. This shows that large key lengths of RSA is not suitable for large file sizes. However, no such behaviour was observed for ECC. Since ECC has comparatively shorter keys than RSA, it is expected that the file size increase would be gradual for ECC. The total time taken for encryption and decryption by any scheme significantly affects its performance since the requirement for any scheme is the high speed at which encryption/decryption is done.

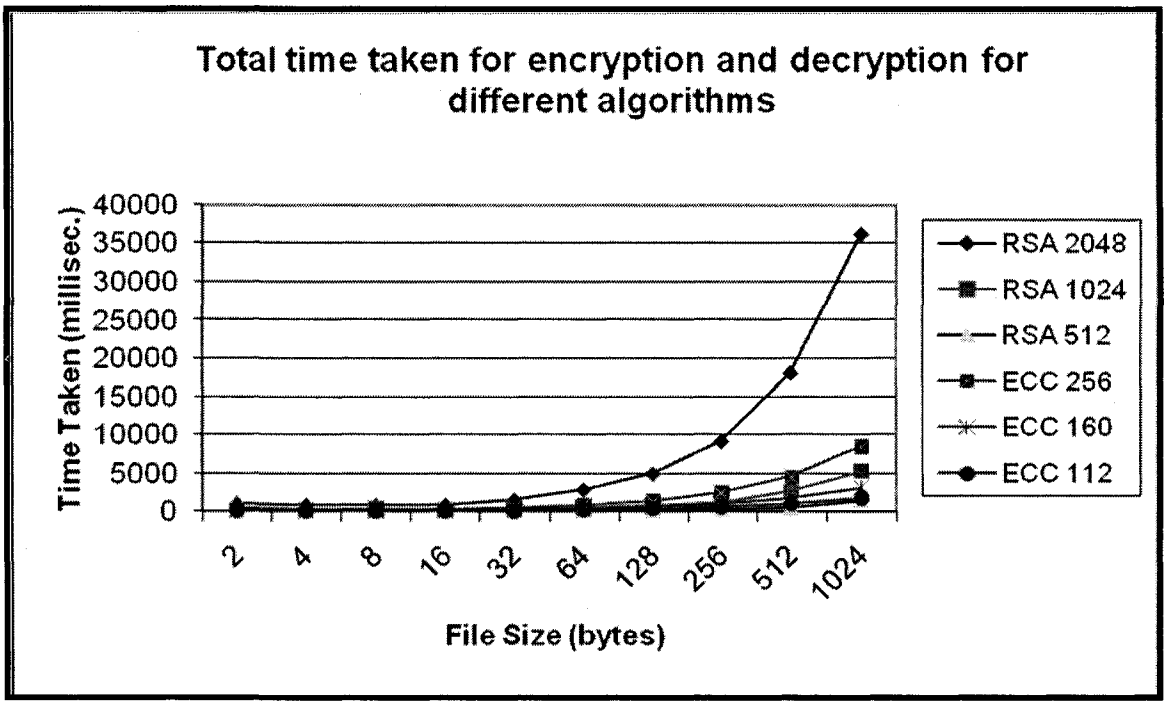

Figure 6-6: Total time taken for Encryption/Decryption for Different Algorithms

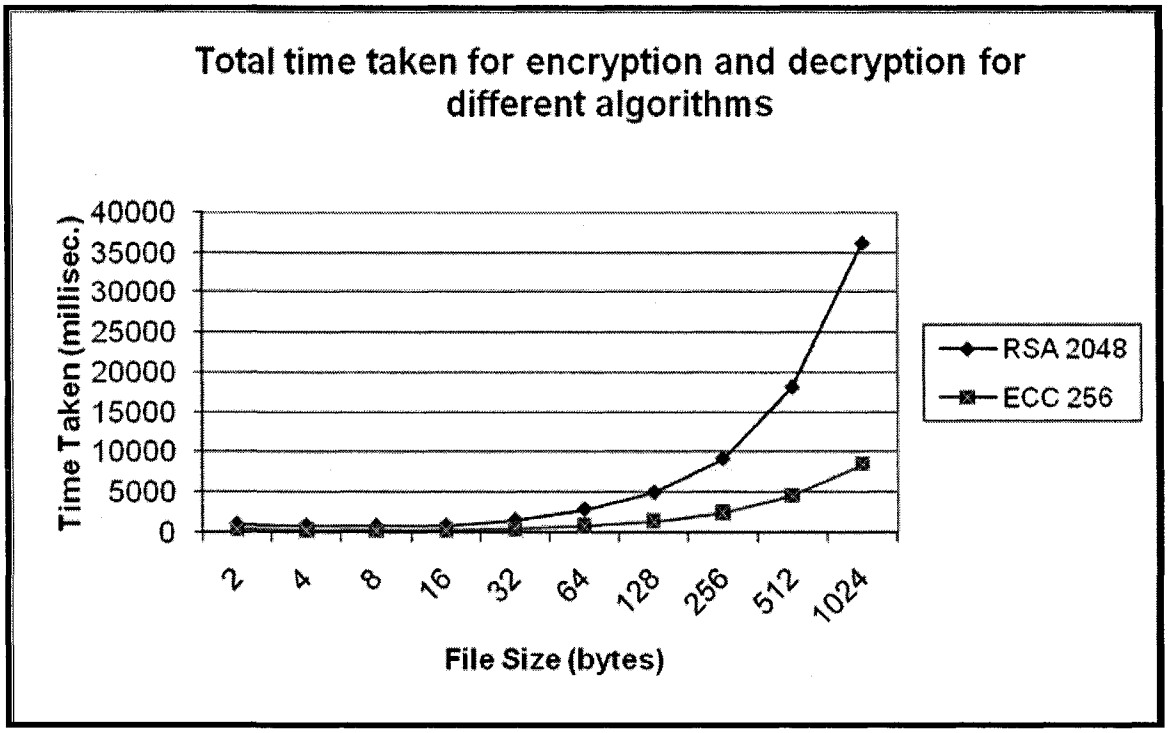

Figure 6-7: Total time taken by ECC 256 and RSA 2048 Algorithms

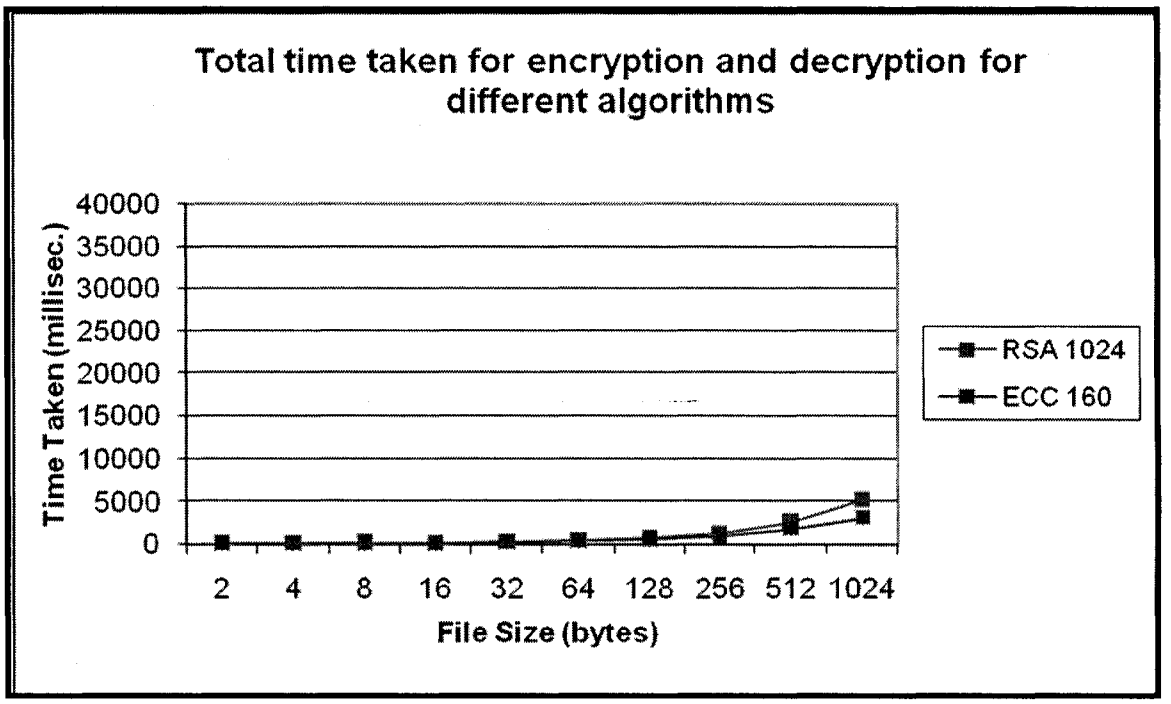

Figure 6-8: Total time taken by ECC 160 and RSA 1024 Algorithms

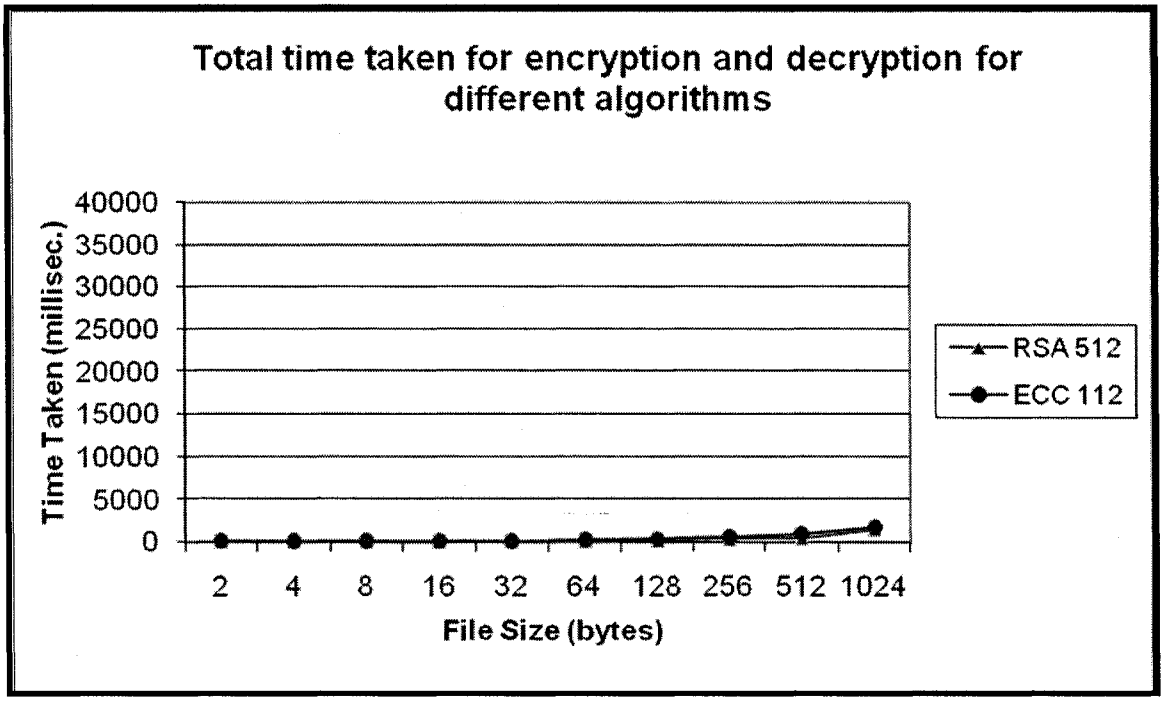

Figure 6-9: Total time taken by ECC 112 and RSA 512 Algorithms

From Figure 6-7, Figure 6-8 and Figure 6-9, it is observed that with the increasing file size of data to be encrypted also increases the time consumption. Bigger data files

generate more transport layer packets transferred over insecure channel and so the session would take longer to terminate for the communicating node in the network. The longer these communicating nodes remain online on an insecure channel, the more likely it is to be attacked. Conserving bandwidth is also an issue. The security technique must consider these issues and therefore, ECC seems to be an optimal choice. It is also seen in Figure 6-9 that for a smaller key size the performance of ECC is slightly better than that of RSA. For a smaller file size (e.g. 2 bytes), RSA takes less time than ECC but at around file size 32 bytes same time is taken by encryption and decryption for RSA and ECC.

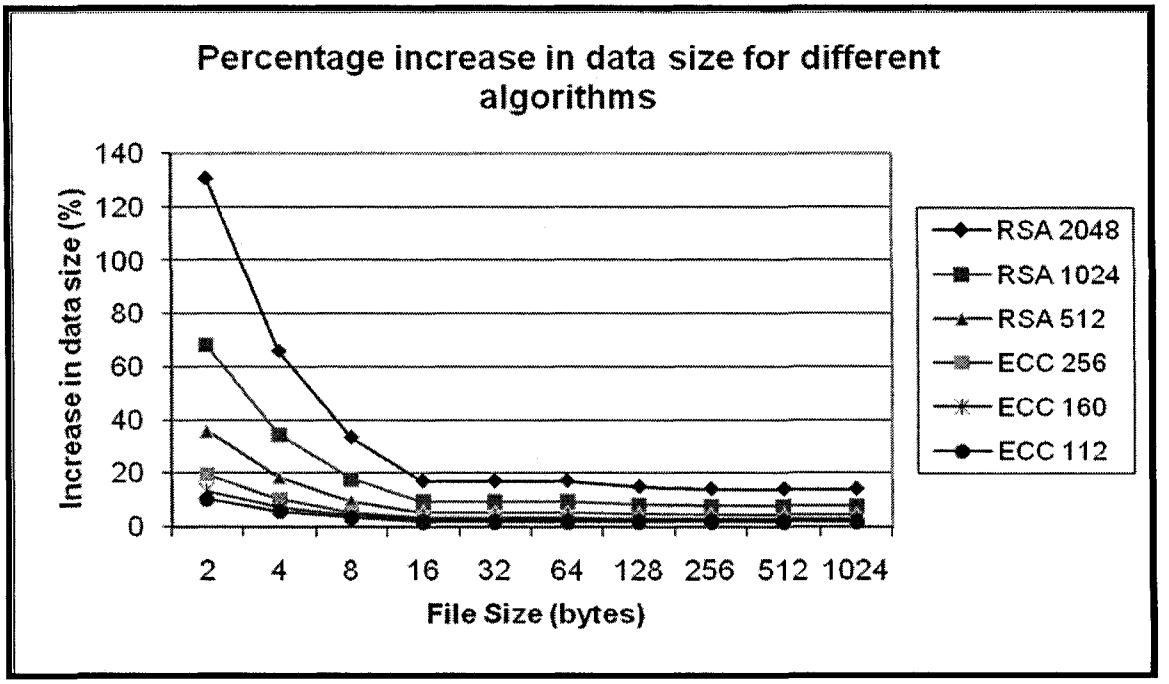

Figure 6-10: Percentage Increase in Data Size for Different Algorithms

In the graph (Figure 6-10, Figure 6-11, Figure 6-12 and Figure 6-13) below the increase in the percentage size of the encrypted file for different algorithms is shown. It is observed that although ECC generally performed much better than RSA, after a certain file size the percentage increase is stagnant. The reason for this is that with the increasing file size, the difference in size of original file and encrypted file in

comparison with the original file size become negligible. Since ECC has comparatively shorter keys than RSA, it was expected that the percentage increase in the data size should reduce with increasing file size for both protocols which can be confirmed from Figure 6-10. Another interesting observation is that the percentage increase in data size starts at a higher value for RSA than for equivalent ECC. However, with the increasing file size the difference is not so high between RSA and ECC. The reason for such behaviour could be the value of delta which defines the difference in size of the original file and the encrypted file. The difference between the delta values obtained from ECC and RSA is low for large file size as compared to small file size where the difference is really huge. The individual comparison between ECC and RSA key sizes can be seen from Figure 6-11, Figure 6-12 and Figure 6-13 which reveal similar trends.

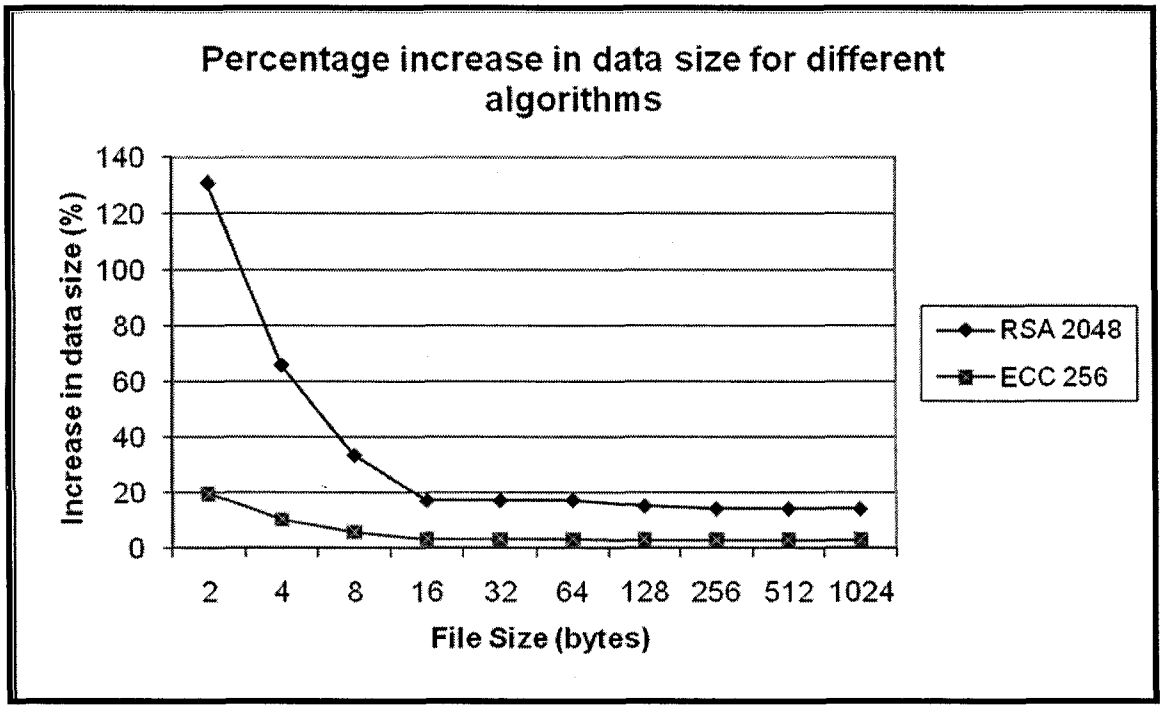

Figure 6-11: Percentage Increase in Data Size for ECC 256 and RSA 2048 Algorithms

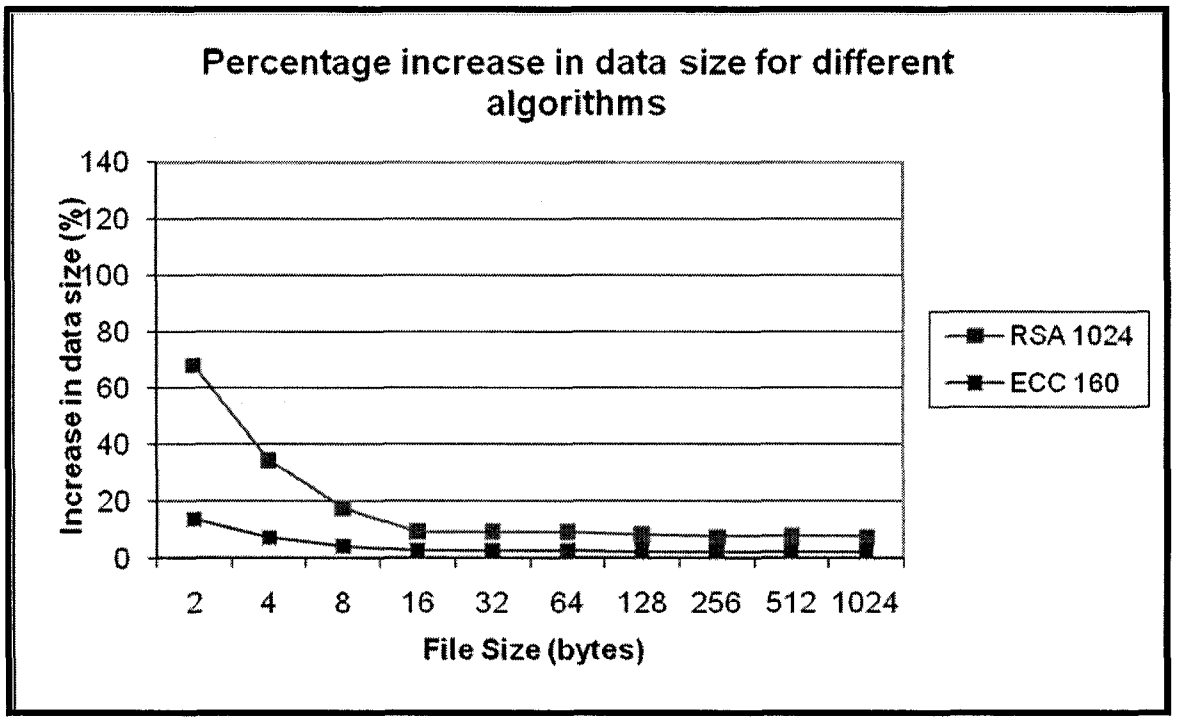

Figure 6-12: Percentage Increase in Data Size for ECC 160 and RSA 1024 Algorithms

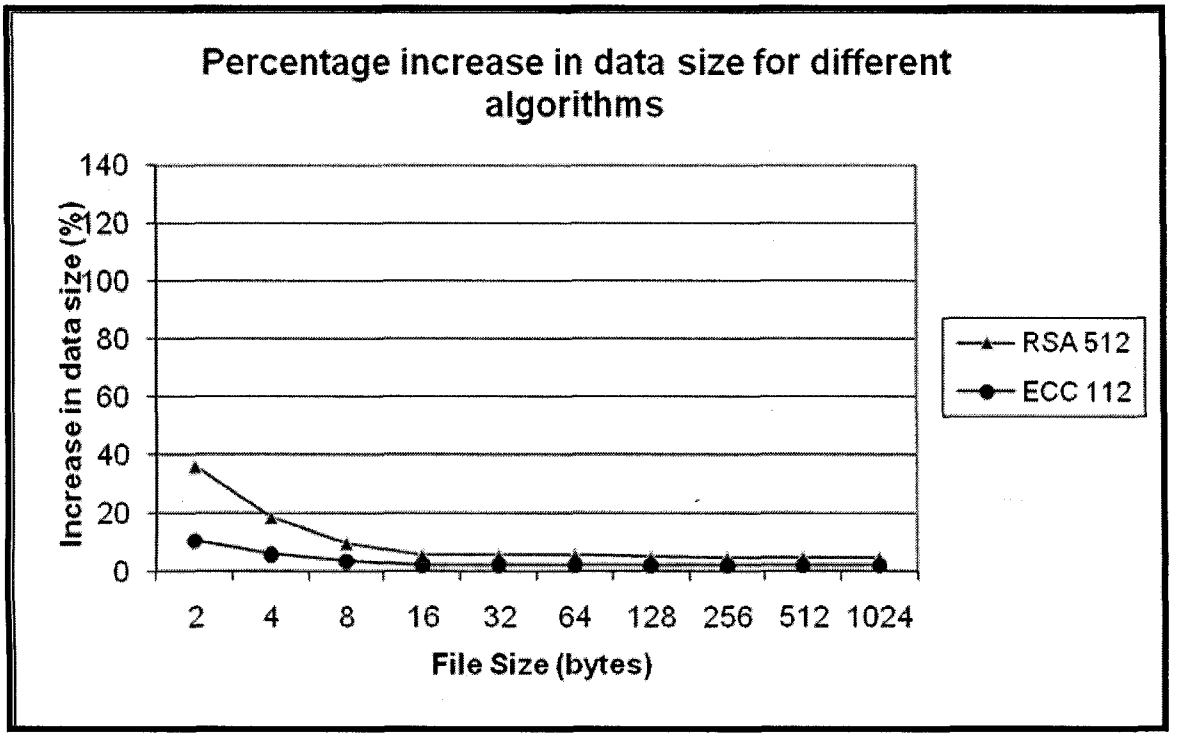

Figure 6-13: Percentage Increase in Data Size for ECC 122 and RSA 512 Algorithms

In the graph (Figure 6-14), the average encryption/decryption time taken by the individual encryption scheme is shown. The interesting observation from this graph is that the 2000 bits RSA which is equivalent to 256 bits ECC takes more time, approximately 4 times, to encrypt and decrypt a file. It is observed that ECC generally performed much better than RSA. From Figure 6-14, it is observed that the ECC is a better choice for a file that is larger in size than RSA because whenever any file is encrypted it increases the size approximately 10 times and using RSA would really strain the bandwidth when the file would be transferred. As a rule, the bigger the file is, the more are the packets generated from it. More packets take longer time to be delivered over the channel thus occupying greater bandwidth. This is one of the main concerns since the packet to be delivered to the recipient should be done quickly.

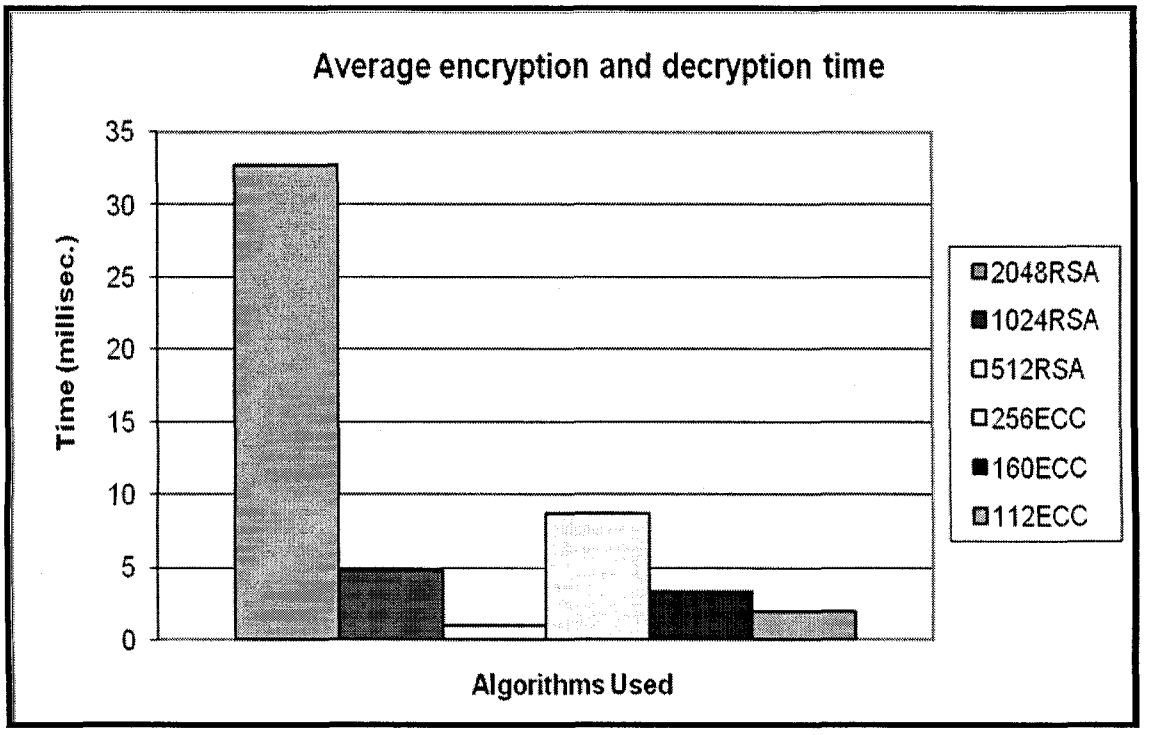

Figure 6-14: Average Encryption and Decryption time taken by RSA and ECC

Since ECC has comparatively shorter keys than RSA, it is the expected behaviour for the average encryption/decryption time to increase manifold for RSA. Further

examination of RSA patterns also reveals that shorter bit RSA key performs better than ECC with compromised security but ECC could not be analyzed because the standard key size set by NIST [NIS06] for ECC is minimum of 163 bits or else the security is compromised. Finally, it could be concluded that ECC would be an optimal choice for encryption/decryption requirements.

6.5. Results based on Computing Power of the Computer Systems

Experiments were conducted to test the effect on performance of different computing power of computer systems when different types of encryption are used. Simulation results obtained on a 2.4GHz Dell Laptop, on a Dell Pentium 4 computer system and on a Dell Dual-Core computer system, are presented.

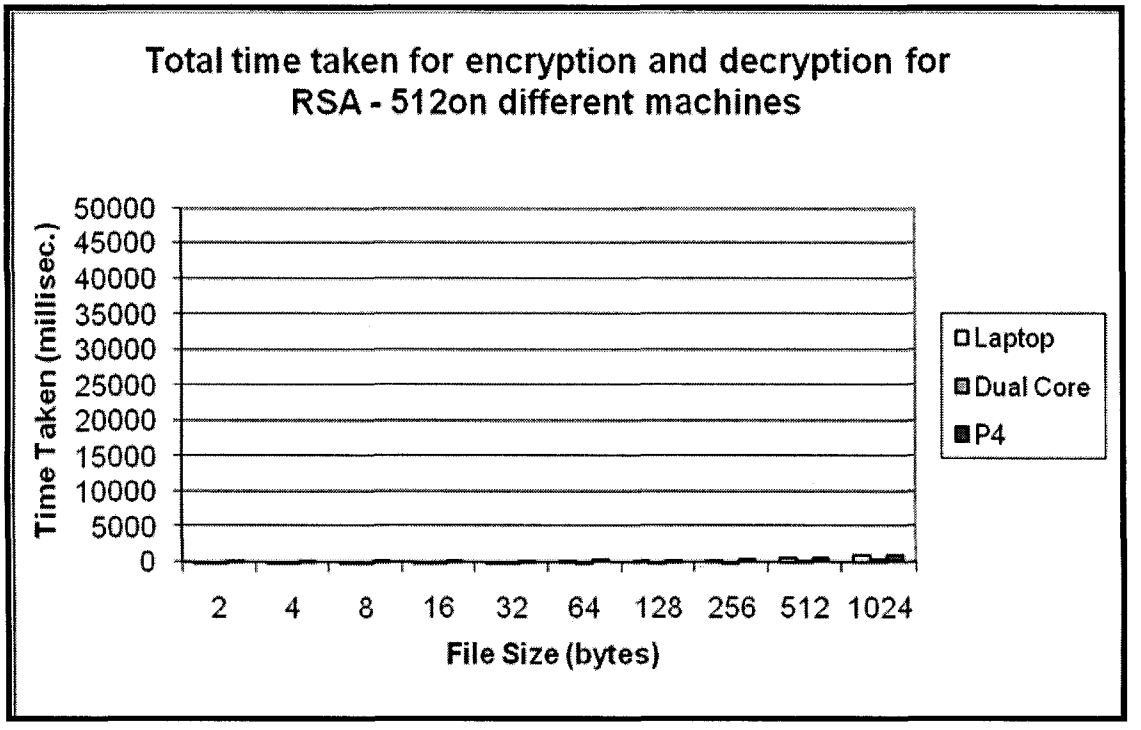

Figure 6-15: Total time taken for Encryption/Decryption for RSA 512 on Different Machines

The results are presented in Figure 6-15, Figure 6-17, Figure 6-18, Figure 6-19, Figure 6-20 and Figure 6-21. The graphs include the time in milliseconds on the y-axis and

variable original file sizes in bytes on the x-axis. In these graphs, the total time taken to perform encryption/decryption on different computer systems is presented.

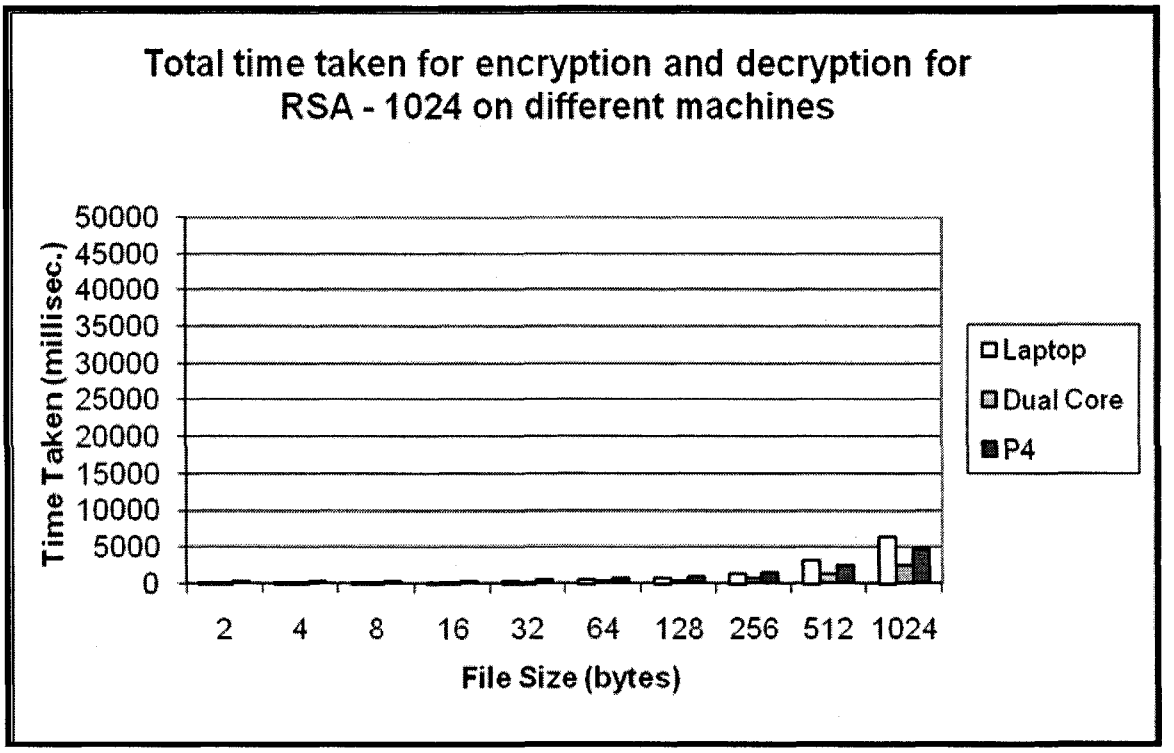

Figure 6-16: Total time taken for Encryption/Decryption for RSA 1024 on Different Machines

In Figure 6-15, it can be seen that the time taken for encryption/decryption for RSA rises sharply when the file size increases from 512 to 1024 bytes. For longer key lengths, it can be seen that the best performance is given by Dual Core processor and the worst performance is given by 2.4 GHz Laptop. Pentium 4 processor always gives an average performance. Also in term of time, the least time is taken by the Dual Core system. It has to be noted that the time taken for encryption/decryption varies for every graph. Similar results are observed in all the other graphs as well.

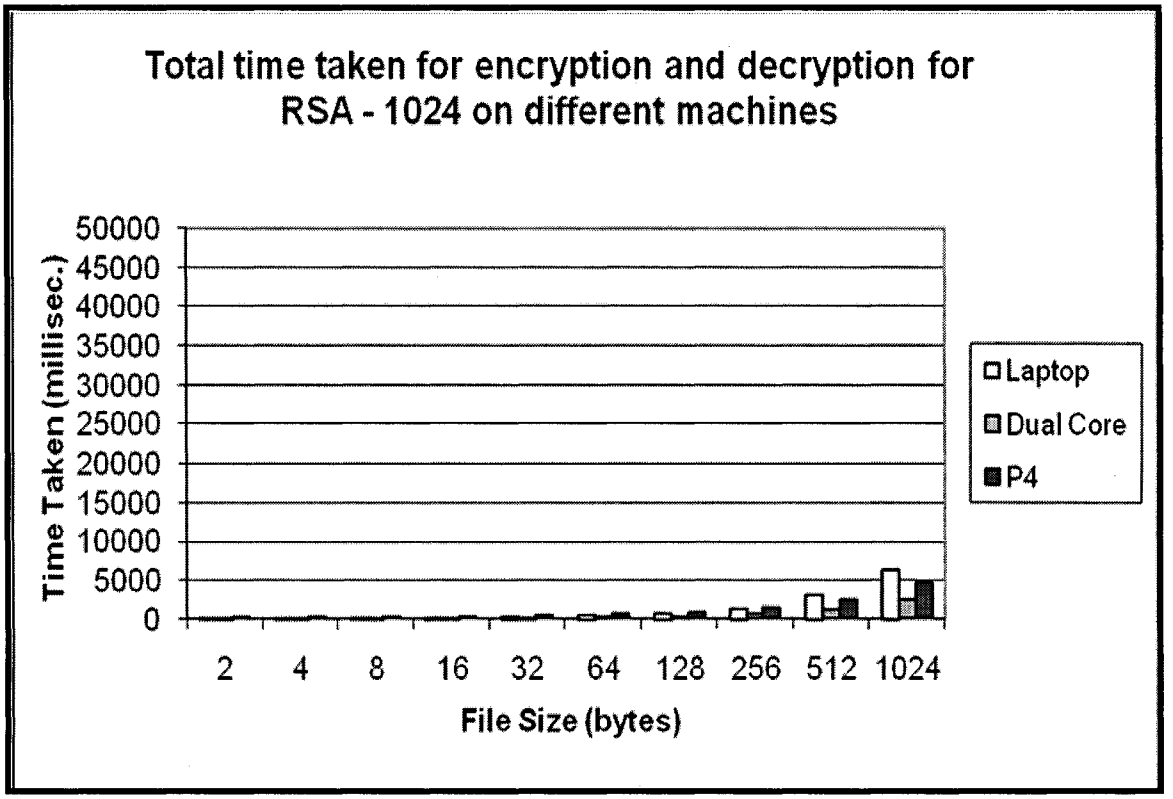

Figure 6-17: Total time taken for Encryption/Decryption for RSA 1024 on Different Machines

It was observed that Dual Core simulation performed comparatively much faster than the simulations running on other systems. In addition to that, since ECC has comparatively shorter keys than RSA, it is the expected behaviour for the Dual Core simulations using shorter key sizes for ECC consumed very less time in comparison to any key size for RSA. Further examination of the graphs reveals that the ECC performs better than RSA for equivalent key lengths on a Dual Core computer system. When graphs for ECC are observed, the worst performance is given by 2.4 GHz Laptop and the best by the Dual Core system.

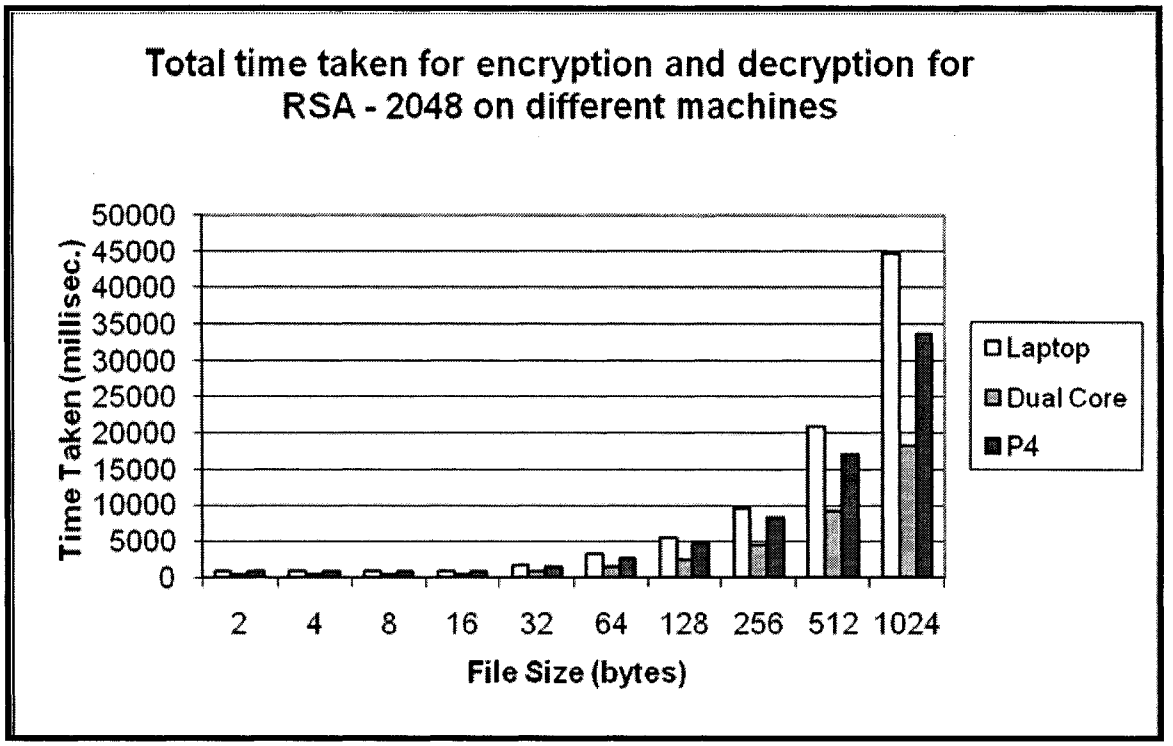

Figure 6-18: Total time taken for Encryption/Decryption for RSA 2048 on Different Machines

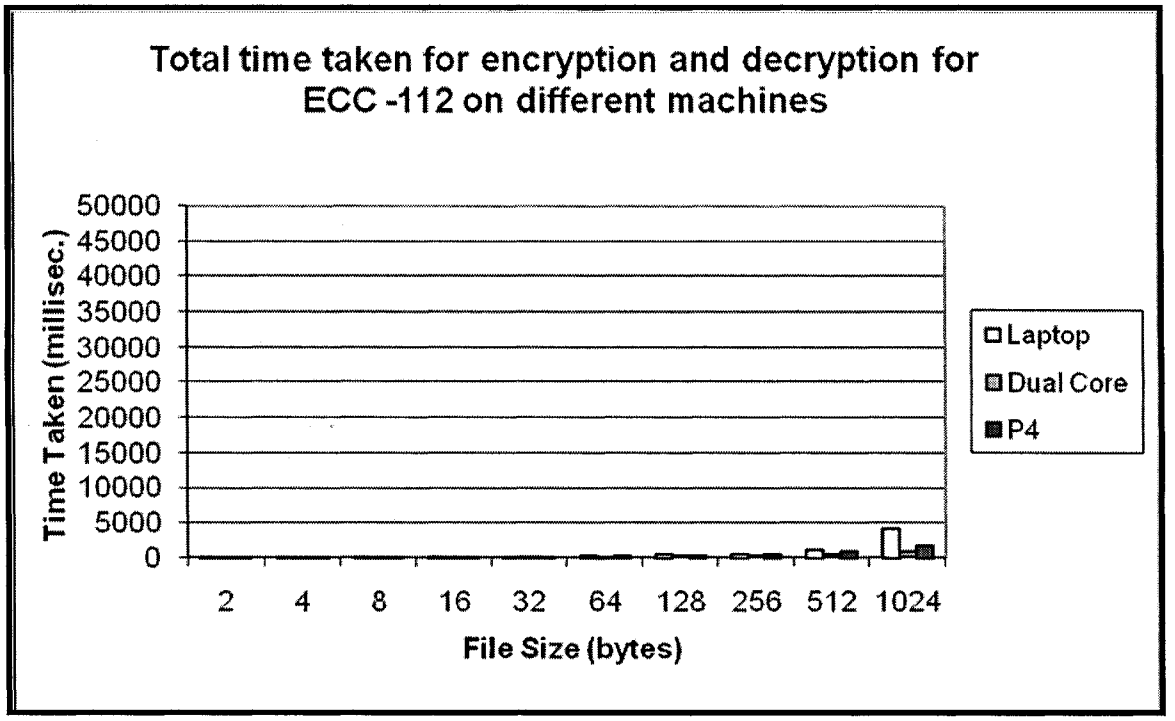

Figure 6-19: Total time taken for Encryption/Decryption for ECC 112 on Different Machines

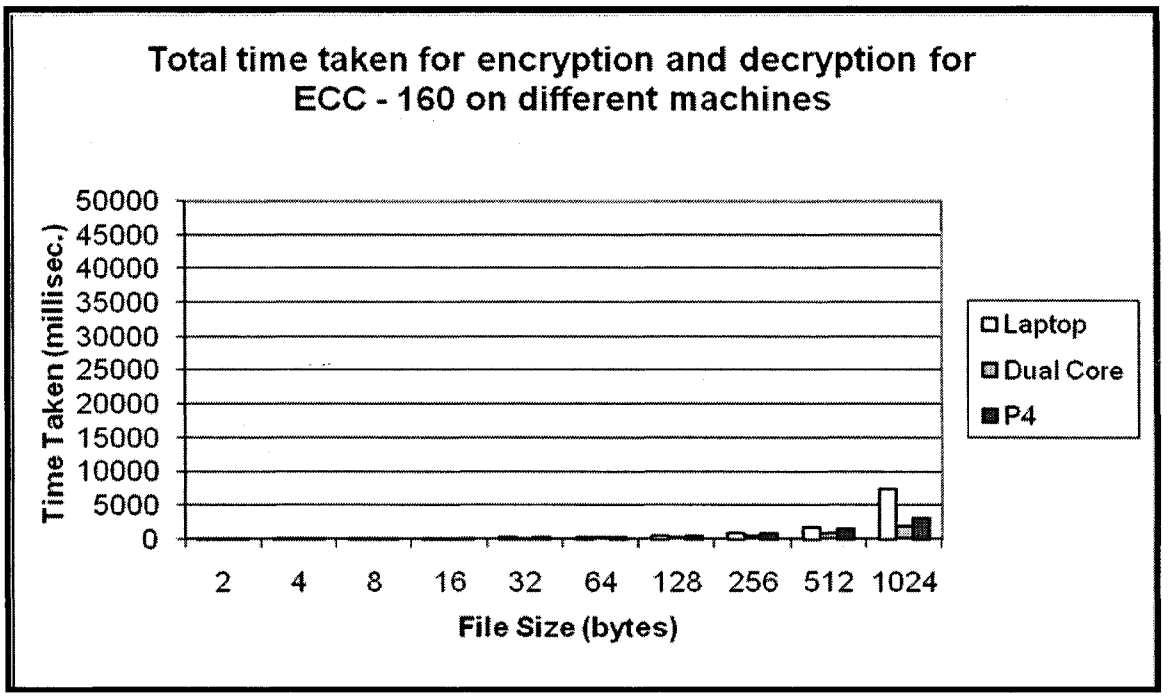

Figure 6-20: Total time taken for Encryption/Decryption for ECC 160 on Different Machines

In general, it is observed that the Dual Core simulation performed comparatively much faster than the simulations running on other systems. It was observed that the time taken by 2.4 GHz Laptop is the highest for all three key sizes that were compared in case of ECC. This means that the performance of 2.4 GHz Laptop is the worst since it takes more time to encrypt/decrypt the message. Also the performance of 2.4 GHz Laptop for 1024 bytes file size shows a sharp rise in the time consumption. In the case of 2.4 GHz Laptop, the time taken by ECC (112 bits and 160 bits) for 1024 byte file size is more than three times the value of time for 512 bytes. For ECC (256 bits) the performance of Pentium 4 machine is the worst followed by 2.4 GHz Laptop and the best performance is given by Dual Core processor. Due to the enhanced performance of a Dual Core processor, it was an expected behaviour but some major conclusions are made regarding the performance of RSA and ECC. For RSA, it is beneficial to use a Dual Core system for a smaller key size for an optimal performance of the scheme. However, for ECC, no such recommendation needs to be made about the key size to

be used. The usage of the key size for ECC is totally dependent on the requirement of the application. Further examination of the graphs reveals that the ECC performs better than RSA for equivalent key lengths on a Dual Core and Pentium 4 computer system.

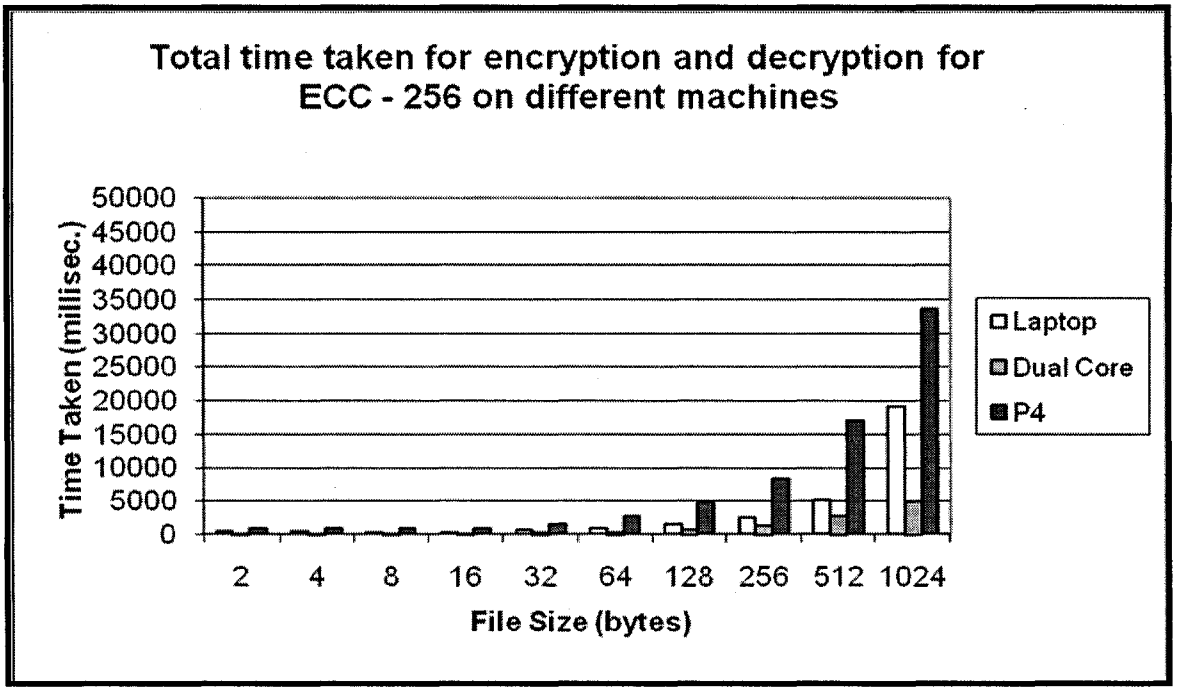

Figure 6-21: Total time taken for Encryption/Decryption for ECC 256 on Different Machines

6.6. Conclusions

Both schemes were compared on the basis of time taken to encrypt and decrypt a file of specific size (varying from 2 to 1024 bytes). Following conclusions can be made from the results.

1. Size of encrypted message and time taken to encrypt/decrypt increases proportionately with the key length.
2. For size of encrypted data, ECC file is almost quarter than that of RSA.
3. For time taken to encrypt/decrypt, ECC time is almost half than that of RSA time.

4. ECC takes longer than RSA for same bit length key. However, for equivalent number of bits, ECC performs better due to shorter key length.
5. It was also observed that for both techniques after certain key lengths the percentage increase in the data size becomes stable and does not increase anymore. However, the packet size is still small for ECC and is an advantage over RSA.
6. While comparing the performance of ECC and RSA on different computer systems, it was observed that Dual core system perform the best.
7. For ECC, higher key lengths could be used without any performance issues but for RSA, it is beneficial to use a high performance system such as Dual Core processor with a small key size for an optimal performance of the scheme.
8. ECC performs better than RSA for equivalent key lengths on a Dual Core and Pentium 4 computer system.

Chapter 7

Experiments with Elliptic Curve Cryptosystem on the Testbed

The performance of the proposed scheme and the improvements made in comparison to an existing secure protocol are discussed in this chapter. Since SAODV is a very well-known standard, it is the best choice to make the comparison between ECC and SAODV using different network sizes and data load. Since simulations are machine dependent, their comparison may not give accurate picture, but still provide a fair idea of performance. Simulations to establish the effect of dependence among the various factors affecting the performance accuracy are also evaluated.

7.1. Experimental Setup

The parameters which affect the performance of the network in the simulations were the number of nodes and their abilities, area of the network and the density of the network and the delay factor as described in Chapter 5. The experiment is performed using variable number of nodes namely 25, 50, 75 and 100. The results and the analyzes are presented in section 7.3.

7.2. Experimental Results

This section is divided in to three subsections based on the experiments performed on variable number of nodes (25, 50, 75 and 100). The results given below are based on different network sizes for different routing algorithms used for packet forwarding.

The data is analyzed for different encryption options for different key lengths. In Figure 7-1, Figure 7-2, Figure 7-3, Figure 7-4 and Figure 7-5, the relations of various evaluation parameters to AODV routing protocol are presented; in Figure 7-6, Figure 7-7, Figure 7-8, Figure 7-9 and Figure 7-10, similar relations using OLSR are given while Figure 7-11, Figure 7-12, Figure 7-13, Figure 7-14 and Figure 7-15 refer to LAR protocol. The figures have evaluation parameter on the y-axis and different encryption schemes on the x-axis. For comparison, equivalent key lengths are 160 bits of ECC and 1024 bits of RSA. The various parameters used for evaluation are:

- Delay is usually expected to have a lesser value as an indicator of better efficiency. Delay with the increase in size of encrypted file may increase the workload of the intermediate nodes and also increases the time taken by the message to be delivered from the sender to the destination.
- NRL is expected to have a lesser value. But NRL with the increase in size of encrypted file may increase the workload of the intermediate nodes and thus increasing the load value. This increase in load value increases the time taken by the message to be delivered from the sender to the destination and adversely affect the performance.
- PDF should be high normally since it is an indicator of how many packets are delivered to the destination. It is affected by factors such as network size. A decrease in PDF value is an indicator of bad network performance.
- Routing overhead should be low since it determines the burden routing messages has put on the network. An increased routing overhead indicates bad network performance.
- Average hop count usually increases the workload of the intermediate nodes and also increases the time taken by the message to be delivered from the sender to the destination. So, lesser hop count is indicative of better efficiency.

7.3. Experimental Data Analysis

The results obtained from the trace file of NS2 are further illustrated graphically. The graphs are described in relation to three different parameter streams: the evaluation parameter on the y-axis, the different encryption schemes on the x-axis and the variable number of nodes in the network. Five evaluation parameters are used for comparison, namely, delay, packet delivery fraction, average hop count, normalized routing load and routing overhead. Comparison experiments are done using no encryption, using ECC and then using RSA. Several key lengths are tested on 112, 160 and 256 bits for ECC and 512, 1024 and 2048 bits for RSA. The results are obtained in terms of evaluation parameters by keeping the other parameter fixed. Five graphs were obtained for every routing protocol. In Table 7-1, the scenario which was used to test proposed scheme using different routing algorithms is given.

Table 7-1: Performance Scenario

X	Y	Pause Time	Seed	Simulation Time	Traffic	Propagation	Nodes
1000	1000	0	10	500	CBR	Two-ray	25/50/75/100

7.3.1. AODV with Variable Network Size– Results

From the results it can be seen that with higher bit length a smaller network produces more delay (Figure 7-1) while a larger network with higher bit length produces relatively less delay. Another observation is that a network with no encryption has less delay as compared to a network with encryption. When equivalent key sizes 160 bits ECC and 1024 bits RSA are compared their delay give same result. However for 2084 bits RSA, the delays is almost double that for equivalent ECC. It can be concluded that ECC produces much less delay in comparison to RSA.

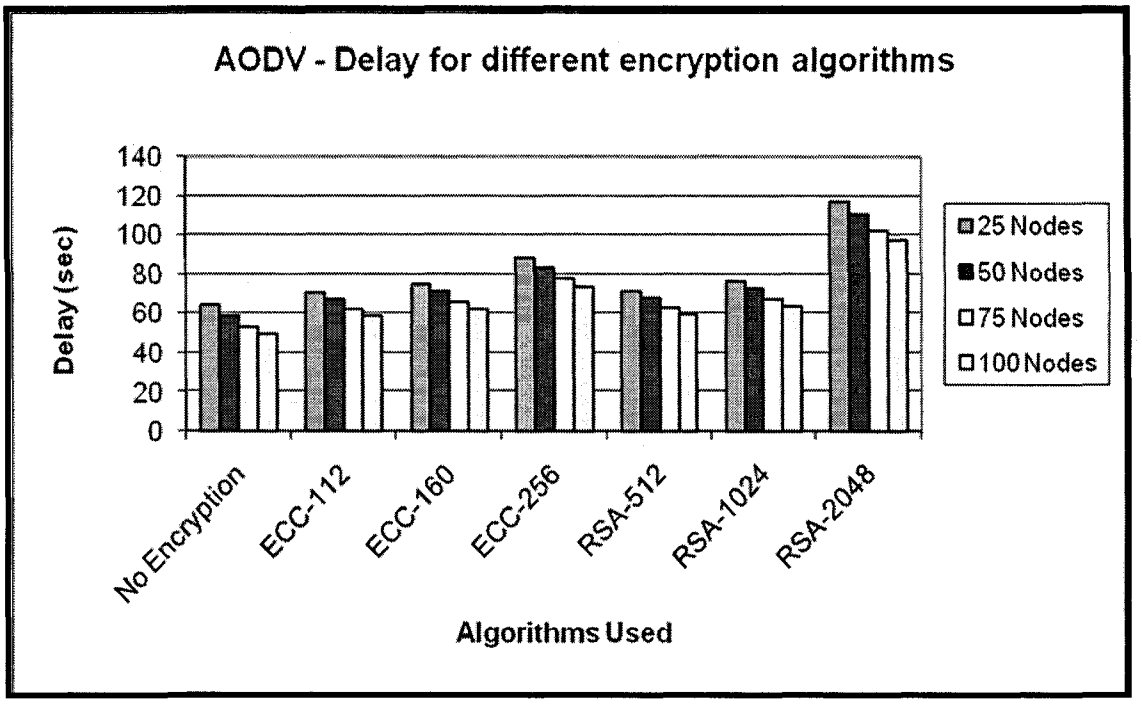

Figure 7-1: AODV-Delay for Different Encryption Algorithms

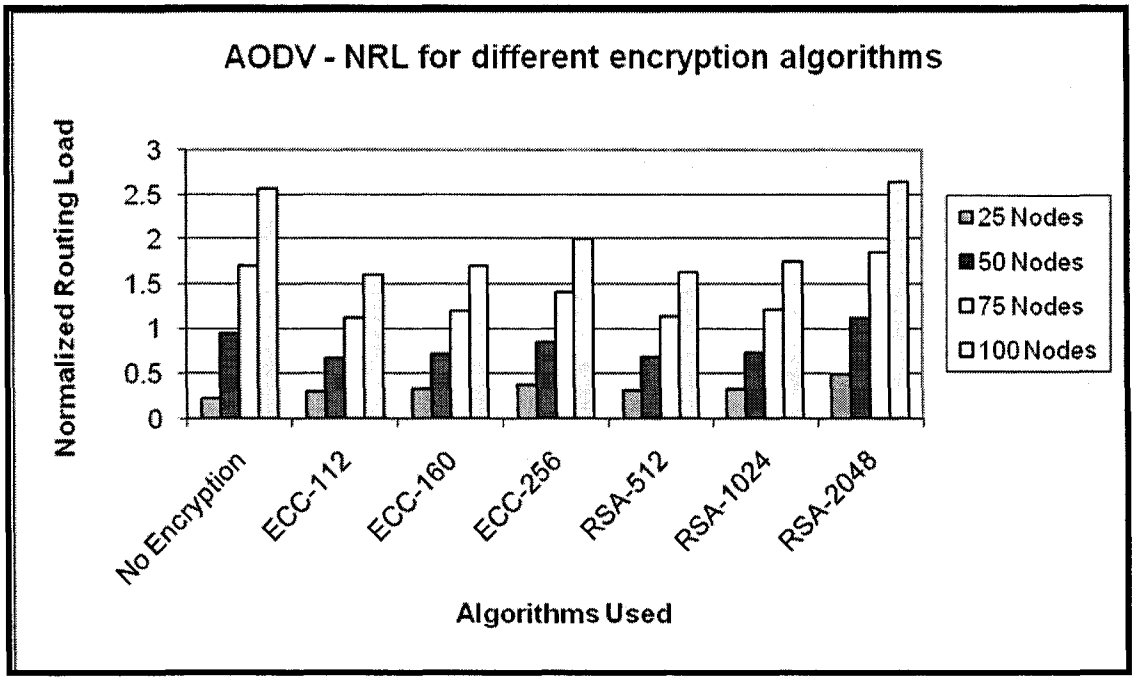

Figure 7-2: AODV-NRL for Different Encryption Algorithms

NRL (Figure 7-2) decreases when encryption scheme is used. However, with larger key sizes of RSA, NRL jumps back to a higher value for a denser network. However,

for an equivalent size, ECC gives a slightly lower load value than that of RSA. For 2048 bits RSA, the NRL has a very high value which indicates towards the trend that increasing key lengths will increase routing load as well. It is to be noted that for both encryption schemes ECC and RSA the load increases with increasing key lengths. Also, denser networks produce higher value of load than smaller less dense network.

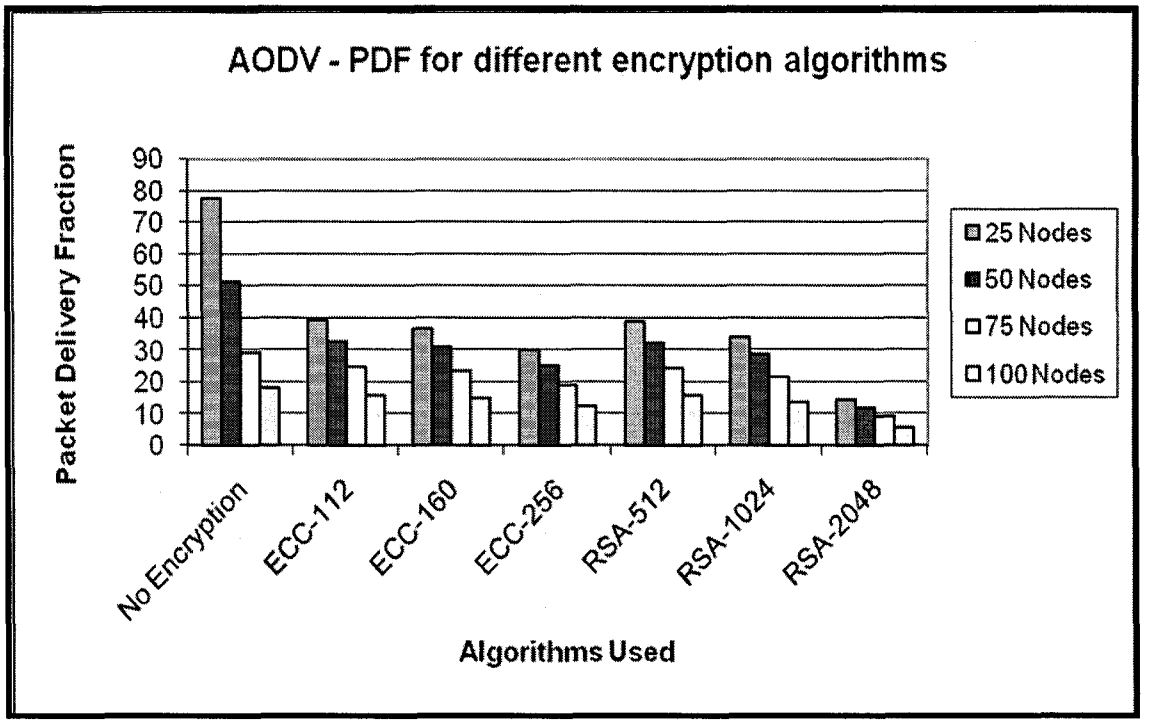

Figure 7-3: AODV-PDF for Different Encryption Algorithms

In Figure 7-3, the PDF is the highest when no encryption is used in a smaller network but it reduces for larger networks. When encryption is applied, the PDF decreases with the increasing key lengths of encryption scheme due to increasing size of encrypted packets. The fraction drops with increasing key sizes of the encryption scheme. However, for an equivalent size, ECC has a higher packet delivery fraction value than RSA. It is to be noted that for both encryption schemes ECC and RSA the load increases with increasing key lengths. Also, denser network produce slightly less fraction value of load than smaller, less dense network.

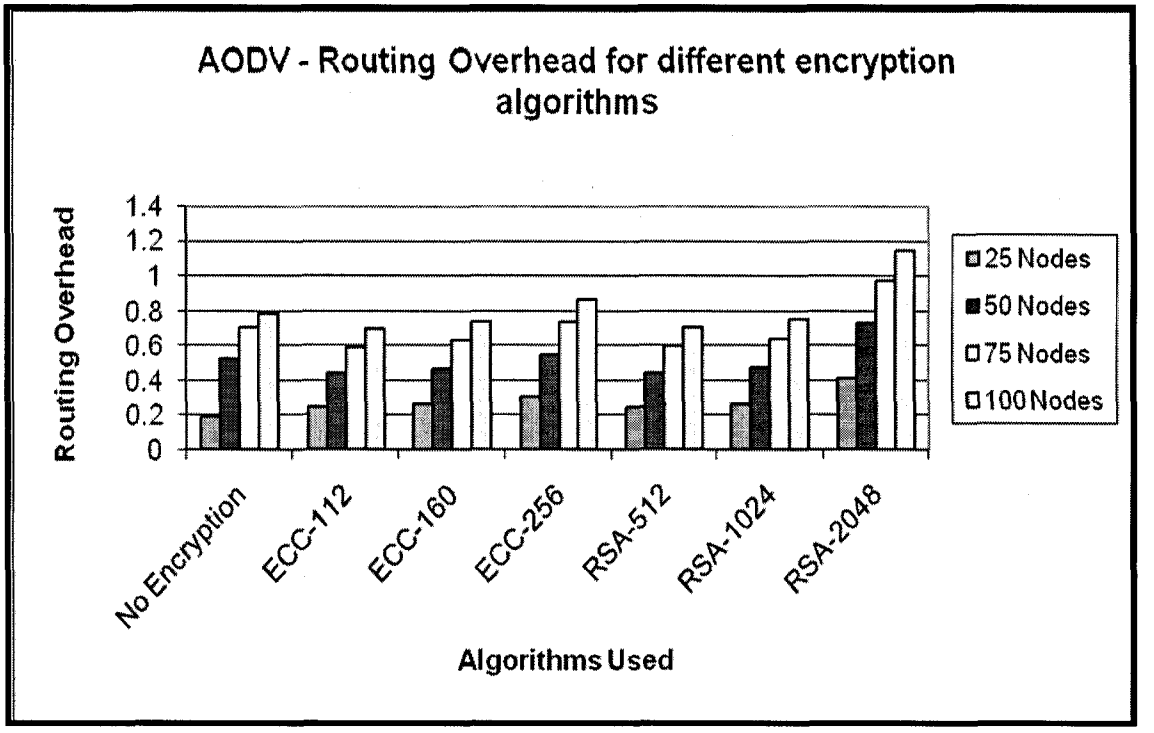

Figure 7-4: AODV-Routing Overhead for Different Encryption Algorithms

In Figure 7-4, the relation between the routing overhead and the encryption schemes for variable network size is shown. Ideally when encryption is applied, the routing overhead should increase due to more delays resulting in packet drops. However, there may not have enough packets to saturate the network in this simulation and hence the results show no appreciable change in routing overhead between encrypted and original files. Routing overhead increases with the increasing key size of encryption scheme. However, for an equivalent size, ECC fares same as RSA. It is to be noted that for encryption schemes ECC and RSA the overhead increases with increasing key lengths.

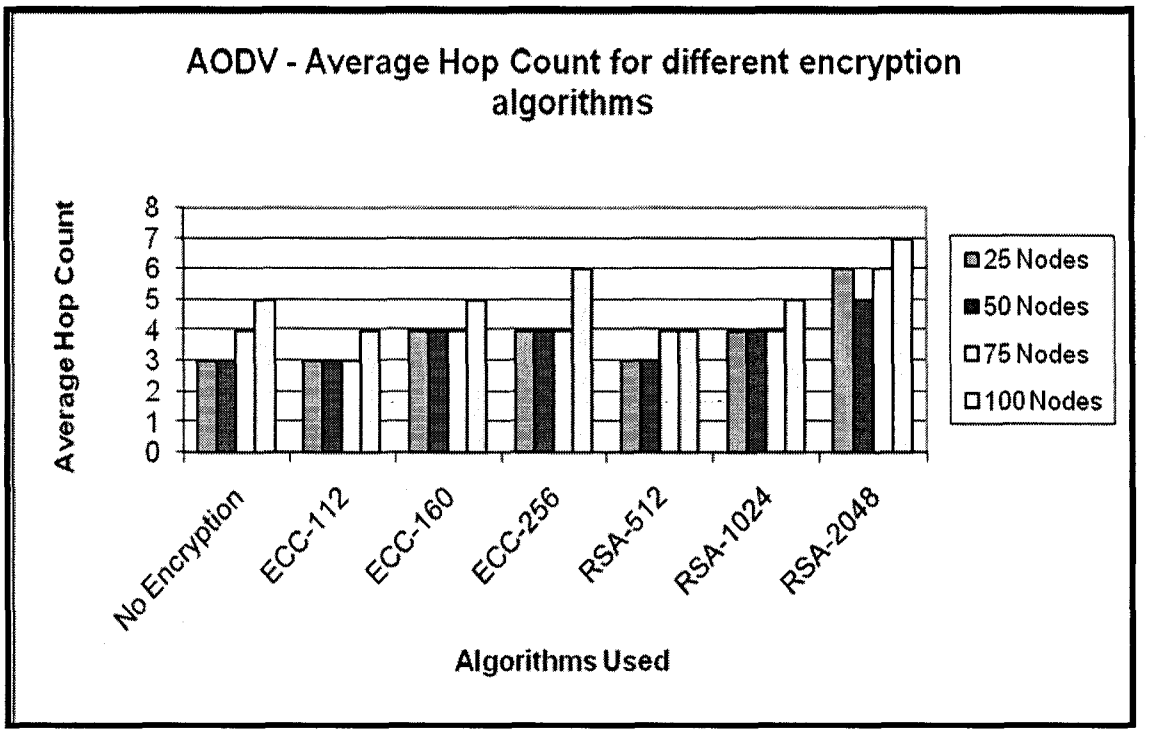

Figure 7-5: AODV-Average Hop Counts for Different Encryption Algorithms

In Figure 7-5, the relation between average numbers of hops made by the message before it is finally delivered to the destination node and the use of different encryption schemes is shown. It can be seen that hop count increases for larger key sizes probably due to the presence of more data packets. When equivalent key sizes for different encryption schemes are compared, it is found that ECC has the same average hop count as RSA. But for 2048 bits RSA, hop count increases sharply. However no such trend is seen in ECC. Hence, the performance of ECC is better than that of RSA. The results also show that a denser network has fairly high hop count than smaller, less dense network. It is also noted that for ECC, the hop count reduces with larger key sizes but for RSA it increases with increasing key sizes.

7.3.2. OLSR with Variable Network Size- Results

In Figure 7-6, it can be seen that delay increases with increasing key length for both encryption schemes. The delay is bound to increase when encryption is applied for

both schemes due to the presence of more packets. However, this increase for RSA is slightly higher than that for ECC. Further, for higher bit length, the delay for RSA shows higher value as compared to ECC. It was also noticed that less dense networks have higher delay values than denser networks because dense networks has more neighbours, therefore, more routes and links for information exchange. Since larger key lengths offer increased complexity, the delay is bound to increase. When equivalent key sizes for RSA and ECC are compared, the delay for RSA is higher than that for equivalent ECC. It can be concluded that ECC produces less delay in comparison to RSA.

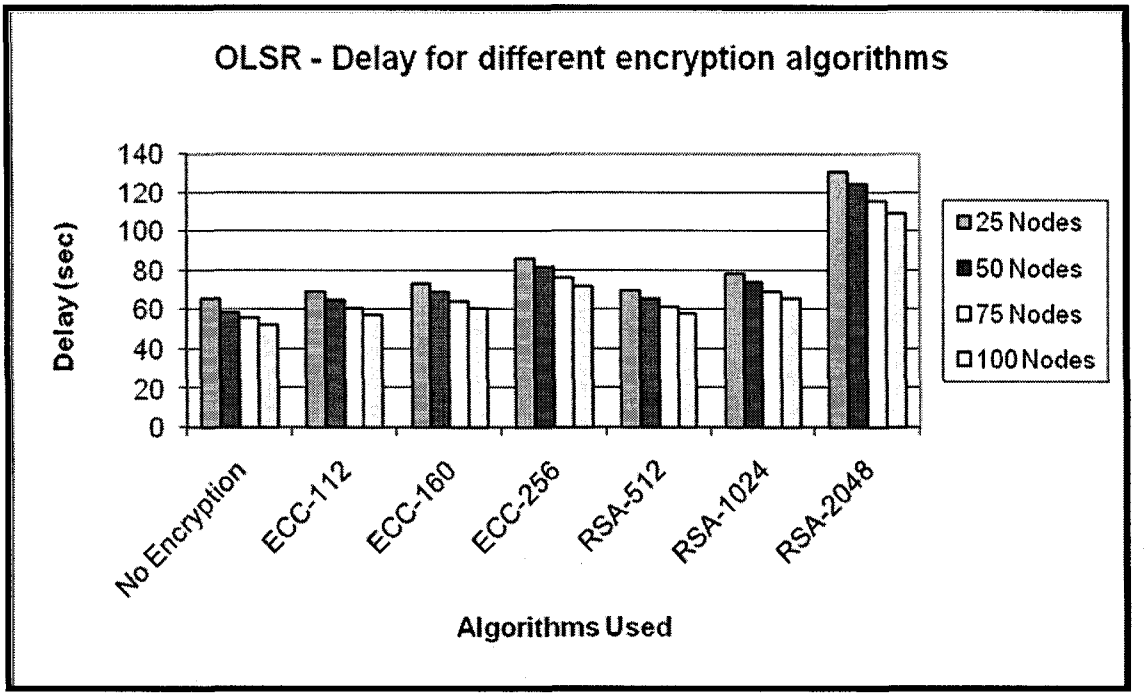

Figure 7-6: OLSR-Delay for Different Encryption Algorithms

In Figure 7-7, the results show that routing load for network increases gradually with increasing key lengths. It is an expected behaviour for the load to increase with increasing key lengths for ECC and RSA encryption schemes. Unlike ECC, the larger key size (2048 bits) for RSA showed a sharp increase in load value. The graphs showed a gradually increasing load value for increasing network size. However, when

comparing equivalent key lengths, ECC showed a much better performance than RSA.

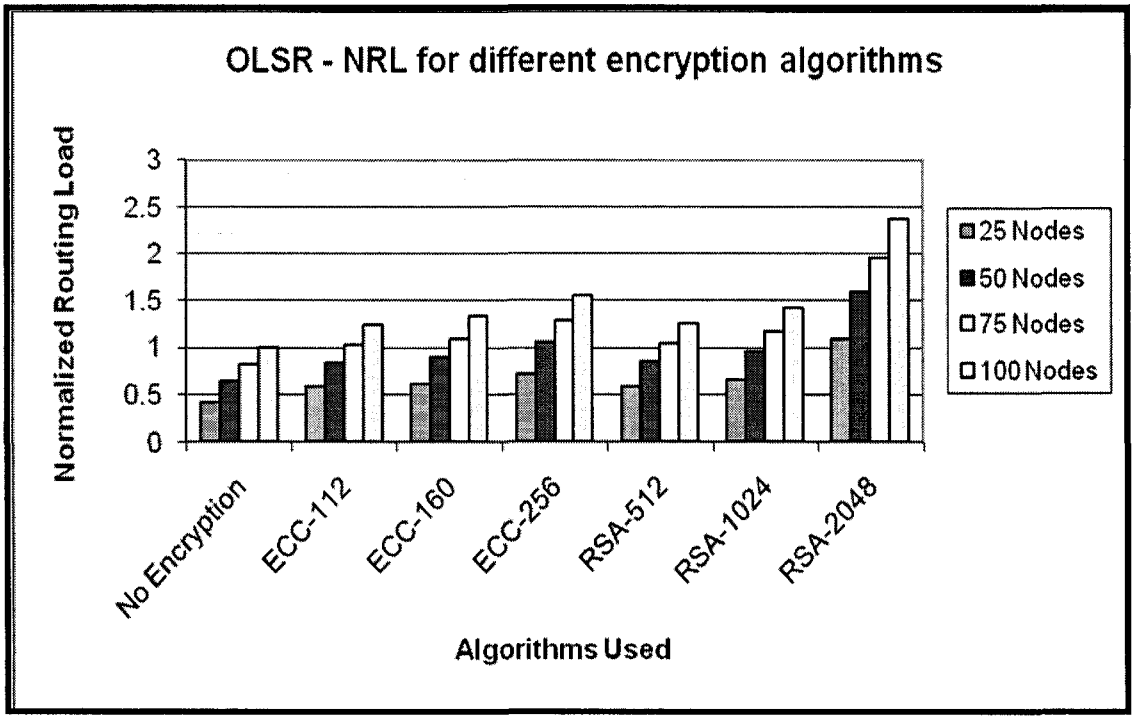

Figure 7-7: OLSR-NRL for Different Encryption Algorithms

In Figure 7-8, the results show that PDF is very high when no encryption is used. PDF indicates the probability of packets being delivered to the receiver. The higher is the PDF, the better is the performance. This fraction is larger when no encryption is used in a smaller network. This fraction decreases with the higher key lengths of encryption scheme. This is so because with higher key lengths the delay and the probability of packet drop increases due to the large size of encrypted file. PDF decreases with increasing network size as well as increasing key lengths in the case of both ECC and RSA encryption schemes. However, for an equivalent size, RSA has a smaller PDF than ECC.

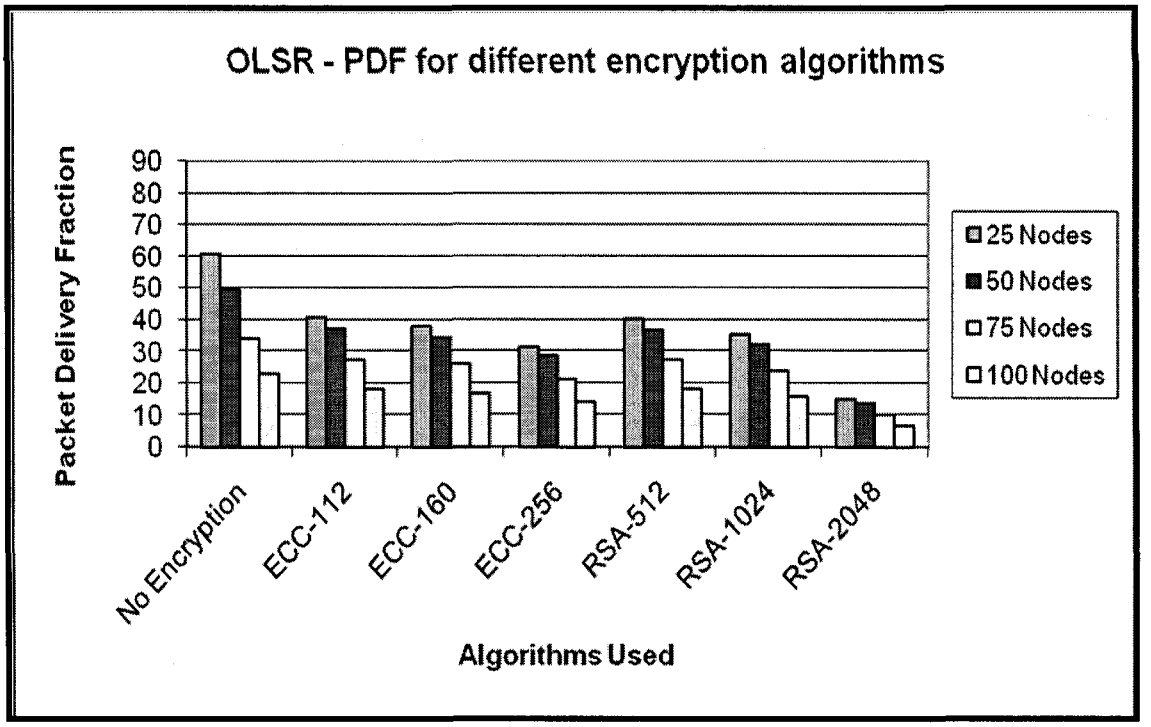

Figure 7-8: OLSR-PDF for Different Encryption Algorithms

In Figure 7-9, the overhead increases gradually with the increase in network size and key size for both ECC and RSA encryption schemes. When encryption is applied, there is only a slight change in the overhead value. For an equivalent key length of ECC, it was also observed that the routing overhead of RSA increases. It was also seen that increasingly dense network showed relative increase in routing overhead.

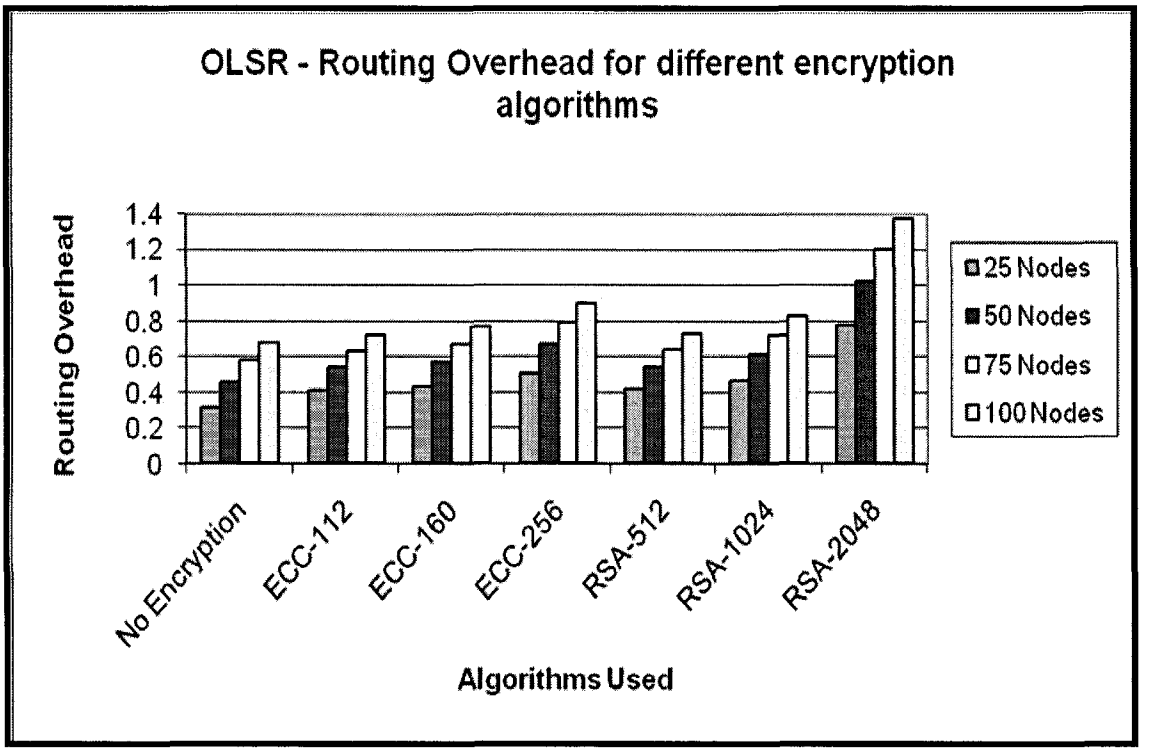

Figure 7-9: OLSR-Routing Overhead for Different Encryption Algorithms

In Figure 7-10, the results for average hop count show an increase for higher key sizes. When comparing network with encryption and without encryption, it was found that the hop count remains almost the same except for highest key lengths. This may be probably due to the presence of more data packets at higher key lengths. The network size does not seem to influence the hop count. When equivalent key sizes for different encryption schemes are compared, it is found that ECC has less average hop count that RSA for all network sizes. The value of RSA average hop count increases sharply and becomes almost double for 2048 bits key lengths. Therefore, it could be said that the performance of ECC is better than of RSA.

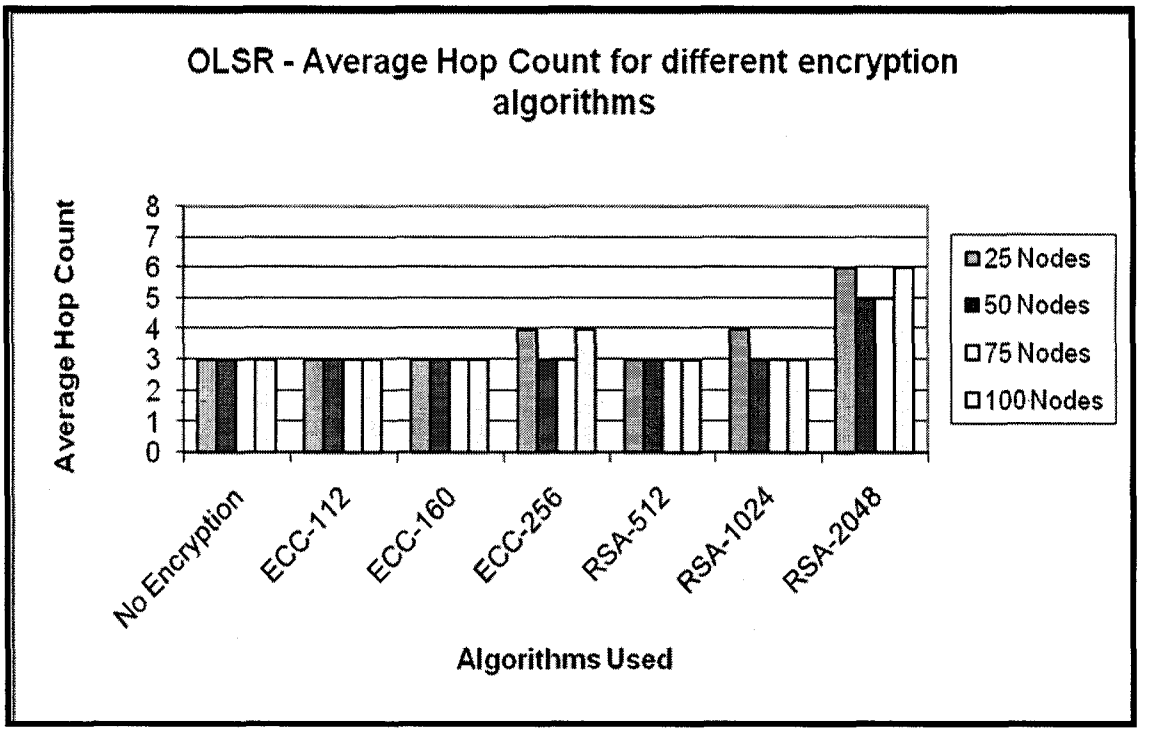

Figure 7-10: OLSR-Average Hop Counts for Different Encryption Algorithms

7.3.3. LAR with Variable Network Size – Results

Experiments were done using LAR protocol and the results show that higher key length produces more delay (Figure 7-11). This is an expected behaviour for any network since the key size increase the transmission time manifolds. It was also seen that delay for a 25 node network is slightly higher than a 100 node network. A network with no encryption has less delay than a network with encryption. When equivalent key sizes are compared, it was observed that ECC has a better performance than RSA. Therefore, it could be concluded that ECC is a better choice for encryption at least in terms of delay.

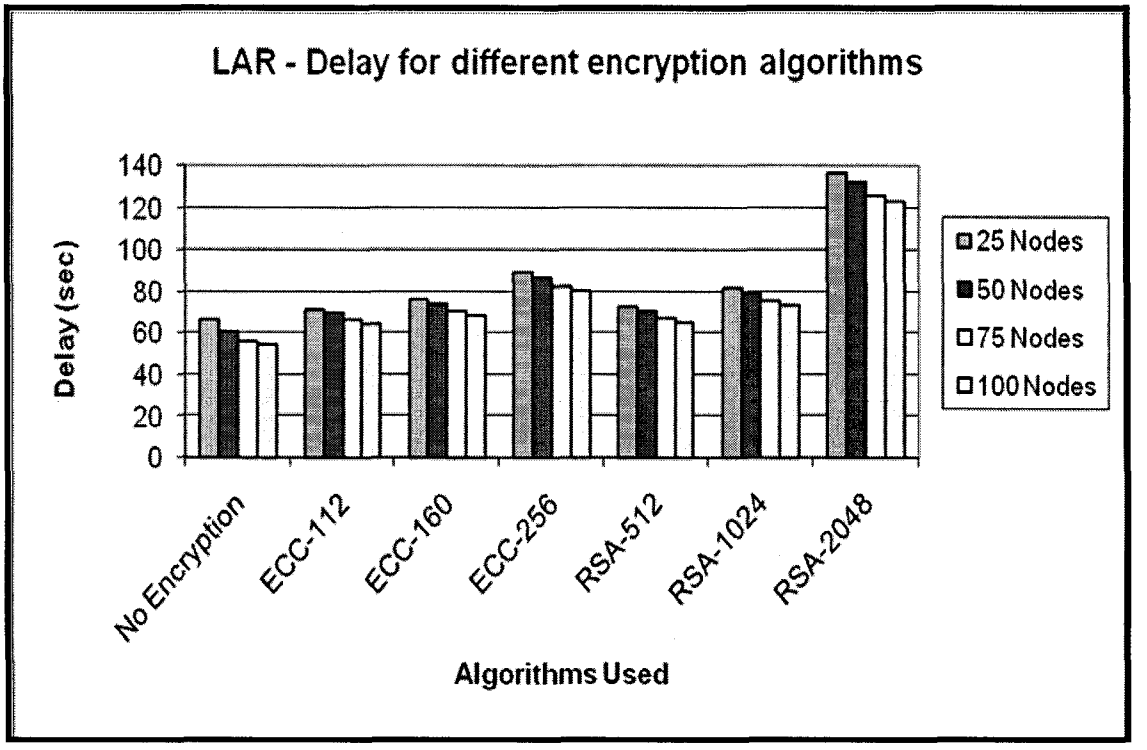

Figure 7-11: LAR-Delay for Different Encryption Algorithms

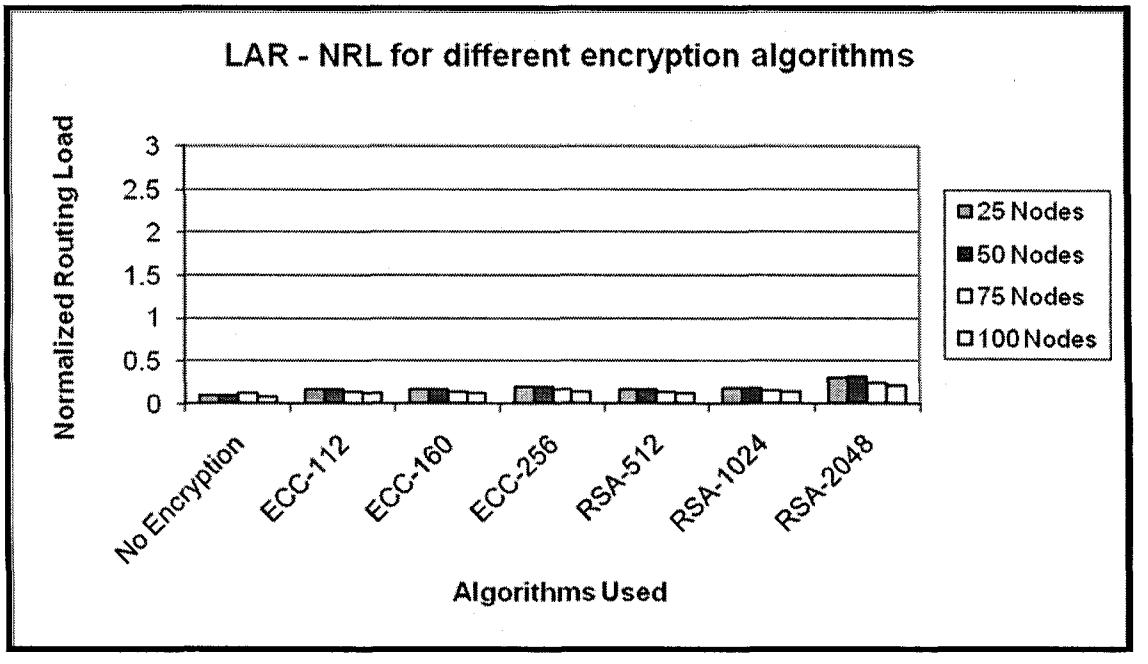

Figure 7-12: LAR-NRL for Different Encryption Algorithms

In Figure 7-12, NRL for network with no encryption and with different types of encryption are shown. It was seen that a network with no encryption showed a less

value of load than in a network with encryption. The first observation is that with larger key lengths, ECC showed a slight increase in load value than RSA. The second observation is that while the load increases gradually with increasing key sizes for ECC, the load increase was sharp in the case of RSA. The third observation is that when equivalent key sizes are compared for ECC and RSA, ECC has a lower load value than RSA. It was also observed that NRL reduces with increasing density of the network. This may be due to the presence of neighbouring node to share the routing overhead.

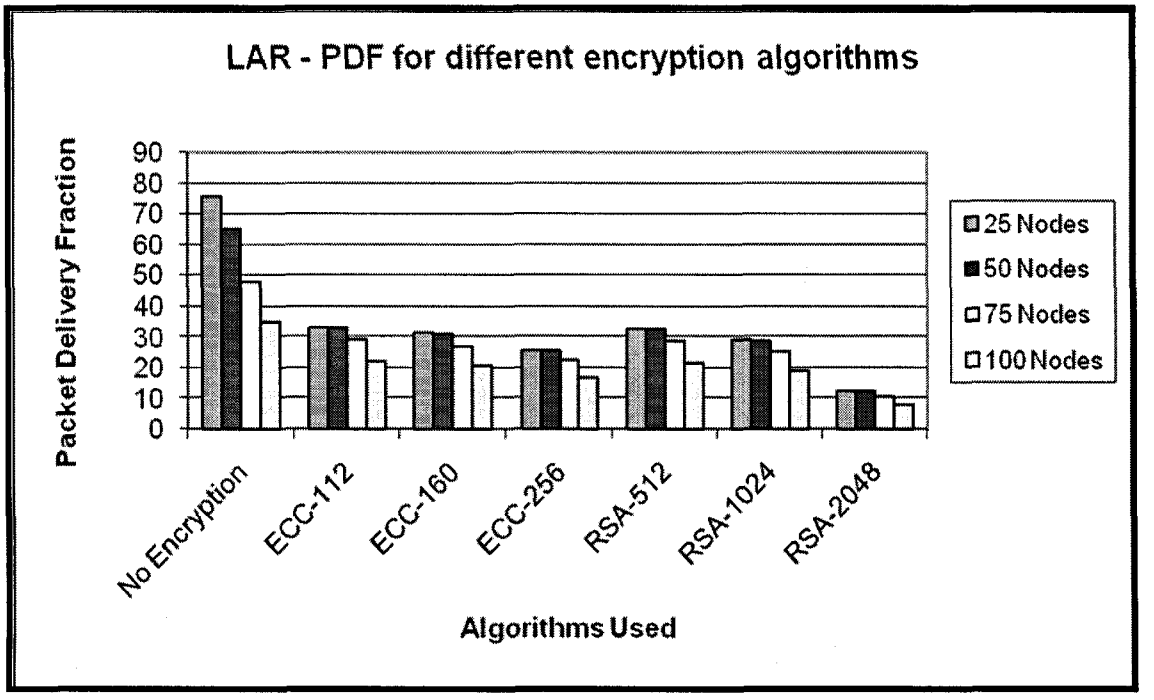

Figure 7-13: LAR-PDF for Different Encryption Algorithms

It is seen in Figure 7-13 that networks with no encryption have higher value while networks with encryption have lower value. This behaviour is expected due to the overhead caused by encryption. PDF decreases with higher key lengths of encryption scheme. PDF decreases sharply for a higher key length of RSA which shows the bad performance of RSA. Also a denser network has a less PDF as compared to a less dense

network. However, for an equivalent key size, ECC has a higher PDF value than that of RSA which shows that ECC is a more reliable encryption scheme.

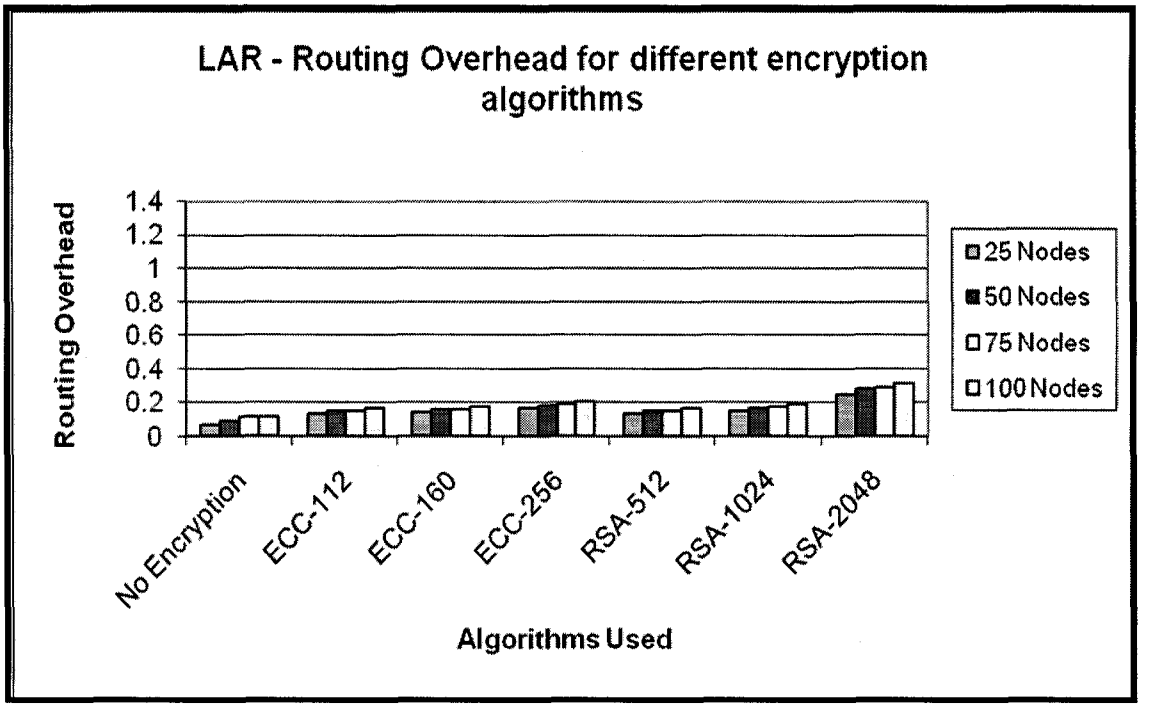

Figure 7-14: LAR-Routing Overhead for Different Encryption Algorithms

In Figure 7-14, the relation between routing overhead and different encryption schemes for a variable network size shows that increasing key sizes and increasing network size increases the overhead. The network using encryption does show an increase in overhead with increasing key size of encryption scheme. It is also that a 25 node network has a slightly lower overhead than a 100 node network. Also when equivalent key sizes for RSA and ECC are compared, RSA showed a higher overhead than ECC. Thus it could be said that ECC fares much better than RSA.

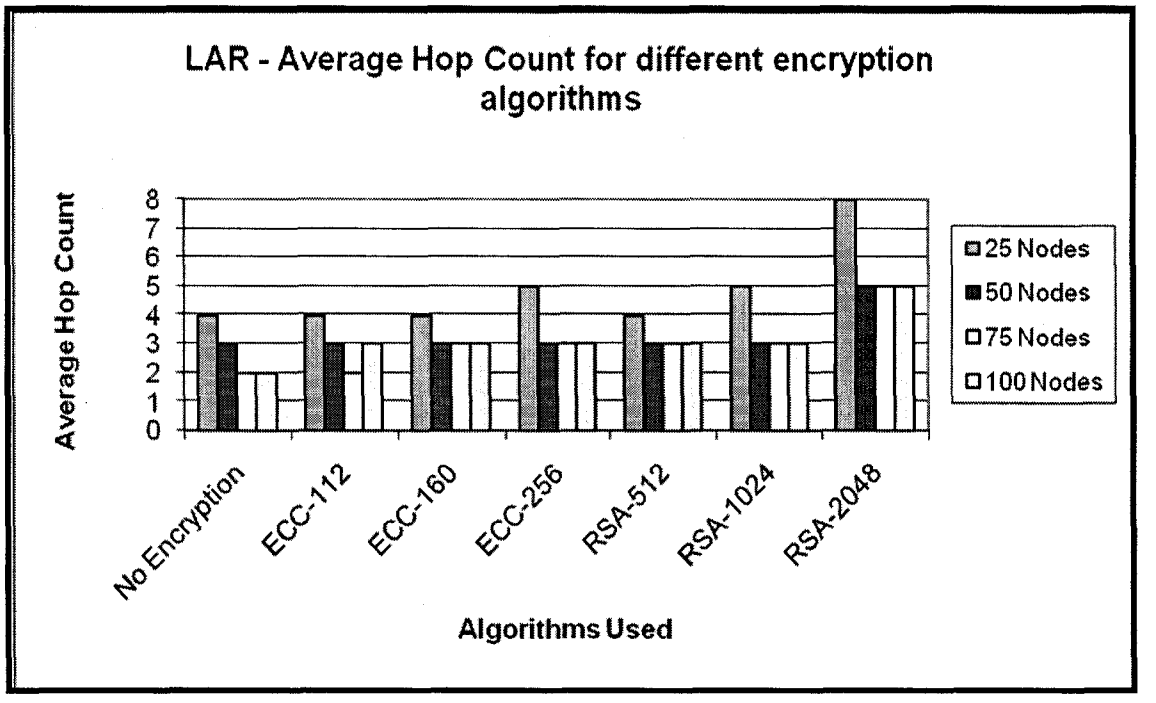

Figure 7-15: LAR-Average Hop Counts for Different Encryption Algorithms

In Figure 7-15, hop count generally increases with increasing key sizes for 25 node networks. For other network sizes and key sizes, the average hop count does not show any appreciable change. The density of the network has no obvious correlation with the average hop count. When equivalent key sizes for different encryption schemes are compared, it is found that ECC has same average hop count than RSA for all network sizes. Hence, the performance of ECC is better than that of RSA.

7.4. Experimental Results for Different Routing Protocols- Comparison

In Figure 7-16, Figure 7-17, Figure 7-18, Figure 7-19 and Figure 7-20, the relation between the different parameters and the different routing protocols is shown. In Figure 7-16, it is seen that with AODV at larger key lengths, RSA has a higher delay than ECC. With OLSR for larger key lengths, ECC shows a gradual delay whereas RSA shows sharp increase in delay. When using larger key lengths, LAR protocols with RSA have higher delay than ECC. It was also observed that the gradual increase in the key lengths results in gradual increase of the delay for all routing protocols. In

terms of delay, the best performance is given by OLSR protocol when used with ECC. Generally, AODV performs average as compared to other schemes.

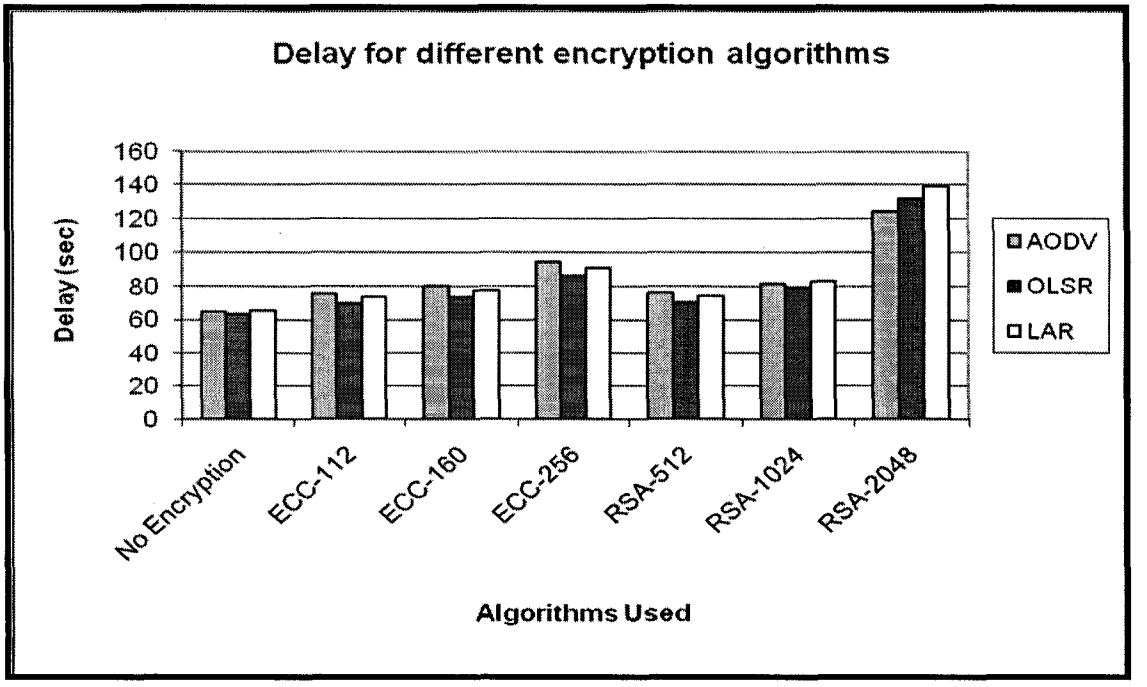

Figure 7-16: Delay for Different Encryption Algorithms

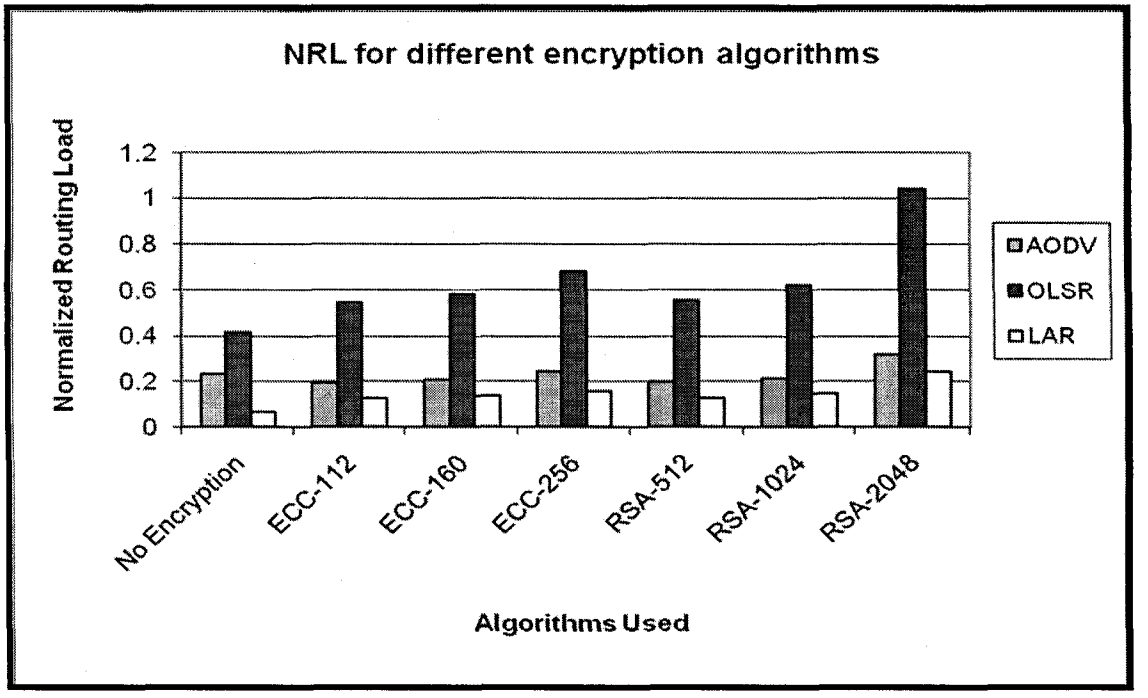

Figure 7-17: NRL for Different Encryption Algorithms

In Figure 7-17, NRL increases gradually for both ECC and RSA encryption schemes when using AODV protocol. OLSR protocol has a highest value of normalized routing load (NRL) than LAR and AODV for all key lengths, thereby, giving the worst performance. LAR showed slight increase in the load value when encryption scheme is used but with no encryption, it remains really low. In terms of routing load at higher value, the best performance is given by LAR protocol.

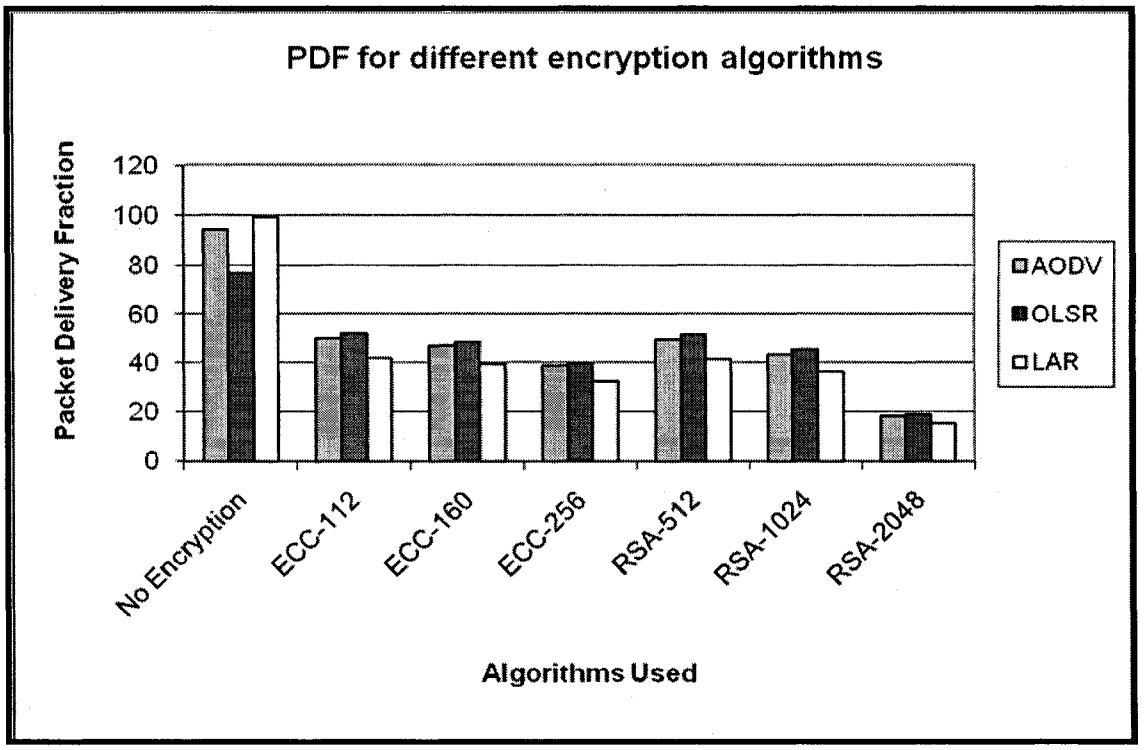

Figure 7-18: PDF for Different Encryption Algorithms

When computing the performance based on packet delivery fraction, the routing protocol that gives the highest PDF is the most efficient. In Figure 7-18, networks which use no encryption have the highest PDF but when comparison is made for a network using encryption, it is seen that ECC gives a better performance than RSA. It is also seen that there is a gradual decrease in PDF for increasing key lengths for both ECC and RSA encryption schemes. Among the three routing protocols, LAR gives the

worst performance even at a larger key length and OLSR gives the best performance for both ECC and RSA encryption schemes.

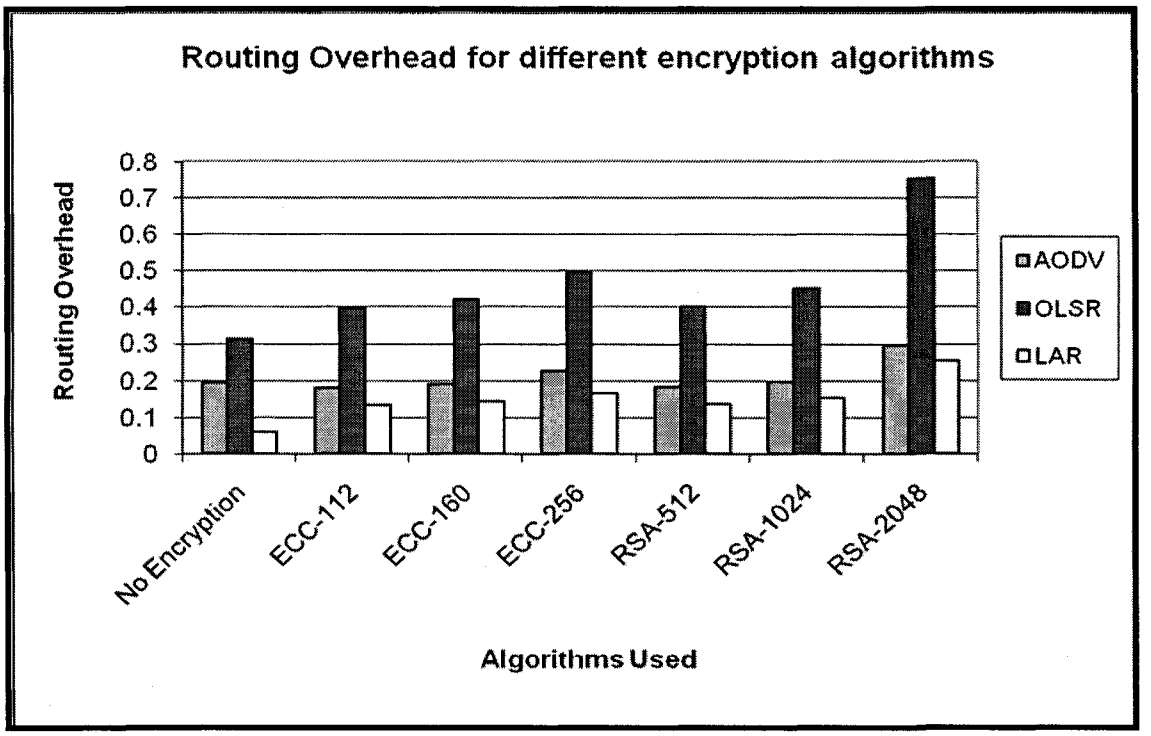

Figure 7-19: Routing Overhead for Different Encryption Algorithms

Routing overhead and hop count is expected to be low when no encryption is used. In Figure 7-19, with AODV for all key lengths, ECC shows slight increase but RSA showed sharp increase confirming better performance of ECC. OLSR protocol has a highest value of routing overhead for every key length than LAR and AODV for all key lengths, thereby, giving the worst performance. Overall, LAR protocol has the best performance for all key sizes for both ECC and RSA encryption schemes.

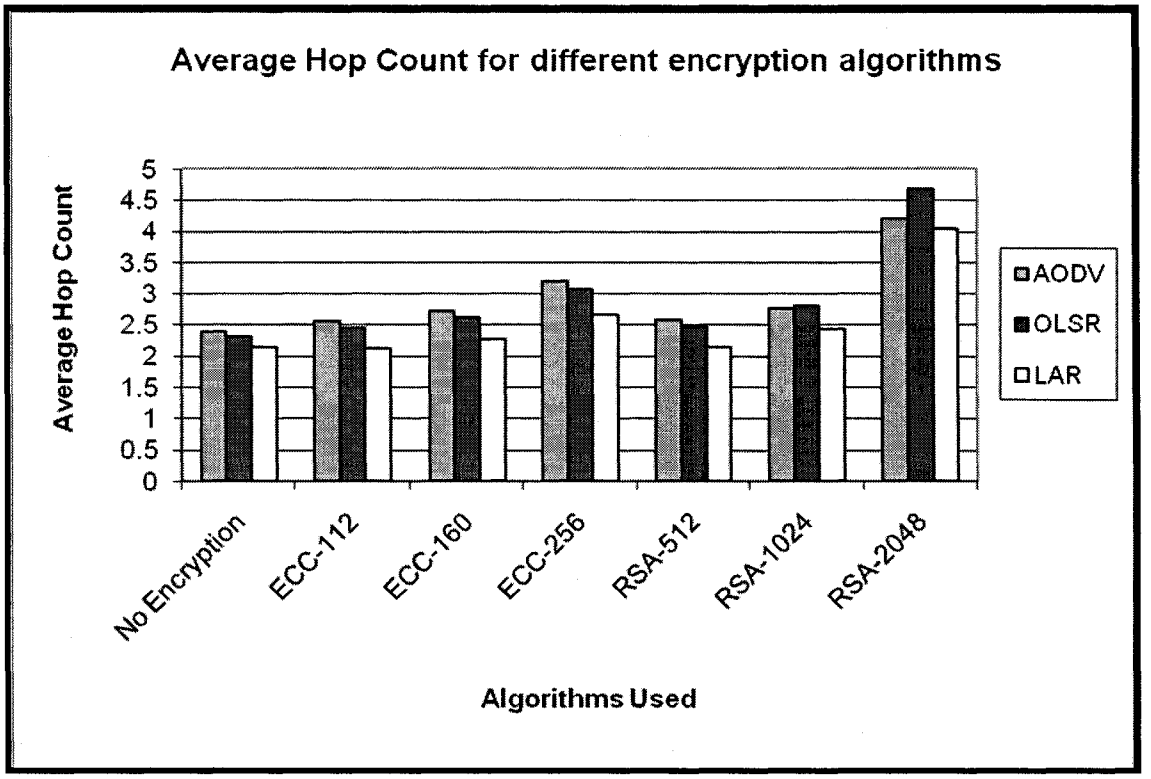

Figure 7-20: Average Hop Counts for Different Encryption Algorithms

In Figure 7-20, AODV shows an increase in hop count with increase in key length for both ECC and RSA encryption schemes. OLSR protocol has a highest value of hop count irrespective of the encryption key length used, thus giving the worst performance. Overall, LAR protocol has the best performance for all key sizes for both ECC and RSA encryption schemes.

7.5. Experimental Comparison between SAODV-RSA and SAODV-ECC

Theoretical comparison between the proposed scheme and other secure schemes is provided in Chapter 2. However, in order to achieve proper comparison, simulations needed to be performed under similar conditions. Therefore, some experiments were performed and their results are provided in Figure 7-21 and Figure 7-22. The experimental results for the other security schemes are taken from [LIN05]. In order to perform comparisons, similar experiments using same parameter settings were

done for several key sizes of ECC and RSA. Since the results in [LIN05] discuss parameters such as packet delivery fraction, delay and routing overhead, results for similar parameter are given under the closest possible scenario in Table 7-2. The results are machine dependent and they affect the performance of the algorithm.

Table 7-2: SAODV-ECC – Performance Scenario

X	Y	Pause Time	Seed	Simulation Time	Traffic	Propagation	Node
250	100	40	10	1800	CBR	Two-ray	10

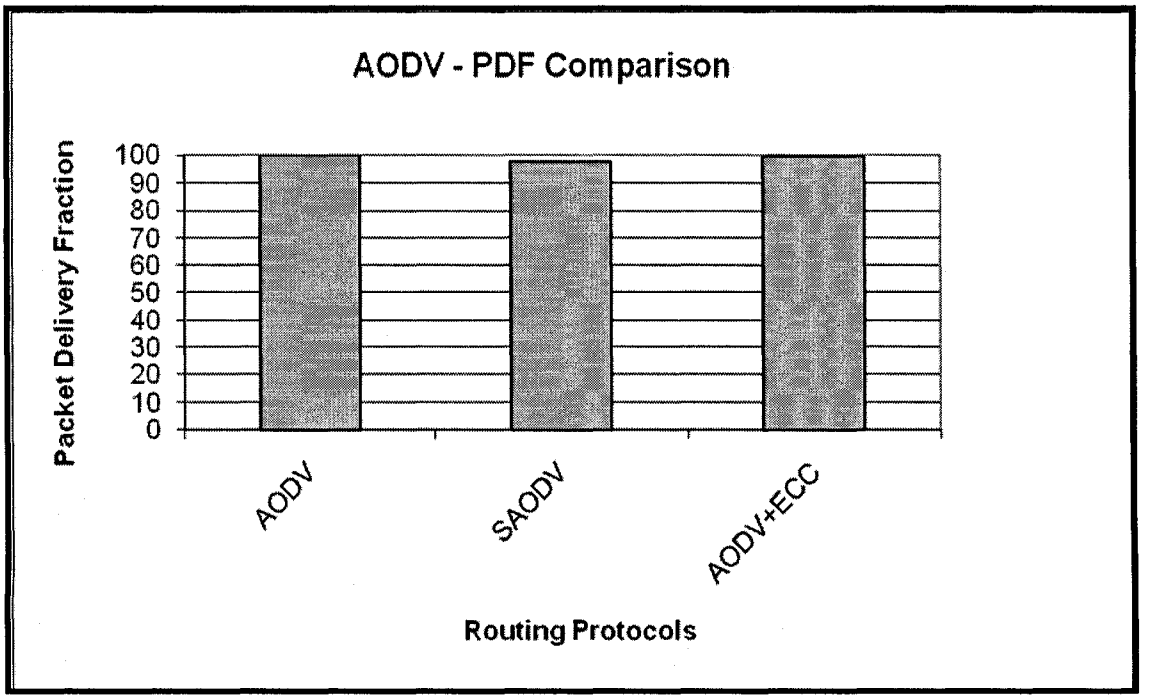

Figure 7-21: Packet Delivery Fraction for Different Routing Protocols

A comparison of the two security schemes is provided in Figure 7-21 and Figure 7-22 [LIN05]. Using the PDF as the comparative factor, it was observed that both schemes showed a similar pattern when tested under similar conditions. Since ECC was tested using a better performing machine, it was expected that its performance will be better than SAODV-RSA. Although from the figures it can be seen that their performance is

also comparable. The results in Figure 7-21 and Figure 7-22 show average results for 10 simulations.

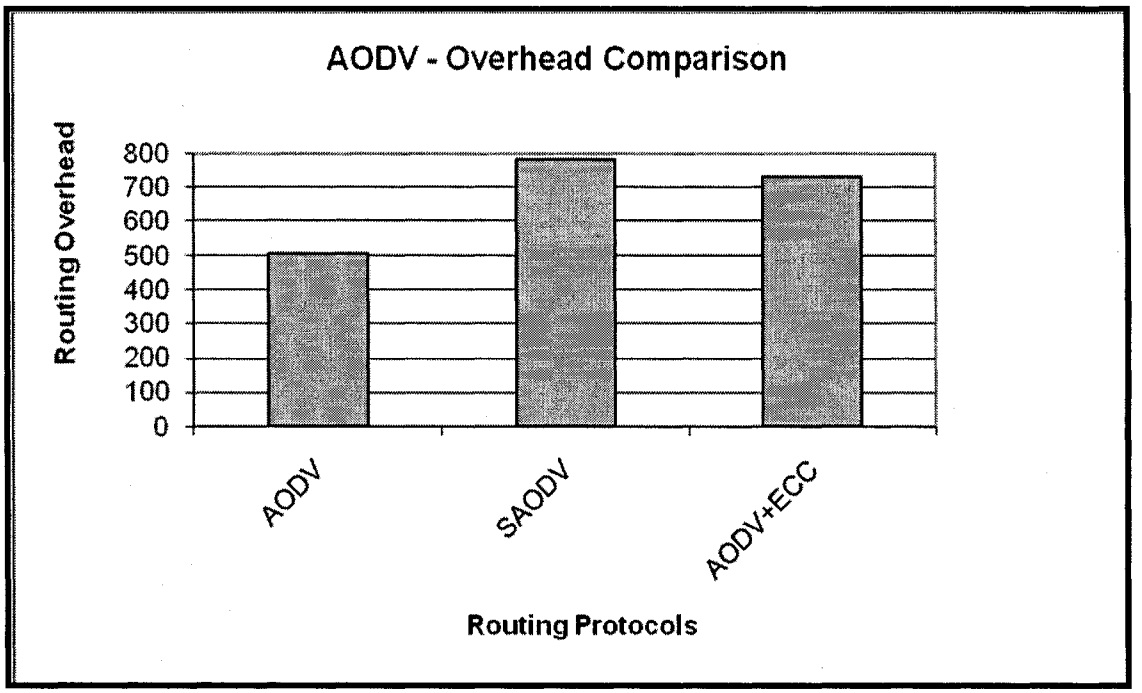

Figure 7-22: Routing Overhead for Different Routing Protocols

In Figure 7-22, a comparison of routing overhead of the two security schemes is provided. The ECC results show that the routing overhead is slightly less than that in case of SAODV-RSA. Therefore it is concluded that Secure AODV with ECC performs as good as Secure AODV with RSA.

7.6. Comparative summary

RSA and ECC are both encryption schemes used mainly for encryption and decryption of data. The purpose of the above section is to study the efficiency of secure version of routing protocol such as AODV, which uses RSA, against the proposed scheme, which uses ECC, in the presence of the variable factors and compare both schemes in terms of performance. Results from realistic scenarios with variable parameters are analyzed and compared. Since SAODV-RSA is a very well-

known standard, it was best to compare these two schemes. Results for SAODV-RSA have been taken from [LIN05] while the results for ECC are generated under similar scenario given in Table 7-2. The evaluation parameters are packet delivery fraction and routing overhead. From the results, it is seen that packet delivery fraction for both scheme is almost the same while routing overhead is slightly high for SAODV-RSA. Therefore, it could be concluded that Secure AODV with ECC is as good as SAODV-RSA.

7.7. Conclusion

All the routing protocols are compared in terms of delay, routing load, routing overhead, packet delivery fraction and average hop count. Following conclusions can be derived from the results.

1. In terms of delay, the best performance is given by OLSR protocol when used with ECC. Generally, AODV performs better than other schemes.
2. In terms of NRL at higher value, the best performance is given by LAR protocol and the worst performance is given by OLSR.
3. Among the three routing protocols for PDF, LAR gives the worst performance even at a larger key length and OLSR gives the best performance for both ECC and RSA encryption schemes.
4. In terms of routing overhead and average hop count, LAR protocol has the best performance for all key sizes for both ECC and RSA encryption schemes and the worst performance is given by OLSR.

Therefore, it can be said that LAR gives the best performance for NRL, average hop count and routing overhead. OLSR gives the best performance for delay and PDF while AODV always gave an average performance. ECC has a better performance in delay, routing load, packet delivery fraction, routing overhead and average hop count.

For equivalent number of bits, ECC performs better due to smaller key length. A network where no encryption is used has less delay, routing load and routing overhead with high packet delivery ratio and acceptable hop count.

One could expect that SAODV with RSA will give better performance and will be widely applicable in suitable cryptographic systems but quite different results were observed. Secure AODV with ECC showed a comparable performance when compared with SAODV with RSA. Since increase in key sizes increases the computational overhead slowly for ECC, it will be an attractive option in future. However, researchers are still experimenting to establish the suitability of ECC to different applications and to validate the efficient performance of ECC.

Chapter 8

Conclusions and Future Work

In this chapter, the major conclusions drawn from this work is summarized. Possible future work arising from this research is also provided.

8.1. Conclusions

Security in the ad hoc network seemed to be the most interesting and complex yet the most researched area. The idea of this research was to achieve secure communication between any two nodes in the network while minimizing the hardware usage and wastage. Since the focus is more on the security aspect of the ad hoc network while trying to minimize the load on individual devices, Elliptic curve fits in perfectly with the requirement by providing smaller key.

To evaluate the performance of ECC, a comparison with a widely used RSA security scheme was done.When RSA and ECC are compared, the overall performance of ECC was found to be better and it requires a fairly less key size to attain the similar security level as RSA.

These results were convincing enough to proceed and an ECC scheme is proposed, tested and compared to existing security scheme. The three routing protocols AODV, OLSR and LAR are used with both ECC and RSA encryption schemes under variable

network size. The experimentation used five evaluation parameters: Delay, NRL, PDF, Average hop count and Routing overhead.

A network where no encryption is used has less delay, routing load and routing overhead with high packet delivery ratio and acceptable hop count.

ECC scheme in general had a better performance in delay, routing load, packet delivery fraction, routing overhead and average hop count than RSA. For equivalent number of bits, ECC performed better than RSA due to smaller key lengths.

ECC with LAR gave the best performance for NRL, average hop count and routing overhead. ECC with OLSR gave the best performance for delay and PDF. ECC with AODV gave an average performance in all cases.

Secure AODV with ECC showed a comparable performance when compared with SAODV with RSA. Since increase in key sizes increases the computational overhead slowly for ECC, it will be an attractive option in future. However, researchers are still experimenting to establish the suitability of ECC to different applications and to validate the efficient performance of ECC.

8.2. Future Work

For studying and monitoring performance of the elliptic curve security scheme, it was tested and compared to existing security scheme. Also for simplicity, the security level of the encryption scheme is set to the current acceptable security level. In future, experiments could be performed to test whether this scheme holds up against the attacks and then may be provide an estimate of the years required to break this security level. This estimate could also be comparatively studied against the existing standard encryption schemes which are widely employed in various applications.

Another factor that was cut off to simplify implementation is the validity of keys and re-issuing of keys. This may also be implemented and tested in future. Once all the above features have been added, the scheme could be ported to a handheld device and then tested in the real environment. It would be interesting to see the performance of the scheme under different conditions. Several factors such as the movement and speed of hardware devices, attackers, weather conditions, area of nodes, etc can affect the performance of the scheme. Also the increase in the scale of the network and interruption in the signals can affect scheme's performance. Routing techniques is another factor which enhances the delivery patterns of a network. Therefore, routing techniques must also be studied in detail to analyze which techniques uses the least resources and provided the best results.

Due to the vast structure of ECC, several other schemes can also be suggested which provide one or more security goals. These schemes were not implemented due to limited time. However, development and experiments of these schemes could be done in near future. Encryption techniques are based on the data which is encrypted and where they are encrypted. Therefore, several combinations can be derived from the two. One such version is where only the headers are encrypted at the intermediate nodes while the data is encrypted at the source. Second version is where only partial header in encrypted by intermediate node while the rest in encrypted at the source. From experimentation and observations, it has been noticed that ECC could be the next big thing. It has the power to change the face of the cryptography. It will not only shrink the special need of the files but also improve the performance of the encryption engines. With all the implementation and testing, ECC is soon set to become a well established standard encryption /decryption scheme.

Glossary

Bit length	The length in bits of a bit string.
Cofactor	The order of the elliptic curve group divided by the (prime) order of the generator point specified in the domain parameters.
Domain parameters	The parameters used with a cryptographic algorithm that are common to a domain of users.
Entity/User	An individual (person), organization, device or process. Party" is a synonym.
Ephemeral key	A key that is intended for a very short period of use. The key is ordinarily used in exactly one transaction of a cryptographic scheme; an exception to this is when the ephemeral key is used in multiple transactions for a key transport broadcast. Contrast with static key.
Hash function	A function that maps a bit string of arbitrary length to a fixed length bit string. Approved hash functions satisfy the following properties: 1. (One-way) It is computationally infeasible to find any input that maps to any pre-specified output and 2. (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output. Approved hash functions are specified in FIPS 180-2.
Identification	A bit string that is associated with a person, device or organization. It may be an identifying name or may be something more abstract (for example, a string consisting of an IP address and timestamp). If a party owns a static key pair that is used in a key agreement transaction, then the identifier assigned to that party is one that is

	bound to that static key pair. If the party does not contribute a static public key as part of a key agreement transaction, then the identifier of that party is a non-null identifier selected in accordance with the protocol utilizing the scheme.
Key agreement	A key establishment procedure where the resultant secret keying material is a function of information contributed by two participants, so that no party can predetermine the value of the secret keying material independently from the contributions of the other parties. Contrast with key transport.
Key agreement mechanism	The instance that results in shared secret keying material among different parties using a key agreement scheme.
Key confirmation	A procedure to provide assurance to one party (the key confirmation recipient) that another party (the key confirmation provider) actually possesses the correct secret keying material and/or shared secret.
Key derivation	The process by which keying material is derived from shared secret and other information.
Key establishment	The procedure that results in shared secret keying material among different parties.
Message authentication code (MAC) algorithm	Defines a family of one-way cryptographic functions that is parameterized by a symmetric key and produces a MacTag on arbitrary data. A MAC algorithm can be used to provide data origin authentication as well as data integrity. In this recommendation, a MAC algorithm is used for key confirmation and validation testing purposes.
Receiver	The party that receives secret keying material via a key transport transaction. Contrast with sender.

Security strength	A number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system.
Sender	The party that sends secret keying material to the receiver using a key transport transaction.
Shared secret	A secret value that has been computed using a key agreement scheme and is used as input to a key derivation function.
Shared secret keying material	The secret keying material that is either 1) derived by applying the key derivation function to the shared secret and other shared information during a key agreement process or 2) is transported during a key transport process.
Static key	A key that is intended for use for a relatively long period of time and is typically intended for use in many instances of a cryptographic key establishment scheme. Contrast with an ephemeral key.
Symmetric key algorithm	A cryptographic algorithm that uses one secret key that is shared between authorized parties.
Trusted third party	A third party, such as a CA, that is trusted by its clients to perform certain services.

Note: The definitions provided above are defined as used in [NIS06, BAR07].

References

[ANS99] ANSI X9.63-199x, "Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography", *American National Standard, Working Draft*, October 1999, Available at http://www.whuecc.com/ECC/X9.63-199x.pdf.

[BAO03] F. Bao, R. Deng and H. Zhu, "Variations of Diffie-Hellman problem", *in Proceedings of the Fifth Conference on Information and Communications Security (ICIS 2003)*, pp. 301–312, 2003.

[BAR07] E. Barker, W. Barker, W. Burr, W. Polk and M. Smid, " Recommendation for Key Management-Part 1: General", *National Institute of Standards and Technology NIST Special Publication 800-57*, March 2007, Available at http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf

[BEL96] M. Bellare, R. Canetti and H. Krawczyk, "Keying Hash Functions for Message Authentication", *in CRYPTO'96, Springer LNCS*, Vol. 1109, pp. 1-15, 1996.

[BLA99] I. Blake, G. Seroussi and N. Smart, "Elliptic Curves in Cryptography", *Cambridge University Press*, 1999.

[CER00] Certicom Research, "Standards for efficient cryptography", SEC 1: Elliptic Curve Cryptography, Version 1.0", *Certicom*, September 2000, Available at http://www.secg.org/collateral/sec1_final.pdf.

[CLA03a] T. Clausen, P. Jaqcuet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum and L. Viennot, "Optimized link state routing protocol", *IETF Draft*, 2003.

[CLA03b] T. Clausen, C. Adjih, P. Jacquet, A. Laouiti, A. Muhlethaler and D.

Raffo, "Securing the OLSR Protocol", *in Proceeding of IFIP Med-Hoc-Net*, June 2003.

[CRA98] R. Cramer and V. Shoup, "A practical public key encryption system provably secure against adaptive chosen cipher text attack", *in Proceedings of Crypto'98*, Vol. 1642 of LNCS, pp. 13–25, August 1998.

[DES07] V. S. Deshpande, "Security in Ad-Hoc Routing Protocols", *in Auswireless'06 Conference*, 2007, Available at http://epress.lib.uts.edu.au/dspace/bitstream/2100/155/1/8_Deshpande.pdf, Viewed on 15 May, 2007.

[DIF76] W. Diffie and M. E. Hellman, "New directions in cryptography", IEEE Trans. Info. Theory IT-22, pp. 644-654, November 1976.

[ENG62] D. C. Engelbart, "Augmenting Human Intellect: A Conceptual Framework", *Summary Report AFOSR-3233*, Stanford Research Institute, USA, October 1962, Available at http://www.bootstrap.org/augdocs/friedewald030402/augmentinghumanintellect/ahi62index.html, Viewed on 20 May 2007.

[FAL00] K. Fall and K. Varadhan, "The ns Manual (formerly ns Notes and Documentation)", *The VINT Project*, UC Berkeley, LBL, USC/ISI and Xerox PARC, August 2000, Available at http://www.isi.edu/nsnam/ns/tutorial/index.html, Viewed on 15 May 2007.

[HAN04] D. Hankerson, A.J. Menezes and S.A. Vanstone, "Guide to elliptic curve cryptography", *Springer Professional Computing*, New York, USA, 2004.

[HUB01] J. P. Hubaux, L. Buttyan and S. Capkun, "The Quest for security in Mobile Ad Hoc Networks", *in 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing*, October 2001.

[IEE00] IEEE P1363, "Standard Specifications for Public-Key Cryptography", *Institute of Electrical and Electronics Engineers*, 2000, Available at http://grouper.ieee.org/groups/1363/, Viewed on 15 May, 2007.

[ISL05] Md. S. Islam, "Efficient Key Management Scheme for Mobile Ad Hoc Network", *Master's Thesis, Department of Computer and System Sciences*, Stockholm, Sweden, February 2005.

[ISO02] ISO/IEC 15946-1, "Information technology-Security techniques-Cryptographic techniques based on elliptic curves-Part 1:General", 2002, revised 2008, Available at http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=31075, Viewed on 15 May, 2007.

[ISO08] ISO/IEC 14888-1, "Information technology-Security techniques-Digital signatures with appendix-Part 1: General", 1998, Revised 2008, Available at http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=44226, Viewed on 15 May 2007.

[ISO98] ISO/IEC 15946-3, "Information technology-Security techniques-Cryptographic techniques based on elliptic curves-Part 3: Key establishment", *Working draft*, 1998.

[JOH94] D. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts", *in proceedings of the Workshop on IEEE Computer Society*, Mobile Computing Systems and Applications, pp. 158-163, USA, December 1994.

[JOH96] D. Johnson, "Diffie-Hellman Key Agreement Small Subgroup Attack", *a Contribution to X9F1 by Certicom*, July 1996.

[KEM02] R. A. Kemmerer and G. Vigna, "Intruder detection: A brief history and overview", *IEEE Computer*, Vol. 35, No. 4: pp 27–30, 2002. and ",

IEEE Security & Privacy, 2002, Available at http://www.computer.org/security/supplement1/htm/>.e, Viewed on 15 May, 2007.

[KO98] Y. Ko and N.H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks", *in proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM),* pp. 66–75, 1998.

[KOB00] N. Koblitz, A.J. Menezes and S.A. Vanstone, "The state of elliptic curve cryptography", *in Designs, Codes and Cryptography'02*, Vol. 19, No. 2-3, pp. 173–193, 2000.

[KOB04] N. Koblitz and A. Menezes, "A survey of public-key cryptosystems", SIAM Review, Vol. 46, No. 4, pp. 599–634, 2004.

[KOB87] N. Koblitz, "Elliptic curve cryptosystems", *Mathematics of Computation*, Vol. 48, pp. 203–209, 1987.

[KRA97] H. Krawcyzk, M. Bellare and R. Canetti, "HMAC: Keyed Hashing for Message Authentication", *Internet Engineering Task Force, Internet RFC 2104*, 1997. Available at: http://www.ietf.org/, Viewed on 15 May February 2007.

[KUM05] K. P. Kumar, G. Shailaja and A. Saxena, "Secure and Efficient Threshold Key Issuing Protocol for ID-based Cryptosystems", *Institute for Development and Research in Banking Technology*, Hyderabad, India, 2005.

[LIN05] Y. Lin, A. H. M. Rad, V. W. S. Wong and J.H. Song, "Experimental comparisons between saodv and aodv routing protocols", *In WMuNeP '05: Proceedings of the 1st ACM workshop on Wireless multimedia networking and performance modeling*, ACM Press, pp. 113–122, New York, NY, USA, 2005.

[LOP00] J. Lopez and R. Dahab, "Performance of elliptic curve cryptosystems",

Technical report'00, May 2000. Available at http://www.dcc.unicamp.br/ic-main/publications-e.html, Viewed on 15 May, 2007.

[MEN96] A. J. Menezes, S. A. Vanstone and P. C. Van Oorschot, "Handbook of Applied Cryptography", *CRC Press Incorporation*, Florida, 1996.

[NIS06] NIST, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", *Special Publication 800-56A*, March 2006.

[ODL95] A. Odlyzko, "The Future of Integer Factorization", *CryptoBytes*, Vol. 1, No. 2, pp. 5–12, 1995.

[OOR94] P.C. van Oorschot and M. Wiener, "Parallel collision search with applications to hash functions and discrete logarithms", *in 2nd ACM Conference on Computer and Communications Security*, pp. 210–218, ACM Press, 1994.

[PER99] C. Perkins, E. Belding-Royer and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", *in proceedings of IEEE workshop on mobile computing system and applications*, pp. 90-100, February 1999.

[PER94] C. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", *in proceedings of SIGCOMM'94 conference on communications architectures ,protocols and applications*, pp. 234-244, August 1994, Available at http://www.cs.umd.edu/projects/mcml/papers/Sigcomm94.ps. Viewed on 15 May, 2007.

[PFL97] C. Pfleeger, "Security in computing", Upper Saddle River, NJ, Prentice Hall, 1997.

[RAM02] R. Ramanathan and J. Redi, "A Brief Overview of Ad Hoc Networks: Challenges and Directions", *IEEE Communications Magazine*, Vol. 40, No. 5, pp. 20-22, May 2002.

[ROY99] E. M. Royer and C. K. Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks", *IEEE Pers. Communications*, Vol. 6, No. 2, April 1999.

[RUE94] R. Rueppel and P.C. van Oorschot, "Modern key agreement techniques", Computer Communications, Vol.17, No.7, pp. 458-465, July 1994.

[SEC00] SEC 2, "Recommended Elliptic Curve Domain Parameters. Standards for Efficient Cryptography Group", *Working Draft*, September 2000. Available at http://www.secg.org/. Viewed on 15 May, 2007.

[SHA84] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes", *Advances in Cryptology: in proceedings of CRYPTO 84*, Lecture Notes in Computer Science, pp. 47-53, 1984.

[SMA99] N. Smart, "The discrete logarithm problem on elliptic curves of trace one", *Journal of Cryptology*, Vol. 12, pp. 193–196, 1999.

[STA02] W. Stallings, "Cryptography and Network Security: Principles and Practice", Pearson Education, 2002

[SUB99] M. W. Subbarao, "Ad Hoc Networking Critical Features and Performance Metrics", *Wireless Communications Technology Group, NIST*, October 1999.

[VIN02] P. Vinayakray-Jani, "Security within Ad hoc Networks", *in PAMPAS'02 Workshop Requirements for Mobile Privacy & Security*, London, September 2002.

[WAS03] L.C. Washington, "Elliptic Curves: Number Theory and Cryptography", *CRC Press*, 2003.

[WIL97] S. B. Wilson, D. Johnson and A.J. Menezes, "Key agreement protocols and their security analysis", *in proceedings of the sixth IMA International Conference on Cryptography and Coding*, Vol. 1355, pp 30–45, 1997.

[WU05] B. Wu, J. Chen, J. Wu and M. Cardei, "A Survey on attacks and Countermeasures in Mobile Ad Hoc Networks", *Department of Computer Science and Engineering, Florida Atlantic University,* Florida, USA, *Wireless/Mobile Network Security*, Chapter 12, August 2005

[WU07] B. Wu, J. Wu, E. B. Fernandez and S. Magliveras, "Secure and Efficient Key Management in Mobile Ad Hoc Networks", *Journal of Network and Computer Applications*, Vol. 30, No. 3, pp. 937-954, Florida ,USA, August 2007.

[ZAP02] G. M. Zapata, "Secure Ad hoc On-Demand Distance Vector Routing", ACM *Mobile Computing and Communications Review (MC2R)*, Vol. 6, No.3, pp.106–107, July 2002.

[ZHO03] D. Zhou, M. Ilyas and R. C. Dorf, "Security issues in ad hoc networks", *in the Handbook of Ad Hoc Wireless Networks*, The Electrical Engineering Handbook Series. CRC Press, Florida, USA, pp. 569-582, 2003.

[ZHO99] L. Zhou and Z. J. Haas, "Securing Ad hoc Networks", *IEEE Network Magazine*, Vol. 13, No. 6, pp. 24-30, Cornell University, USA, November/December 1999.