ENERGY EFFICIENT LINK LAYER SECURITY PROTOCOL IN WIRELESS SENSOR NETWORK

Enrollment No.	-101229
Name of Student	- ANSHUL SINGLA
Name of supervisor(s)	- Dr. YASHWANT SINGH

MAY - 2014

Project report submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

DEPARTMENT OF **COMPUTER SCIENCE AND ENGINEERING**

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

Table of Contents

CERTIFICATE

This is to certify that the work titled "**ENERGY EFFICIENT LINK LAYER SECURITY PROTOCOL IN WIRELESS SENSOR NETWORK**" submitted by "**ANSHUL SINGLA**" in partial fulfillment for the award of degree of B.Tech of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor	……………………..
Name of Supervisor	……………………..
Designation	……………………..
Date	……………………..

ACKNOWLEDGEMENT

I would like to express my greatest gratitude to my mentor **Dr.Yashwant Singh** who has helped and supported me throughout our project. I’m grateful to him for his continuous support for the project, from the initial advice in the early stages of conceptual inception and through ongoing advice to this day.

Signature of the student	……………………..
Name of Student	……………………..
Date	……………………..

Abstract

An ad-hoc mobile Wireless Sensor Network is a collection of mobile sensor nodes that are dynamically and randomly located in such a manner that the interconnections between nodes are capable of changing on a continual basis. Although the primary goal of such an ad-hoc network is correct and efficient route establishment between source and destination so that messages may be delivered in a timely manner but military tactical and other security-sensitive operations are still the main applications of ad hoc networks. One main challenge in design of these networks is their vulnerability to security attacks. In this project, threats an ad hoc network faces and the security goals to be achieved have been reviewed. In particular, it takes advantage of multiple routes between nodes — to defend routing against denial of service attacks and cryptographic schemes to prevent from malicious attacks by intruders and to build a highly secure and highly available key management service, which forms the core of our security framework.

Signature of Student	Signature of Supervisor
Name	Name:
Date	Date

Table of Figures

List of Symbols and Acronyms

WSN: Wireless Sensor Network.

MANETs: Mobile ad hoc networks.

LAR: Location -Aided Routing

GPS: Global Positioning System

SDLC: Software Development Life Cycle

Chapter 1: Introduction

1.1 INTRODUCTION

Wireless technology has expanded the limits of our world. Through this innovation, people have been given freedom to work away from their desks or even outside. The newfound freedom that people are beginning to enjoy with their computers has started making the world of technology and nature blend. Wireless Sensor Networks are the next stage of this technology-nature cohesion.

A WSN is formed by densely deployed sensor nodes in an application area. In most deployments, the sensor nodes have self-organizing capabilities, to form an appropriate structure in order to collaboratively perform a particular task. Wireless Sensor Networks are found suitable for applications such as surveillance, precision agriculture, smart homes, automation, vehicular traffic management, habitat monitoring, and disaster detection. . Although the primary goal of such an ad-hoc network is correct and efficient route establishment between source and destination so that messages may be delivered in a timely manner but military tactical and other security-sensitive operations are still the main applications of ad hoc networks. One main challenge in design of these networks is their vulnerability to security attacks.

Ad hoc networks are a new paradigm of wireless communication for mobile hosts (which we call nodes). In an ad hoc network, there is no fixed infrastructure such as base stations or mobile switching centers. Mobile nodes that are within each other's radio range communicate directly via wireless links, while those that are far apart rely on other nodes to relay messages as routers. The key constraints in the development of WSNs are limited battery power, cost, memory limitation, limited computational capability, and the physical size of the sensor nodes. In particular, it takes advantage of multiple routes between nodes to defend routing against denial of service attacks and cryptography schemes to prevent from malicious attacks by intruders and to build a highly secure and highly available key management service, which forms the core of our security framework.

1.2 Genesis of Problem

In present world, everything is becoming wireless, from home technology to military service and from environmental to hospital services. All traditional mobile wireless networks, ad hoc networks rely on fixed infrastructure or on a centralized approach which had many drawbacks. With increase in sensors, more messages were passed to base station consuming more bandwidth. It lacked scalability and main application areas of WSN such as military tactical and other security-sensitive operations needed a highly secure and highly available key management service to prevent from malicious attacks by intruders.
So, this is where this project comes into play. In this project hosts rely on each other to keep the network connected. This project is designed in such a way that it result it provides an efficient routing method and needed security for the transmission of confidential data/messages to prevent

from intruders. I have applied waterfall model for the development of this project. In particular, it takes advantage of multiple routes between nodes to defend routing against denial of service attacks. It also uses cryptographic schemes i.e. RSA, to build a highly secure and highly available key management service.

1.3 Problem Statement

The traditional routing algorithms make use of a centralized approach or rely on fixed infrastructure. As the number of sensors increases in the network, more messages are passed on towards the base station and will consume additional bandwidth. Thus, this approach is not fault tolerant as there is single point of failure and lacks scalability and with many of the applications involving communication of highly sensitive data, security became a primary concern. However, incorporating security features in WSNs is challenging due to the specific constraints such as limited memory and restricted energy supply.

Therefore, this project includes implementation of a security protocol in the network which provides node authentication and confidentiality. In addition to stronger security measures, the simulation results demonstrate removed dependency from the fixed infrastructure approach.

1.4 OBJECTIVE

The main objective of our project is:

- To provide a high level security in the network which provides-
  - Message Confidentiality
  - Node Authentication

- To provide an efficient routing free from centralized approach.

1.5 PROPOSED SYSTEM

In proposed system does not rely on any stationary infrastructure. The concept is, instead of making data transit through a fixed base station, nodes consequentially forward data packets from one to another until a destination node is finally reached. Typically, a packet may travel through a number of network points before arriving at its destination. The routers and hosts are free to move randomly and organize themselves in an arbitrary fashion, thus the network topology changes rapidly and unpredictably. Thus the Ad Hoc network co-operates to forward packets for each other to communicate without the help of a base station increasing the scalability and fault tolerance. But there are several issues like selfish nodes, malicious behavior, routing challenges, security etc. To incorporate the security feature RSA encryption scheme is used in which source sends the encrypted data packet to the destination through the route discovered and then destination decrypts the data packet received from the source and sends the acknowledgement.

1.6 APPROACH USED

When some source node originates a new packet addressed to some destination node, the source node places in the header of the packet a "source route" giving the sequence of hops that the packet is to follow on its way to the destination. Normally, the sender will obtain a suitable source route by searching its "Route Cache" of routes previously learned; if no route is found in its cache, it will initiate the Route Discovery protocol to dynamically find a new route to this destination node. For example, suppose a node A is attempting to discover a route to node E. The Route Discovery initiated by node A in this example would proceed as follows:

```
                 ^    "A"    ^   "A,B"   ^  "A,B,C"  ^ "A,B,C,D"
    |   id=2    |   id=2    |   id=2    |   id=2
 +-----+     +-----+     +-----+     +-----+     +-----+
 |  A  |---->|  B  |---->|  C  |---->|  D  |---->|  E  |
 +-----+     +-----+     +-----+     +-----+     +-----+
    |           |           |           |
    v           v           v           v
```

Figure 1.1: Routing Example

To initiate the Route Discovery, node A transmits a "Route Request" as a single local broadcast packet, which is received by (approximately) all nodes currently within wireless transmission range of A, including node B in this example. Each Route Request identifies the initiator and target of the Route Discovery, and also contains a unique request identification (2, in this example), determined by the initiator of the Request. When another node receives this Route Request (such as node B in this example), if it is the target of the Route Discovery, it returns a "Route Reply" to the initiator of the Route Discovery, giving a copy of the accumulated route record from the Route Request; when the initiator receives this Route Reply, it caches this route in its Route Cache for use in sending subsequent packets to this destination. Otherwise, this node appends its own address to the route record in the Route Request and propagates to E.

And for DATA SECURITY-

i) Message Encryption
ii) Message Decryption

DATA SECURITY USING RSA

- Source sends the encrypted data packet to the destination through the route discovered.
- Destination decrypts the data packet received from the source.

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

Approach used in developing this project is waterfall model. The following figure shows the various aspects of waterfall model approach of SDLC (Software Development Life Cycle).

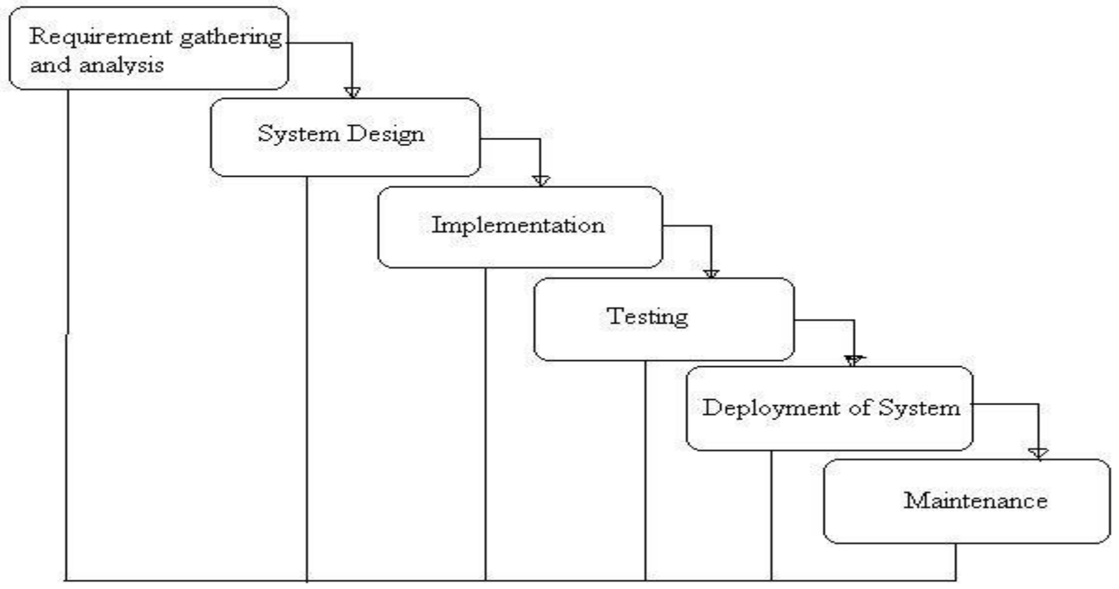


Figure 1.2: Software Development Life Cycle

1.7 Organisation of Thesis

In Chapter 2, we have discussed the literature survey. We have looked upon the various routing protocols of WSN and encryption schemes for its applications where security of data is a major concern like military services and from home security system. We have also seen the various the challenges faced by WSN in implementing the desired protocols t. We have also seen the various features of WSN.

In Chapter 3 we have done feasibility study and requirement analysis. We have discussed the various designs like Data Flow Diagrams, Flow Chart, and Use Case etc. Next part includes coding which discusses the basic functioning, the working of clusters, etc.

In Chapter 4 we have thereby concluded our report.

CHAPTER 2: LITERATURE SURVEY

2.1 Introduction

A wireless sensor network consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding.

2.2 Parts of a WSN

While we associate a computer with a PC, the technical definition of a computer is a thing that computes, be it human or machine. Most sensors or WSN consist of five crucial components. These components include a number of sensors, such as temperature, moisture, and vibration sensors, a power source, in the case of older motes, 2 AA batteries, a radio transmitter/receiver, and an electric.

Sensors: When motes are under construction, their intended purpose often dictates the sensors that are added to the mote. The mote in Figure 2 contains three types of sensors: temperature, moisture, and vibration. This is a fairly typical mote, but some motes have many more functions. There are motes that take photographs of the surroundings, sense motion, measure light intensity, and much more. The sensors are attached to the mote base and communicate readings to the electronic brain.

Power Source: The power source for the mote also depends the mote's intended use. If the mote is designed to last a very long time, say one year, it will have a larger power source than a mote that is only meant to run for a month. The power sources usually range between a couple of AA batteries, and a watch battery, but with the new smart-dust motes, also called "Spec," they can collect enough energy to sustain themselves from ambient light, or even vibrations. The power source is connected to the mote base and provides energy required to run the sensors, electronic brain, and radio.

Radio: The radio consists of a radio transmitter and a radio receiver. Both of these parts must exist for any mote to fully communicate with the other motes. The radio, when transmitting, receives information from the electronic brain and broadcasts the data to other motes according to the network connections. In the other direction, when receiving, the radio receives information from another mote's radio and transmits it to the electronic brain. The radio is connected to the mote base.

The Electronic Brain: The older motes' brains consist of a microprocessor and some flash memory. Many of them have connectors to add other processes and sensors with ease. The MEMS motes also contain an analog-digital converter. The basic functions of the electronic brain are to make decisions and deal with collected data. The electronic brain stores collected data in its memory until enough information has been collected. Once this point is reached, the microprocessor portion of the electronic brain then puts the data in "envelopes," or packages of data formatted for greatest transferring efficiency. These envelopes are then sent to the radio for broadcast. The brain also communicates with other motes to maintain the most effective network in much the same way it deals with data. The electronic brain is connected to the base and interacts with the sensors and radio.

2.3 Applications of WSN

- **Area monitoring**

  Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. A military example is the use of sensors detect enemy intrusion; a civilian example is the geo-fencing of gas or oil pipelines.

- **Health care monitoring**

  The medical applications can be of two types: wearable and implanted. Wearable devices are used on the body surface of a human or just at close proximity of the user. The implantable medical devices are those that are inserted inside human body. There are many other applications too e.g. body position measurement and location of the person, overall monitoring of ill patients in hospitals and at homes.

- **Air pollution monitoring**

  Wireless sensor networks have been deployed in several cities (Stockholm, London and Brisbane) to monitor the concentration of dangerous gases for citizens. These can take advantage of the ad hoc wireless links rather than wired installations, which also make them more mobile for testing readings in different areas.

- **Forest fire detection**

  A network of Sensor Nodes can be installed in a forest to detect when a fire has started. The nodes can be equipped with sensors to measure temperature, humidity and gases which are produced by fire in the trees or vegetation. The early detection is crucial for a successful action of the firefighters; thanks to Wireless Sensor Networks, the fire brigade will be able to know when a fire is started and how it is spreading.

- **Landslide detection**

  A landslide detection system makes use of a wireless sensor network to detect the slight movements of soil and changes in various parameters that may occur before or during a landslide. Through the data gathered it may be possible to know the occurrence of landslides long before it actually happens.

- **Water quality monitoring**

  Water quality monitoring involves analyzing water properties in dams, rivers, lakes & oceans, as well as underground water reserves. The use of many wireless distributed sensors enables the creation of a more accurate map of the water status, and allows the permanent deployment of monitoring stations in locations of difficult access, without the need of manual data retrieval.

- **Natural disaster prevention**

  Wireless sensor networks can effectively act to prevent the consequences of natural disasters, like floods. Wireless nodes have successfully been deployed in rivers where changes of the water levels have to be monitored in real time.

- **Machine health monitoring**

  Wireless sensor networks have been developed for machinery condition-based maintenance (CBM) as they offer significant cost savings and enable new functionality. In wired systems, the installation of enough sensors is often limited by the cost of wiring. Previously inaccessible locations, rotating machinery, hazardous or restricted areas, and mobile assets can now be reached with wireless sensors.

➢ **Data logging**

Wireless sensor networks are also used for the collection of data for monitoring of environmental information, this can be as simple as the monitoring of the temperature in a fridge to the level of water in overflow tanks in nuclear power plants. The statistical information can then be used to show how systems have been working. The advantage of WSNs over conventional loggers is the "live" data feed that is possible.

2.4 Routing Techniques in Wireless Sensor Networks: A Survey

Wireless Sensor Networks (WSNs) consist of small nodes with sensing, computation, and wireless communications capabilities. Many routing, power management, and data dissemination protocols have been specially designed for WSNs where energy awareness is an essential design issue. The focus, however, has been given to the routing protocols which might differ depending on the application and network architecture. The routing techniques are classified into three categories based on the underlying network structure: flat, hierarchical, and location-based routing. The design tradeoffs between energy and communication overhead savings in every routing paradigm are studied.

Routing challenges and design issues that affect routing process in WSNs-

- Node deployment: Node deployment in WSNs is application dependent and affects the performance of the routing protocol. The deployment can be either deterministic or randomized. In deterministic deployment, the sensors are manually placed and data is routed through pre-determined paths. However, in random node deployment, the sensor nodes are scattered randomly creating an infrastructure in an ad hoc manner.

- Energy consumption without losing accuracy: sensor nodes can use up their limited supply of energy performing computations and transmitting information in a wireless environment. As such, energy-conserving forms of communication and computation are essential. Sensor node lifetime shows a strong dependence on the battery lifetime. In a multihop WSN, each node plays a dual role as data sender and data router. The malfunctioning of some sensor nodes due to power failure can cause significant topological changes and might require rerouting of packets and reorganization of the network.

- Fault Tolerance: Some sensor nodes may fail or be blocked due to lack of power, physical damage, or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network. If many nodes fail, routing protocols must accommodate formation of new links and routes to the data collection base stations.

- Scalability: The number of sensor nodes deployed in the sensing area may be in the order of hundreds or thousands, or more. Any routing scheme must be able to work with this huge number of sensor nodes. In addition, sensor network routing protocols should be scalable enough to respond to events in the environment.

2.5 Location-Aided Routing (LAR) in mobile ad hoc networks

A mobile ad hoc network consists of wireless hosts that may move often. Movement of hosts results in a change in routes, requiring some mechanism for determining new routes. An approach to utilize location information to improve performance of routing protocols for ad hoc networks is discussed. By using location information, the proposed *Location-Aided Routing* (LAR) protocols limit the search for a new route to a smaller "request zone" of the ad hoc network. This results in a significant reduction in the number of routing messages. Routes between two hosts in a Mobile Ad hoc NETwork (MANET) may consist of hops through other hosts in the network. Host mobility can cause frequent unpredictable topology changes. Therefore, the task of finding and maintaining routes in MANET is nontrivial. Approach to decrease overhead of route discovery by utilizing location information for mobile hosts is used. Such location information may be obtained using the global positioning system (GPS). How location information may be used by means of two *Location-Aided Routing* (LAR) protocols for route discovery is demonstrated.

The route discovery algorithm using flooding as, when a node S needs to find a route to node D, node S broadcasts a route request message to all its neighbours. Thereafter, node S will be referred to as the *sender* and node D as the *destination*. A node, say X, on receiving a route request message, compares the desired destination with its own identifier. If there is a match, it means that the request is for a route to itself (i.e., node X). Otherwise, node X broadcasts the request to its neighbours – to avoid redundant transmissions of route requests, a node X only broadcasts a particular route request once (repeated reception of a route request is detected using sequence numbers). Figure illustrates this algorithm. In this figure, node S needs to determine a route to node D. Therefore, node S broadcasts a route request to its neighbours. When nodes B and C receive the route request, they forward it to all their neighbours. When node F receives the route request from B, it forwards the request to its neighbours. However, when node F receives the same route request from C, node F simply discards the route request.

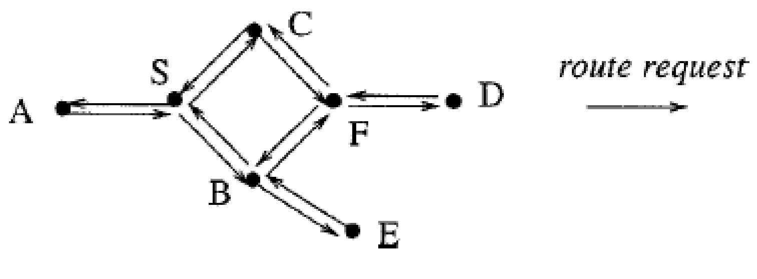


Figure 2.1: Routing Path

As the route request is propagated to various nodes, the path followed by the request is included in the route request packet. Using the above flooding algorithm, provided that the intended destination is reachable from the sender, the destination should eventually receive a route request message. On receiving the route request, the destination responds by sending a *route reply* message to the sender the route reply message follows a path that is obtained by reversing the path followed by the route request received by D

2.6 Asymmetric Encryption in Wireless Sensor Networks

Security requirements of WSNs are similar to conventional computer networks, therefore parameters such as confidentiality, integrity, availability and authenticity must be taken into account in creation of a network environment. Due to limitations of WSNs, not all security solutions designed for conventional computer networks can be implemented directly in WSN. For a long time, it was believed that the public key cryptography was not suitable for WSNs because it was required high processing power, but through studies of encryption algorithms based on curves was verified the feasibility of that technique in WSN.

The cryptographic algorithm RSA is currently the most used among the asymmetric algorithms, working from the difficulty of factoring large prime numbers. This algorithm is widely used in transactions on the Internet. Algorithms involved in this study are asymmetric, but each one works with a specific encryption mode.

In most of applications, sensor devices are spread over large areas, what difficult a individual control of network components. Moreover, wireless communication allows an attacker can trigger attacks without having physical access to the device, so attacks on WSNs can be divided into three main types: (1) Attack of authentication and confidentiality: Consists of attacks change, repetition or modification packages. (2) Availability network Attack: Generally known as DoS attacks or negation of service, this attack involves the application of techniques that make the network unavailable. (3)Attack on integrity: this type of attack the attacker's goal is to inject false data on the network, keeping the network available, but travelling fictitious data.

In RSA, to implement a public key cryptosystem whose security is based on the difficulty to be factoring large prime numbers. Through this technique it is possible to encrypt data and to create digital signatures. It was so successful that today is the RSA public key algorithm used most in the world. The encryption and decryption schemes are presented below. The decryption works because $c^{d} \equiv (m^{e})^{d} \equiv m(mod n)$. The safety lies in the difficulty of computing a clear text m from a ciphertext $cm^{e} mod n$ and the public parameters n (e). RSA public key algorithm is the most commonly used is standardized, and achieves efficiency relatively good.

<u>KEY GENERATION:</u>

i. Select two prime numbers p and q such that p not equal to q.
ii. Calculate n=p x q
iii. Calculate ø (n)=(p-1)(q-1)

iv. Select integer e such that

$\gcd(ø(n),e)=1; 1<e< ø(n)$

v. Calculate d such that $d=e^{1} \bmod ø(n)$

vi. Public key KU={e,n}

vii. Private key KR={d,n}

ENCRYPTION:

The plain text M (M<n) is encrypted to cipher text using public key e.
C=M pow e (mod n)

DECRYPTION:

The cipher text C is decrypted to plain text using private key d.
M=C pow d (mod n)

2.7 Coverage Problem in Wireless Sensor Network: A Survey

The coverage of WSN has answered the questions about quality of service (surveillance) which can be provided by WSN. Therefore, maximizing coverage using the resource constrained nodes is a non-trivial problem. Generally, there are many different criterions (factors) can affect the coverage performance of WSN.

Deployment strategy: random versus deterministic. A Deterministic sensor placement can be applied to a small to medium sensor network in a friendly environment and random deployment, where sensor nodes are distributed within the field stochastically and independently.

Sensing model: there are two mainly two different sensing models: one is Boolean sensing model where each sensor has a fixed sensing area and a sensor can only sense the environment or detect events within its sensing area and In reality, sensor detection is imprecise hence needs to be expressed in probabilistic terms.

Algorithm Characteristics: a coverage scheme can operate in either a centralized or distributed. In centralized, information from all nodes needs to be transferred to the central node. In distributed (localized) scheme, the coverage algorithm is executed based on information from only some nodes in WSN, and the decision is made locally.

Sensor mobility: the coverage performance of stationary sensor network can be determined by the initial network configuration, and it remains unchanged over time after deployment. Contrarily, mobile sensor network can improve or maintain coverage performance by sensor mobility. It is extremely valuable in situations where deployment mechanisms fail or coverage maintenance.

Current researches in coverage of WSN focus on evaluating the coverage performance and improving the coverage performance with some of the primary limitations:

Firstly, in the literature most existing works assume that the boundary of sensing and communication of sensor node is a perfect circle which is a known and static radius in considering the coverage problem. However, the sensing ranges are very irregular and dynamic in real situation for wireless sensor network, which usually employ low quality radio modules to reduce the cost.

Secondly, in applications where nodes move around in a certain pattern, mobility could further be exploited to improve coverage and connectivity. On the one hand, mobility poses challenges in guaranteeing coverage at all times, while on the other hand, it enables nodes to cover areas that would have been left uncovered using only static nodes.

Although many schemes have been proposed and progress has been made in coverage problems of WSN, there are still many open research issues. Effective coverage scheme should be proposed to implement real applications but limited to theoretical study. Therefore, most existing centralized solutions need to be developed include the distributed and localized algorithms or protocols.

Chapter 3: Implementation

3.1 Feasibility Study

“FEASIBILTY STUDY” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources. The feasibility of the project is analyzed in this phase. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed environment must be feasible. For feasibility analysis, some understanding of the major requirements for system is essential.

The key objective of the feasibility study is to weigh up two types of feasibility. They are:

- Operational feasibility
- Technical feasibility

3.1.1 Operational feasibility

Operational feasibility is necessary as it ensures that the project developed is a successful one. As the execution process of the proposed work is very much user friendly, the operational feasibility of the project is high.

3.1.2 Technical feasibility

Technical feasibility analysis makes a comparison between the level of technology available and that is needed for the project development of the project. The level of technology consists of the factors like software tools, machine environment, and platform developed and so on.

3.2 Requirement Analysis:

To create an environment where sensors are randomly deployed and sensors in the environment can detect the moving object in wireless sensor network.

3.2.1 Software Interface

Describes the connections between this product and other specific software components, including data bases, operating systems, tools, libraries, and integrated commercial components.

Identify the data items or messages coming into the system into and going out and describe the purpose of each. Describe the services needed and the nature of the communications.
Language: Java

Front End Tool: Swing

Operating System: Windows 7.

3.2.2 Hardware requirements

Describes the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of data and control interactions between the software and hardware, communication protocols to be used.

Processor	: Pentium IV or any higher version
Memory	: 1GB RAM (Min)
Hard Disk	: 10GB (Min)
Input Device	: Keyboard, mouse and other peripherals

3.3 Modules

This project is divided into 2 modules.

1. Password module
2. Request module

1. Password module:

This module authenticates the user to enter. The user can type the text they want to send or browse the file they want to attach .this message is transferred to the destination after the data is encrypted.RSA algorithm is used for encryption.

2. Request module:

This module describes the wireless network communication protocol mechanism

3.4 Block Diagram

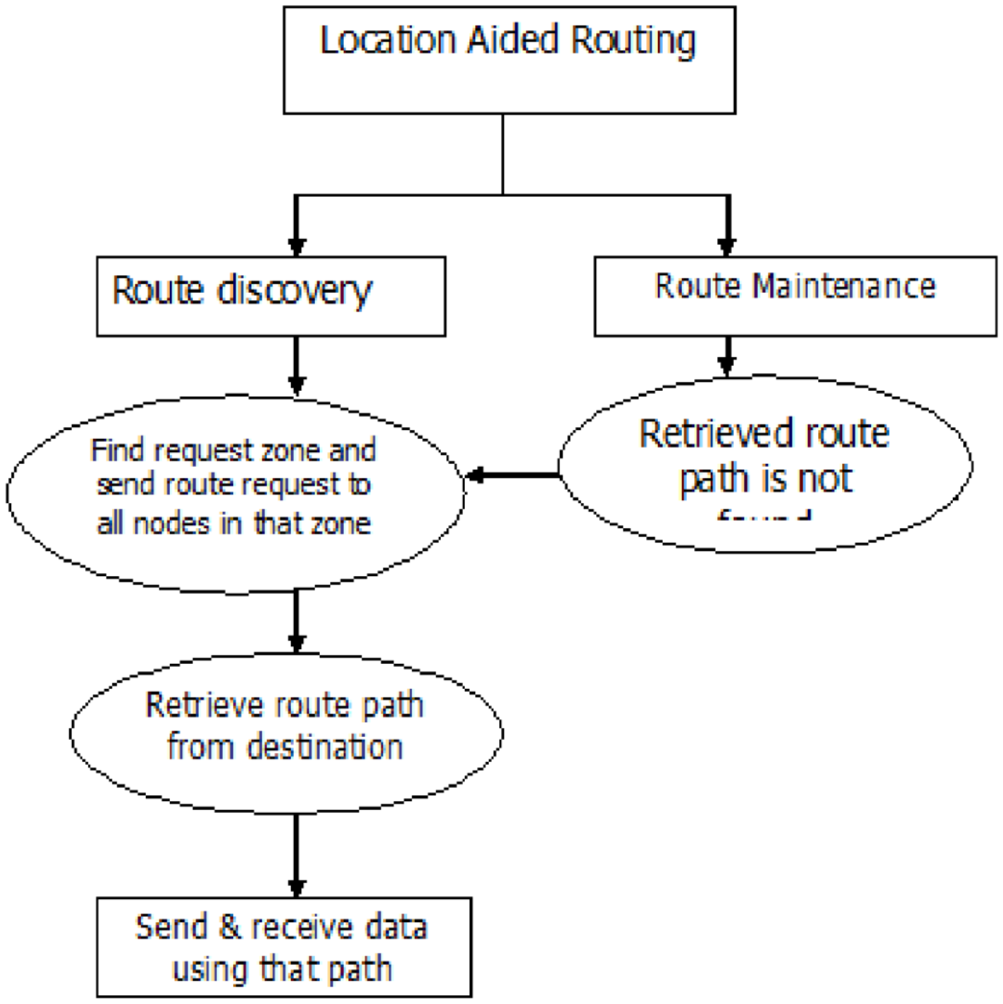

Location Aided Routing
Route discovery
Route Maintenance
Find request zone and send route request to all nodes in that zone
Retrieved route path is not
Retrieve route path from destination
Send & receive data using that path

Figure 3.1: Block Diagram

In the above figure, block diagram is representing the steps of implementation with methodologies used during implementation.

3.5 Activity Diagram:

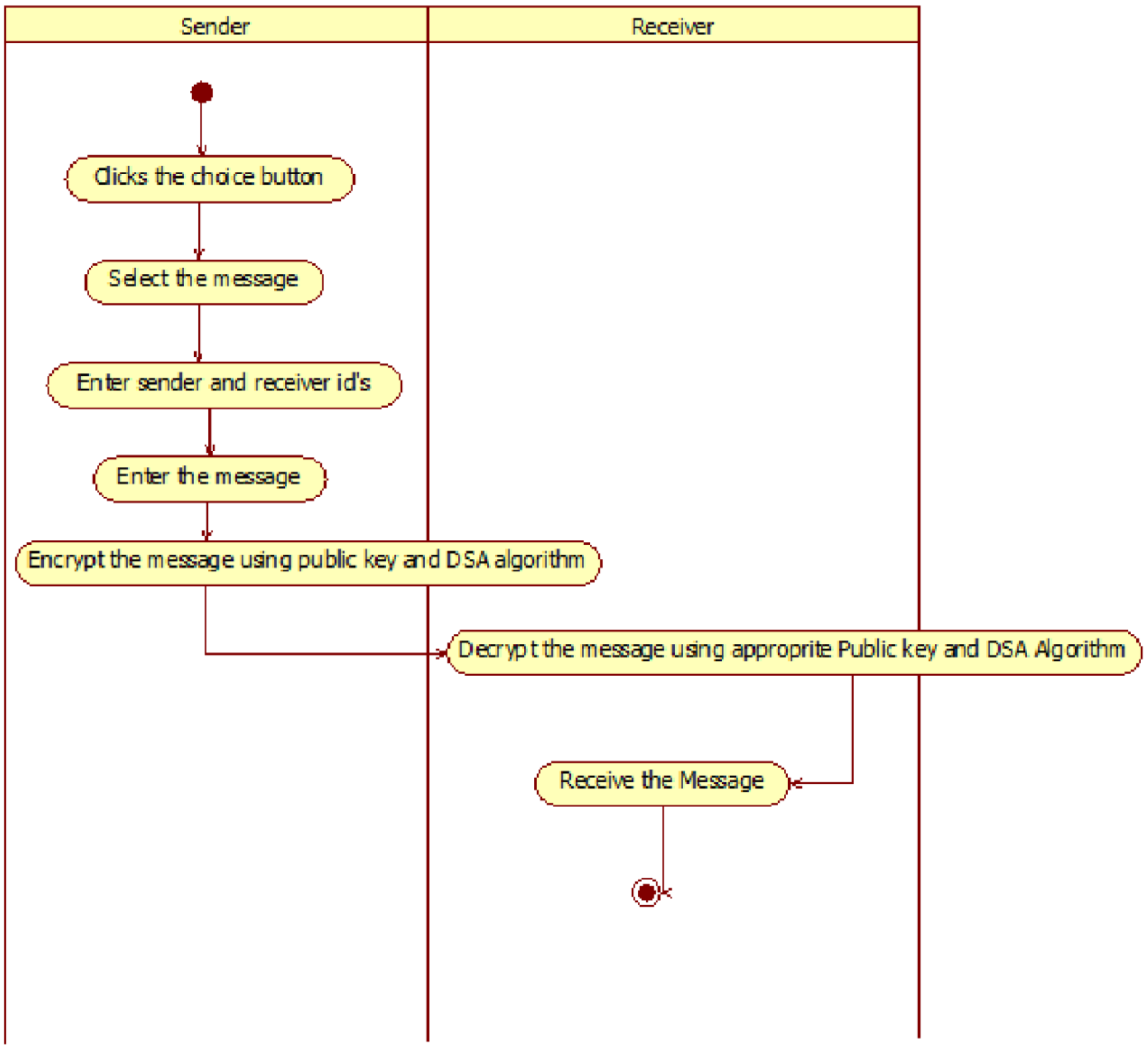

Sender
Receiver
Clicks the choice button
Select the message
Enter sender and receiver id's
Enter the message
Encrypt the message using public key and DSA algorithm
Decrypt the message using approprite Public key and DSA Algorithm
Receive the Message

Figure 3.2: Activity Diagram

Activity diagram is graphical representation of workflow of stepwise activities and actions with support for choice, iteration and concurrency. Activity diagram shows the overall flow of control between targets to base station.

3.6 Sequence Diagram:

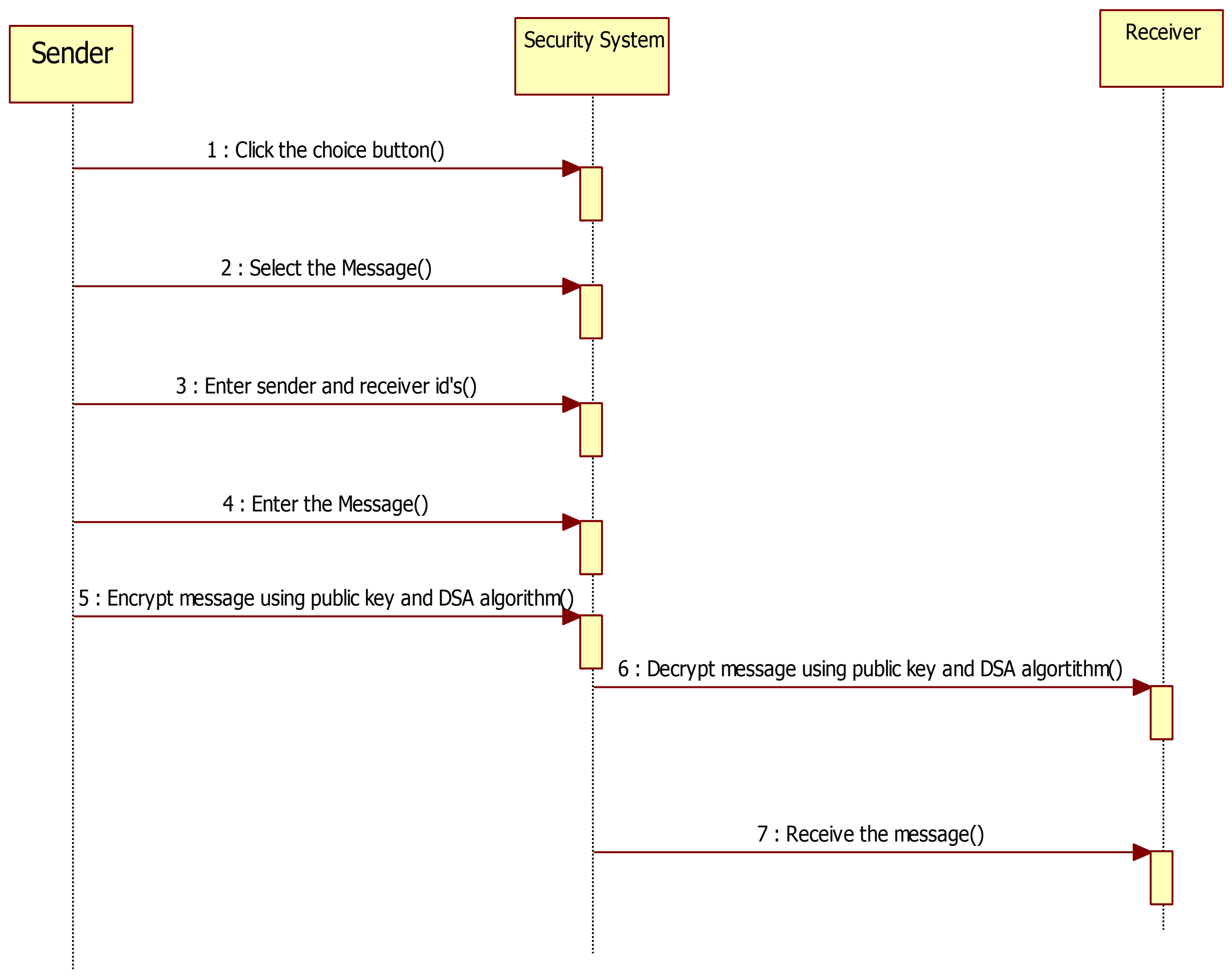

Sender
Security System
Receiver
1 : Click the choice button()
2 : Select the Message()
3 : Enter sender and receiver id's()
4 : Enter the Message()
5 : Encrypt message using public key and DSA algorithm()
6 : Decrypt message using public key and DSA algortithm()
7 : Receive the message()

Figure 3.3: Sequence Diagram

In the above sequence diagram, we can see the sensor senses the data and then send the collected information to the cluster head. After that local cluster head aggregates the data from local cluster nodes and transmit it to the base station in encrypted form. Base station decrypt the collected data after verification.

3.7 Use Case Diagram

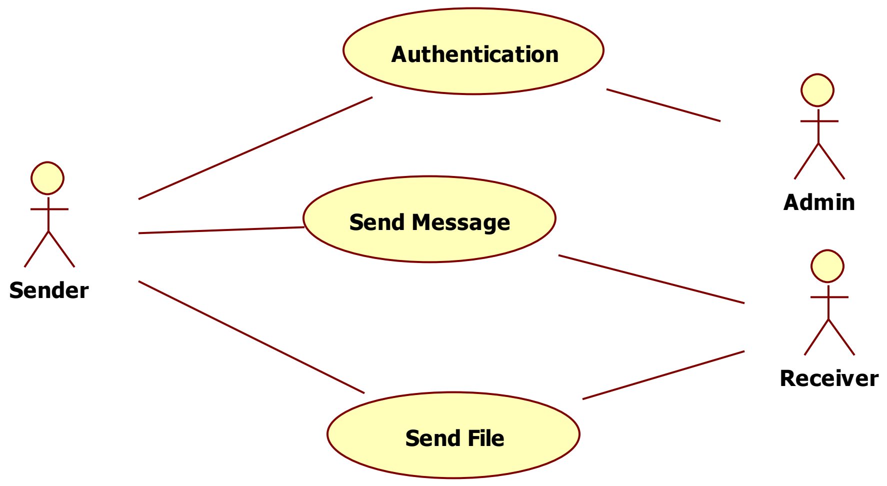


Figure3.4: Use Case diagram

Use Case Diagram describes the way in which components of the designed system are interacting with each other, sensing nodes send the data of the target to its respective cluster head which assembles the data and transmit to base station.

3.8 Data Flow Diagram:

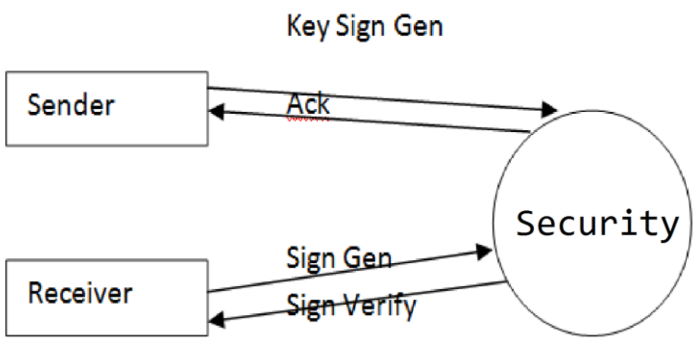


Figure 3.5: Data Flow Diagram

3.9 Simulation

Password.java

```
import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

import java.sql.*;

public class Password  extends JFrame implements ActionListener

{

                                JPasswordField password;

                                JTextField Text1;

                                JLabel login,pass,hea1;

                                JButton ok,cancel;

                                final String log = "routing";

                                final String passW = "location";

                                JFrame fpass;
```

```
Password()
 {


    fpass = new JFrame("Login");
     fpass.getContentPane().setLayout(null);
    hea1 = new JLabel("LOG IN SCREEN");
    login = new JLabel("USER-ID");
    Text1 = new JTextField(10);
    pass = new JLabel("PASSWORD");
    password = new JPasswordField(10);
    ok = new JButton("OK");
    cancel = new JButton("CANCEL");
    hea1.setBounds(130,10,100,50);
    login.setBounds(70,40,70,50);
    Text1.setBounds(157,55,125,20);
    pass.setBounds(70,70,70,50);
    password.setBounds(158,83,125,20);
    ok.setBounds(80,130,90,20);
    cancel.setBounds(180,130,110,20);
    fpass.getContentPane().add(hea1);
    fpass.getContentPane().add(login);
    fpass.getContentPane().add(Text1);
```

```
                                        fpass.getContentPane().add(pass);

                                        fpass.getContentPane().add(password);

                                        fpass.getContentPane().add(ok);

                                        fpass.getContentPane().add(cancel);

                                        ok.addActionListener(this);

                                        cancel.addActionListener(this);

                                        fpass.setSize(400, 250);

                                        fpass.setVisible(true);


                                      }

                                      public void actionPerformed(ActionEvent e)

                                      {

                                           String s = e.getActionCommand();

                                           if(s.equals("OK"))

                                           {

try

        {

                                     int count = 1;

                                     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

                                     Connection con =
                          DriverManager.getConnection("Jdbc:Odbc:connection");

                                     Statement st=con.createStatement();

                                     String str = "select * from login";
```

```
                                        ResultSet rs = st.executeQuery(str);

                                        while(rs.next())

                                        {

                                        if(((Text1.getText()).equals(rs.getString(1))) &&

                                          ((password.getText()).equals (rs.getString(2))))

                                            {

                                                 count = 0;

                                                 break;

                                             }


                                         else

                                                 count = 1;

                                        }



          if(count ==1)

      {

                                                              JOptionPane.showMessageDialog(null,
"Invaild UserName and PassWord","Alert", JOptionPane.ERROR_MESSAGE);

                                                              Text1.setText("");

                                                              password.setText("");

                                                              fpass.setVisible(true);

                                                        }

else
```

```
                {
                                                  fpass.setVisible(false);
                                                   Sender sen=new Sender();
        }


        }catch(Exception e1){System.out.println(e1);}
        }
                                                     else if(s.equals("CANCEL"))
                                                      {
                                                              System.exit(0);
                                                      }
                                                  }
                                                  public static void main(String ar[])
                                                  {
                                                     Password ps = new Password();
                                                     ps.addWindowListener( new WindowAdapter()
                                                             {
                                                              public void windowClosing(WindowEvent e)
                                                              {
                                                                System.exit(0);
                                                              }
                                                             }
```

```
                                                        );
 }
}
```

Sender.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import java.util.*;
class Sender implements ActionListener
{

                JFrame jf;
                static JTextArea jta=new JTextArea();
                JButton jb,browse;
                JTextField jt1,jt2,jt3;
                JLabel jl1,jl2,jl3,jl4;
                JScrollPane jsp;
                Container c;
                static String destination="";
                static String ttt;
                static String tte;
                static Vector vvv=new Vector();
  static int i=0;
```

```
Sender()

{

    jf=new JFrame("Sender");

     jb=new JButton("Send");

     jt1=new JTextField();

     jt2=new JTextField();

     jt3=new JTextField();

     jl1=new JLabel("Destination");

     browse=new JButton("Browse");

     jl2=new JLabel("Request Zone Limit");

     jsp=new
JScrollPane(jta,ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,ScrollPaneConstants.HORI
ZONTAL_SCROLLBAR_AS_NEEDED);


     jf.setSize(800,800);

     c=jf.getContentPane();

     c.setLayout(null);

     c.add(jsp);

     c.add(jb);

     c.add(browse);

     c.add(jt1);

     c.add(jl1);

     c.add(jl2);

     jsp.setBounds(100,100,400,400);

     jb.setBounds(550,350,100,25);

     jt1.setBounds(550,150,100,25);

     browse.setBounds(550,250,100,25);
```

```
                              jl1.setBounds(550,100,100,25);
                              jb.addActionListener(this);
                              browse.addActionListener(this);
                              jf.setVisible(true);

}
public void actionPerformed(ActionEvent ae)
{

                            destination=jt1.getText();
                            ttt=jta.getText();
                          try
                            {
                                    FileOutputStream fop=new FileOutputStream("msg.txt");
                                    fop.write(ttt.getBytes());
                                    ObjectOutputStream oop=new ObjectOutputStream(fop);
                            }
                            catch(Exception e){}
      // vvv.add(ttt);
                            System.out.println("The vect size is "+vvv.size());
                            if(jb==ae.getSource())
                            {
                              try
                              {
                                    System.out.println("The String sent is :"+ttt);
                                    RSAKey rsa=new RSAKey();
```

```
                    }
                    catch(Exception e)
                    {
                            System.out.println("Error  : "+e);
                    }
                   }
                   if((JButton)browse==ae.getSource())
    {
                            JFileChooser fc = new JFileChooser();
                            int option = fc.showOpenDialog(jf);
                            if(option == JFileChooser.APPROVE_OPTION)
                            {
                                  try
                                  {
                                        String sf=fc.getSelectedFile().getAbsolutePath();
                                        FileInputStream in = new FileInputStream(sf);
                                        byte str[] = new byte[in.available()];
                                        in.read(str,0,str.length);
                                        jta.setText(new String (str));
                                  }
                                  catch(Exception e){}
                            }
                    }
 }

public static void Socket2(int y,String tt)throws Exception
```

```
{
                FileInputStream fis=new FileInputStream("path"+y+".txt");
                ObjectInputStream ois=new ObjectInputStream(fis);
                Vector d=(Vector)ois.readObject();
                int siz=d.size();
                System.out.println("The size of the vector is ::"+siz);
                for(int j=0;j<siz;j++)
                {
                  System.out.print("++__++-->>"+d.get(j));


                }
                if(siz==2)
                {


                  Socket soc=new Socket((String)d.get(1),8888);
                  ObjectOutputStream oos=new ObjectOutputStream(soc.getOutputStream());
                  oos.writeObject("file");
                  oos.writeObject(tt);
                }
                else
                {
                  d.removeElementAt(0);
                  int m=d.size();
                  System.out.println("The size after removing the element..."+m);
                  Socket soc=new Socket((String)d.get(1),8888);
                  ObjectOutputStream oos=new ObjectOutputStream(soc.getOutputStream());
```

```
                        oos.writeObject("file++");
                        oos.writeObject(tt);
                        oos.writeObject(d);

                    }

    }
}
```

RSA KEY.java

```
import javax.swing.*;
import java.awt.event.*;
class RSAKey implements ActionListener
{
        JButton jb,jb1,jb2;
        JFrame jf;

        RSAKey()
        {
                jf=new JFrame("RSA KEY");
                JInternalFrame jf1=new JInternalFrame("KEY Value");
                jb=new JButton("RSAKeyGen");
                jb1=new JButton("Send");
                jb2=new JButton("Exit");
                JPanel jp=new JPanel();
                JPanel jp1=new JPanel();
                jp.add(jb);
```

```
		jp.add(jb1);

		jp.add(jb2);

		jb.addActionListener(this);

		jb1.addActionListener(this);

		jb2.addActionListener(this);

		jp.setLayout(null);

		jb.setBounds(90,100,120,25);

		jb1.setBounds(50,180,75,25);

		jb2.setBounds(140,180,75,25);

		jf1.setContentPane(jp);

		jp1.setLayout(null);

		jf1.setBounds(30,30,180,180);

		jf1.setVisible(true);

		jf.setContentPane(jf1);

		jf.setSize(300,300);

		jf.setVisible(true);

	}

	public void actionPerformed(ActionEvent ae)

	{

		if(ae.getSource()==jb)

		{

			RSAKeyDsgn rsad=new RSAKeyDsgn();

		}

		if(ae.getSource()==jb1)

		{

			try
```

```
				{
					new Timers();
					jf.setVisible(false);
					EnRSA en=new EnRSA();
					en.WriteRSA(keyrsa1.d,keyrsa1.n);
				}
				catch(Exception ex){}
			}
			if(ae.getSource()==jb2)
			{

				jf.setVisible(false);
			}
		}
		public static void main(String arg[])
		{
			RSAKey rsa=new RSAKey();
		}
}
```

RSAKeyDesgn.java

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.FileInputStream;
import java.io.*;
import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;

import javax.swing.*;

public class RSAKeyDsgn extends JFrame implements ActionListener

{

        private JLabel pval,qval,rsa;

        private JTextField pfield,qfield;

        private JButton ok,cancel;

        String pstr,qstr;

        JFrame frsa;

        RSAKeyDsgn()

         {


                frsa = new JFrame("RSA KeyGeneration");

                JPanel jp=new JPanel();

                jp.setLayout(null);

                rsa = new JLabel("RSA KEYGENERATION");

                pval = new JLabel("First Prime Number (P)");

                qval = new JLabel("Second Prime Number (Q)");

                pfield = new JTextField(20);

                qfield = new JTextField(20);

                ok = new JButton("OK");

                cancel = new JButton("CANCEL");
```

```
 jp.add(pval);
 jp.add(qval);
 jp.add(pfield);
 jp.add(qfield);
 jp.add(ok);
 jp.add(cancel);
rsa.setBounds(140,20,140,20);
pval.setBounds(45,55,150,20);
pfield.setBounds(210,55,150,20);
qval.setBounds(45,90,150,20);
qfield.setBounds(210,90,150,20);
ok.setBounds(110,150,90,20);
cancel.setBounds(210,150,90,20);
frsa.setContentPane(jp);
ok.addActionListener(this);
cancel.addActionListener(this);
frsa.setSize(450,250);
frsa.setVisible(true);
frsa.addWindowListener( new WindowAdapter()
{
        public void windowClosing(WindowEvent e)
        {
                frsa.setVisible(false);
        }
}
       );
	}


	public void actionPerformed(ActionEvent arg0)
	{
	   if(arg0.getSource()==ok)
	   {
			pstr=pfield.getText();
			qstr=qfield.getText();
			try
			{
				if(Integer.parseInt(pstr) >=100 || Integer.parseInt(qstr) >=100)
				{
					JOptionPane.showMessageDialog(null,"You Have Entered into HIGH
LEVEL TRUST ","LEVEL OF TRUST",JOptionPane.INFORMATION_MESSAGE);
					try{
					FileOutputStream fos=new FileOutputStream("trust");
					   ObjectOutputStream oos=new ObjectOutputStream(fos);
						oos.writeObject("high");}
						catch(Exception e){}
				}
				else if(Integer.parseInt(pstr) >=25 || Integer.parseInt(qstr) >=25)
				{
					JOptionPane.showMessageDialog(null,"You Have Entered into
MEDIUM LEVEL TRUST ","LEVEL OF TRUST",JOptionPane.INFORMATION_MESSAGE);
					try{
					FileOutputStream fos=new FileOutputStream("trust.txt");
					   ObjectOutputStream oos=new ObjectOutputStream(fos);
```

```
                                        oos.writeObject("medium");}
                                        catch(Exception e){}
                                }
                                else
                                {
                                        JOptionPane.showMessageDialog(null,"You Have Entered into LOW
LEVEL TRUST ","LEVEL OF TRUST",JOptionPane.INFORMATION_MESSAGE);
                                        try{
                                        FileOutputStream fos=new FileOutputStream("trust");
                                          ObjectOutputStream oos=new ObjectOutputStream(fos);
                                                oos.writeObject("low");}
                                                catch(Exception e){}
                                }
                                FileInputStream fis=new FileInputStream("msg.txt");
                                byte str[]=new byte[fis.available()];
                                fis.read(str,0,str.length);
                                String str1=new String(str);
                          System.out.println(str1);
                                keyrsa1 k1=new keyrsa1(str1);
                                k1.Key(pstr,qstr);
                                frsa.setVisible(false);
                                }
                        catch(Exception ew){}
                        }
   }
}
```

EnRSA.java

```
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.*;
import java.net.*;
import javax.swing.JOptionPane;
public class EnRSA
{
  static String s,str="",ch="";
  long e,n,c,f,data;
  int k,i;
  String estr,nstr;
  public void GetRSA(long es, long ns,String s) throws IOException
  {
                          System.out.println(s);
                          byte b[]=s.getBytes();
                          FileOutputStream fos=new FileOutputStream("data1.txt");
                          fos.write(b);
                          e=es;
                          n=ns;
                          FileInputStream file=new FileInputStream("data1.txt");
      System.out.println("The value for e :"+e);
      System.out.println("The value of n :"+n);
                         while((data=file.read())!=-1)
```

```
                    {
                         if(Math.max(data,n) == data)
                         {
                           JOptionPane.showMessageDialog(null,"To Encrypt Message is too Big, must
(Message < n)","Error",1);
                           System.exit(0);
                         }
                         if ( e % 2 == 0)
                         {

                           c = 1;
                          for ( i = 1; i <= e/2; i++)
                          {
                            f = (data*data) % n;
                            c = (f*c) % n;
                          }
                         }
                         else
                         {

                            c = data;
                            for (  i = 1; i <= e/2; i++)
                            {
                                    f = (data*data) % n;
                                    c = (f*c) % n;
                            }
```

```
                }
                k=(int)c;
                str=Long.toString(c);
                ch=ch+str+" ";
        }
        System.out.println(ch);
        file.close();
        }

        public void WriteRSA(long d,long n) throws IOException
        {
          try
        {
                String s=d+"";
                String s1=n+"";
           FileWriter put = new FileWriter("rsain.txt");
           put.write(ch);
           put.close();
           System.out.println("Destination Name:"+Sender.destination);
           Send.send(Sender.destination);

        }
        catch(Exception es){}
  }
}
```

DERSA.java

```
import java.io.BufferedReader;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.*;

public class DERSA

{


                    static String s,sn1="";

                    int info[] = {48,49,50,51,52,53,54,55,56,57};

                    long a[]    = {0,1,2,3,4,5,6,7,8,9};

                    long d,n,g,f,value,temp=0;

                    int k,l=0,data,i;

                    char c[] = new char[1000];

                    String dstr,nstr;


                    public void ProcessDecrp(long ds,long ns,String sn) throws IOException

                    {

                     d=ds;

                     n=ns;

                     byte b[]=sn.getBytes();

                     FileOutputStream fos=new FileOutputStream("rsain.txt");

                     fos.write(b);

                     FileInputStream f0=new FileInputStream("rsain.txt");

                     FileOutputStream put = new FileOutputStream("Output.txt");
```

```
while((data=f0.read()) != -1)
{
            System.out.println("Data:"+data);
            value=0;
            for(i=0;i<info.length;i++)
                        {
                          if(data==info[i])
                        temp=a[i];
                        }
                        value=(value*10)+temp;
                        data=f0.read();

                        while(data!=32)
                        {
                         for(i=0;i<info.length;i++)
                         {
                          if(data==info[i])
                         temp=a[i];
                         }
                         value=(value*10)+temp;
                         data=f0.read();
                        }
                        if ( d % 2 == 0)
                        {
                          g = 1;
                          for ( i = 1; i <= d/2; i++)
                          {
                         f = (value*value) % n;
                         g = (f*g) % n;
                         System.out.println("if:"+g);
                          }
                        }
                        else
                        {
                          g = value;
                          for (  i = 1; i <= d/2; i++)
                          {
                         f = (value*value) % n;
                         g = (f*g) % n;
                         System.out.println("else:"+g);
                          }
                        }
                        char k1=(char)g;
                        sn1=sn1+k1;
                        put.write(sn1.getBytes());
                }
                Recv.ja.setText(sn1);
                put.close();
  }


}
```

Recv.java

```
import java.io.*;

import java.net.*;

import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

class Recv implements ActionListener

{

        JFrame jf;

        static JTextArea ja;

        JButton jb;

        JLabel jl;

        JScrollPane jsp;

        static String sr,sr1,sr2;

        static long d,n;

        Recv()

        {

                jf=new JFrame("Receiver");

                jl=new JLabel("Receive the Message");

                ja=new JTextArea();

                jsp=new JScrollPane(ja);

                jb=new JButton("Receive");

                JPanel jp=new JPanel();

                jp.add(jb);

                jp.add(jsp);

                jp.add(jl);
                jb.addActionListener(this);

                jp.setLayout(null);

                jl.setBounds(40,60,150,25);

                jsp.setBounds(40,90,250,300);

                jb.setBounds(100,420,100,25);

                jf.setContentPane(jp);

                jf.setSize(350,500);

                jf.setVisible(true);

        }

public void actionPerformed(ActionEvent ae)

        {

                if((JButton)jb==ae.getSource())

                {

                        RSAKeyDecry dn=new RSAKeyDecry();

                }

        }

}
```

Req.java

```
import java.net.*;

import java.util.*;

import java.io.*;

import javax.swing.*;

public class Req

{


        static Socket s1,s2,d;
        static Vector localvect=new Vector(10);

        static Vector localvecttemp=new Vector(10);

        static Vector recvect=new Vector(10);

        static int i=0;

        static String file="";

        static Vector vv=new Vector();

        static String mm;

        static String tte;

        static Sender sen;

        static String jdd;

        static String dd;

        public static void main(String a[]) throws Exception

        {


                String path="";

                Enumeration it;

                String sender="";

                String localaddr=(InetAddress.getLocalHost()).getHostName();

                String destination="";

                Socket s1,s2,d;

                String sockaddr="";

                String next="";

                int vectsize;


    try

            {
                ServerSocket ss=new ServerSocket(8888);
                new tab();
while(true)
                {
                        System.out.println("Connected ");
                        tab.setTable("Connected ");

                   System.out.println(localaddr+" is Waiting....... ");
                   tab.setTable(localaddr+" is Waiting....... ");
                        Socket s=ss.accept();
                        InputStream ins=s.getInputStream();
                        OutputStream ous=s.getOutputStream();
                        ObjectOutputStream oos=new ObjectOutputStream(ous);
                        ObjectInputStream ois=new ObjectInputStream(ins);
                        FileReader fr=new FileReader("nextnodes.txt");
                        BufferedReader br=new BufferedReader(fr);
                        String fd=(String)ois.readObject();
                         if(fd.equals("reqroute"))
              {
                                        System.out.println("Route Request...");
                                        tab.setTable("Route Request...");
                                        localvect=(Vector)ois.readObject();
                                        sender=(String)localvect.firstElement();
                                        System.out.println("The Sender address is : "+sender);
                                        tab.setTable("The Sender address is : "+sender);
```

```
destination=(String)localvect.get(1);
                                        System.out.println("The Destined address is : "+destination);
                                        tab.setTable("The Destined address is : "+destination);
                                        System.out.println("The Local Address is : "+localaddr);
                                        tab.setTable("The Local Address is : "+localaddr);
                                        if(destination.equalsIgnoreCase(localaddr))
                                        {
                                                req(localvect);
}
else
{
                                                while((next=br.readLine())!=null)
                                                {

                                                  try
                                                  {
                            s1=new Socket(next,8888);
                            OutputStream ous1=s1.getOutputStream();
                            ObjectOutputStream oos3=new ObjectOutputStream(ous1);
                            localvect.addElement(localaddr);
                                    oos3.writeObject("reqroute");
                                    oos3.writeObject(localvect);
                                                }
                                                catch(Exception e)
                                                {
                                                                System.out.println("Error in file "+e);
                                                   tab.setTable("Error in file "+e);
                                          }



                                                }
                                        }


                   }
                 else if(fd.equals("ack"))
                   {
                                        Timers.rece();
                                        System.out.println("Acknowledgement from destination ...");
                                        tab.setTable1("Acknowledgement from destination ...");
                                        localvect=(Vector)ois.readObject();
                                        it=localvect.elements();
                                        while(it.hasMoreElements())
                                        {
                                                path=(String)it.nextElement();
                                                System.out.print(" "+path+" --> ");
                                                tab.setTable1(" "+path+" --> ");
                                        }
                                  System.out.println();
                                    if(((String)localvect.firstElement()).equalsIgnoreCase(localaddr))
                                    {
                      i++;
                                                System.out.println("Acknowledgement sent from the
              destination ....");
```

```
}

                    tab.setTable1("Acknowledgement sent from the
destination ....");

                    System.out.println("The sender can send the data
now....");

                    tab.setTable1("The sender can send the data now....");

                    String destin=(String)localvect.remove(1);

                    localvect.add(destin);

                    it=localvect.elements();

                    System.out.println("Source  ");

                    tab.setTable1("Source ");

                    System.out.println("The path from source to destination
is as follows :");

                    tab.setTable1("The path from source to destination is as
follows :");

                    while(it.hasMoreElements())

                    {

                        path=(String)it.nextElement();

                        System.out.println(" "+path+" --> ");

                        tab.setTable1(" "+path+" --> ");


                    }

                    new Store_Path(i,localvect);

                    Store_Path.file();

                    System.out.println("   Destination ");

                    tab.setTable1("    Destination   ");

                    Socket1();


                }
```

```
                                        else if(((String)localvect.get(2)).equalsIgnoreCase(localaddr))
                                        {

                                              localvect.remove(localvect.size()-1);



                                              s2=new Socket((String)localvect.firstElement(),8888);
                                              OutputStream os2=s2.getOutputStream();
                                              ObjectOutputStream ous2=new
ObjectOutputStream(os2);
                                              ous2.writeObject("ack");
                                              FileOutputStream foo=new
FileOutputStream("paths.txt");
                                              ObjectOutputStream oopp=new
ObjectOutputStream(foo);
                                              oopp.writeObject(localvect);
                                              ous2.writeObject(localvect);
                                        }

                                        else

                                        {
                              localvect.remove(localvect.size()-1);
                                              sockaddr=(String)localvect.get(localvect.size()-1);
                              Enumeration en=localvect.elements();
                          System.out.println("The next system address is "+sockaddr);
```

```
                                        tab.setTable1("The next system address is "+sockaddr);
                                        s2=new Socket(sockaddr,8888);
                                        System.out.println("Intermediate node "+localaddr+"
forwarding the ack to "+sockaddr);
                                        tab.setTable1("Intermediate node "+localaddr+"
forwarding the ack to "+sockaddr);
                                        OutputStream os2=s2.getOutputStream();
                                        ObjectOutputStream ous2=new
ObjectOutputStream(os2);
                                        ous2.writeObject("ack");
                                        ous2.writeObject(localvect);

                              }
                    }

                    else if(fd.equals("file++"))

                    {
                              i++;
                              String sub=(String)ois.readObject();
                              vv=(Vector)ois.readObject();
                              System.out.println("The String sent is "+sub);
                              new Store_Path(i,vv);
                              Store_Path.file();
                              Sender.Socket2(i,sub);
                    }
                    else if(fd.equals("file"))
                    {
```

```
					JOptionPane.showMessageDialog(null,"File 
Received","File",JOptionPane.INFORMATION_MESSAGE);

					dd=(String)ois.readObject();

					System.out.println("The message sent from source is ::"+dd);

					tab.setTable1("The message sent from source is ::"+dd);

					Recv re=new Recv();

					FileInputStream pa=new FileInputStream("paths.txt");

					ObjectInputStream ooo=new ObjectInputStream(pa);

					Vector n=(Vector)ooo.readObject();

					for(int h=0;h<n.size();h++)

					{

						System.out.println("The value inside vector is 
"+n.get(h));

						tab.setTable("The value inside vector is  "+n.get(h));

					}

					if(n.size()==2)

					{

						Socket ssp=new Socket((String)n.get(0),8888);

						ObjectOutputStream oop=new 
ObjectOutputStream(ssp.getOutputStream());

						oop.writeObject("ackfile");

						oop.writeObject("The message \""+dd+"\" has received 
by the destination "+(String)n.get(1));

					}

					else

					{
```

```
                                        Socket sss=new Socket((String)n.lastElement(),8888);

                                        n.removeElementAt(n.size()-1);

                                        ObjectOutputStream oop1=new
ObjectOutputStream(sss.getOutputStream());

                                        oop1.writeObject("ackfile1");

                                        oop1.writeObject("The message \""+dd+"\" has received
by the destinaton "+(String)n.get(1));

                                        oop1.writeObject(n);


                                }


                        }

                        else if(fd.equals("ackfile"))

                        {

                                jdd=(String)ois.readObject();

                                System.out.println(jdd);

                                tab.setTable1(jdd);

                        }

                        else if(fd.equals("ackfile1"))

                        {

                                String ds=(String)ois.readObject();

                                Vector vl=(Vector)ois.readObject();

                                if(vl.size()==2)

                                {

                                        Socket sspd=new Socket((String)vl.get(0),8888);

                                        ObjectOutputStream oop=new
ObjectOutputStream(sspd.getOutputStream());
```

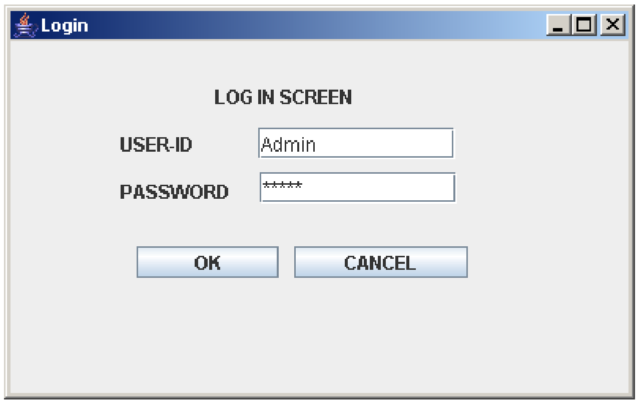
```
                                        oop.writeObject("ackfile");
                                        oop.writeObject(ds);

                                }
                                else
                                {
                                        Socket ssj=new Socket((String)vl.lastElement(),8888);
                                        vl.removeElementAt(vl.size()-1);
                                        ObjectOutputStream oop1=new
ObjectOutputStream(ssj.getOutputStream());
                                        oop1.writeObject("ackfile1");
                                        oop1.writeObject(ds);
                                        oop1.writeObject(vl);

                                }

                        }
                }

            }
            catch(Exception e)
            {
                        System.out.println("Some Error has occurred : "+e);
                        tab.setTable("Some Error has occurred : "+e);
                }

        }
```

```
public static void req(Vector v)
{
            String b="";
            try
            {
                int o=v.size();
             if(o==2)
             {
                                b=(String)v.firstElement();
                        }
             else
 {
                              b=(String)(v.lastElement());
                        }
                        Enumeration en=v.elements();
                        while(en.hasMoreElements())
                        {
                              String f=(String)en.nextElement();
                              System.out.println("The path from source to destination  : "+f);
                              tab.setTable1("The path from source to destination  : "+f);
             }
                        d=new Socket(b,8888);
                        OutputStream os=d.getOutputStream();
                        ObjectOutputStream oos1=new ObjectOutputStream(os);
                        oos1.writeObject("ack");
                String ss="";
                localvect=v;
              FileOutputStream foo=new FileOutputStream("paths.txt");
        ObjectOutputStream oopp=new ObjectOutputStream(foo);
                oopp.writeObject(localvect);
                boolean boo=true;
                if(o==2)
                        {}
                else
        {
                        for(int i=2;i<o;i++)
                        {
                          v.add((String)localvect.get(i));
                        }
        }


                oos1.writeObject(v);


          }
          catch(Exception e)
          {
                        System.out.println("Error at ack sending in destination "+e);
                        tab.setTable1("Error at ack sending in destination "+e);
                }
}
```

```
        public static void Socket1()throws Exception
    {
                System.out.println("Inside Socket1");
                FileInputStream fip=new FileInputStream("rsain.txt");
                byte str[]=new byte[fip.available()];
                fip.read(str,0,str.length);
                String tte=new String(str);
                System.out.println("The mmm is :::"+tte);
                FileInputStream fis=new FileInputStream("path1.txt");
                ObjectInputStream ois=new ObjectInputStream(fis);
                Vector d=(Vector)ois.readObject();
                int siz=d.size();
                System.out.println("The size of the vector is ::"+siz);
                for(int j=0;j<siz;j++)
                {
                        System.out.print("++__++-->>"+d.get(j));
               }
                if(siz==2)
                {

                        Socket soc=new Socket((String)d.get(1),8888);
                        ObjectOutputStream oos=new ObjectOutputStream(soc.getOutputStream());
                        oos.writeObject("file");
                        System.out.println("The string in the textbox ::"+tte);
                        oos.writeObject(tte);
                }
                else
                {
                        d.removeElementAt(0);
                        int m=d.size();
                        System.out.println("The size after removing the element..."+m);
                        Socket soc=new Socket((String)d.get(1),8888);
                        ObjectOutputStream oos=new ObjectOutputStream(soc.getOutputStream());
                        oos.writeObject("file++");
                        System.out.println("The string in the textbox ::"+tte);
                        oos.writeObject(tte);
                        oos.writeObject(d);


                }


        }}
```

3.10 Output

PassWord.Java (Sender Side)

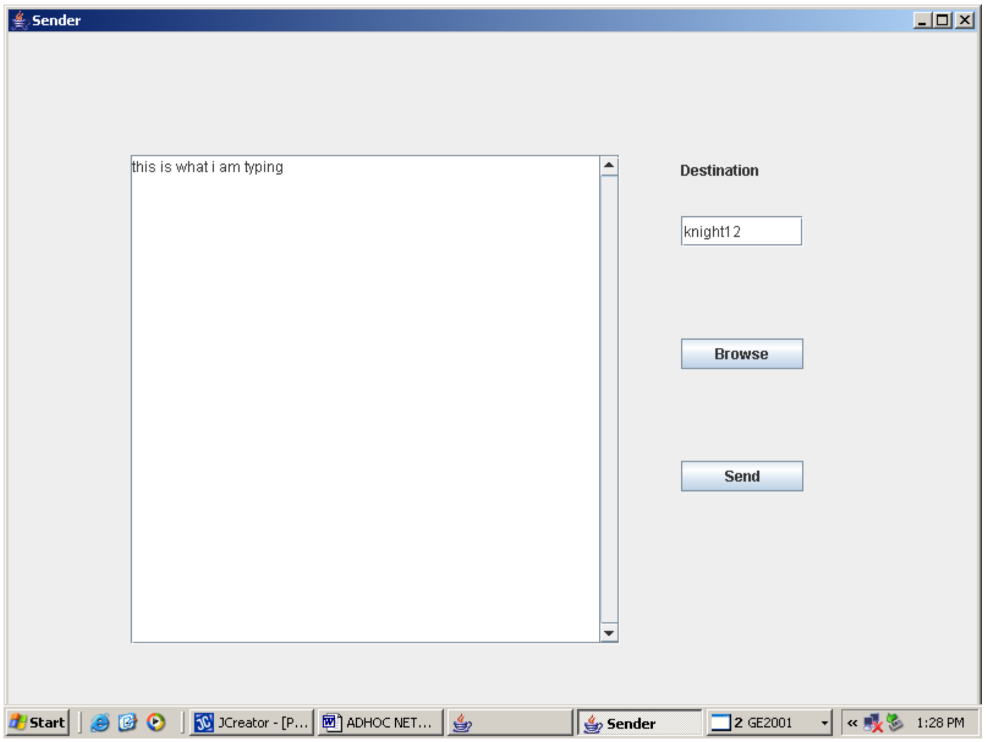


Admin

*****

Figure 3.6: Starting the Sender

Sender Data Entry Form for (Text / File)

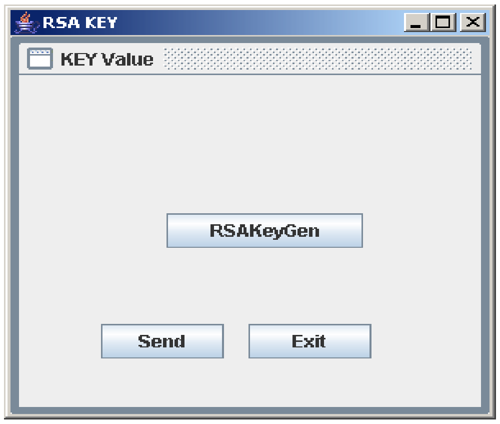


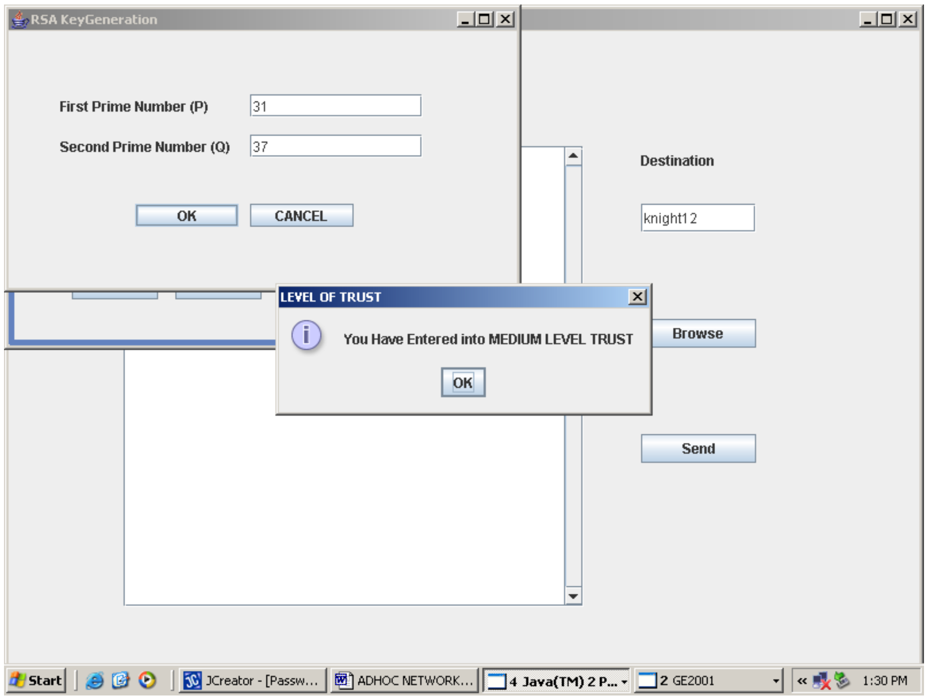
this is what i am typing

knight12

Figure 3.7: Entry Form

RSA KEY Generation Form





First Prime Number (P) 31

Second Prime Number (Q) 37

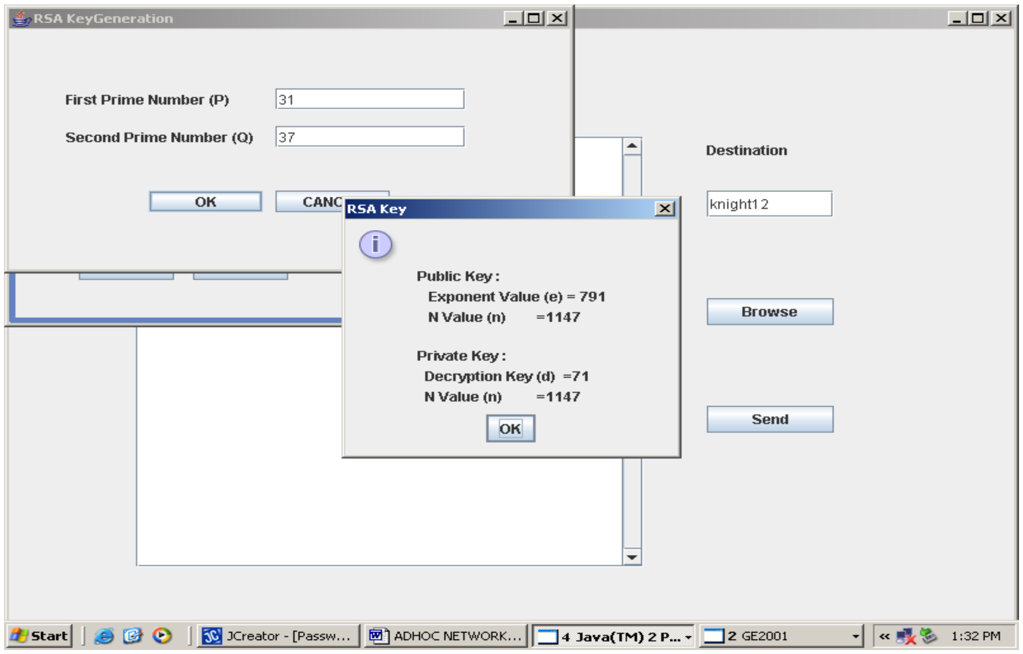
knight12

LEVEL OF TRUST

You Have Entered into MEDIUM LEVEL TRUST

Figure 3.8: Key Generation Form

KEY Generation Display


RSA KeyGeneration

First Prime Number (P) 31

Second Prime Number (Q) 37

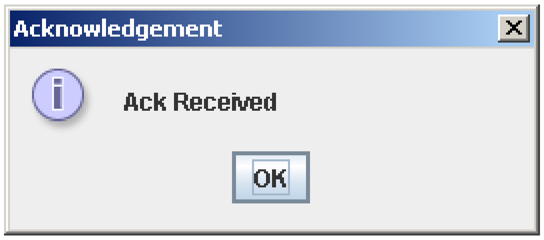
RSA Key

Public Key :
Exponent Value (e) = 791
N Value (n) =1147

Private Key :
Decryption Key (d) =71
N Value (n) =1147

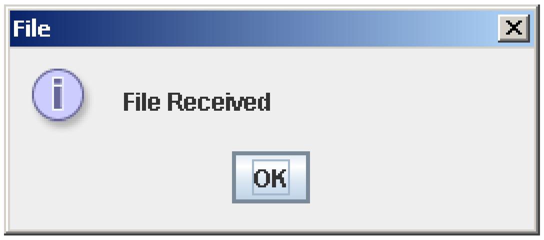
Destination knight12

As soon as the file received the destination it gives the information like this


Acknowledgement

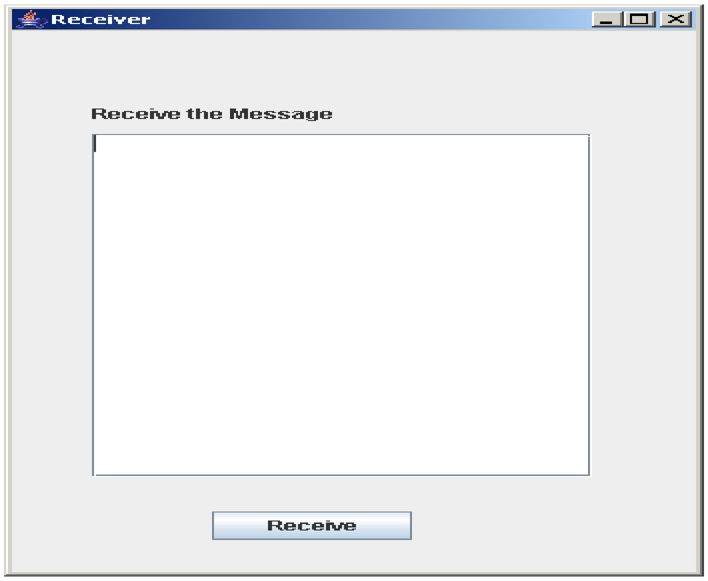
Ack Received

It is mentioning that it has received Socket with Some content

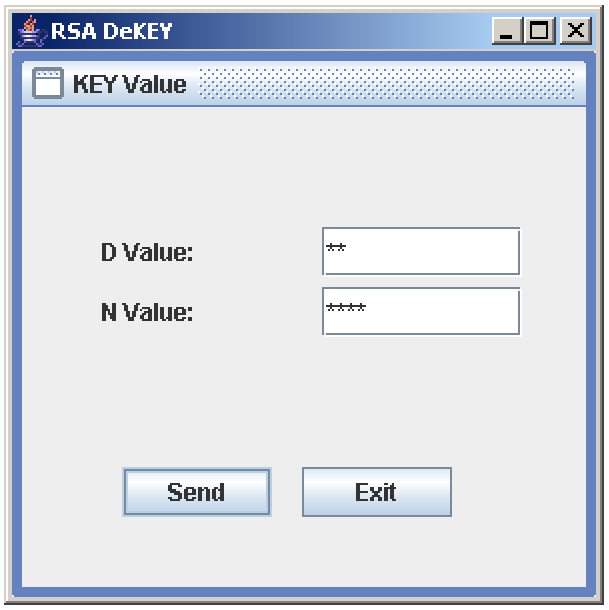

File

File Received

If we are pressing the receive button in below it will be shown the message after getting key.



The value of D and N Value has been entered by user

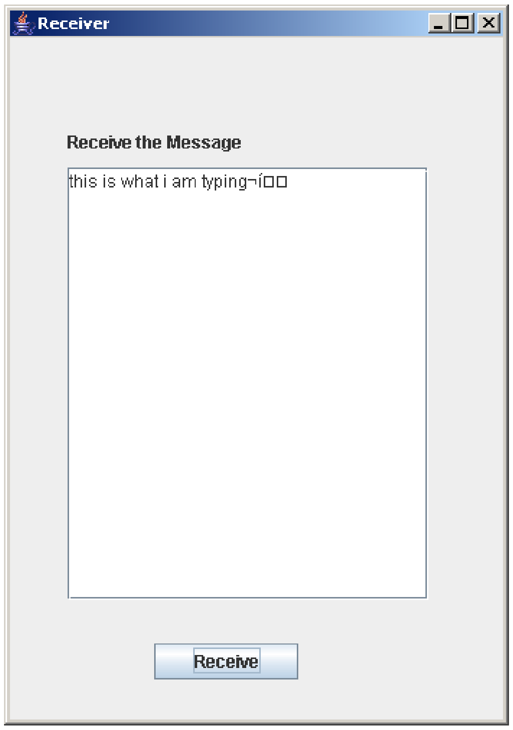


D Value: **

N Value: ****

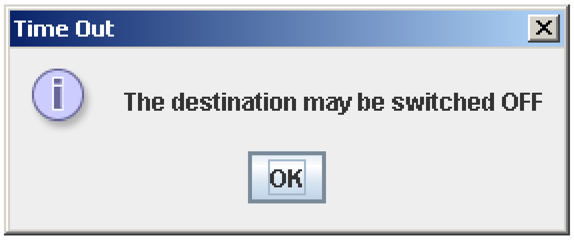
Finally Entered Information will be shown here

Figure 3.9: Key Generation Display



this is what i am typing¬í□□

If the Server has not been started then it indicates at sender



The destination may be switched OFF

Figure: 3.10 Responses

4. Conclusion:

The recent major trends of system modernization and transition to net enabled technologies present a growing need for more security in Data transfer. IT organizations are required to provide greater security for the transformation of messages. Hence we provide security using key Generation and Verification.

The proposed system has fully satisfied the following task:

- Low time consumption
- High Reliability
- Full control of source and target data definitions.
- Efficient utilization of effective utilizations
- High operational speed
- Less manual effort

The Project has been successfully completed as per the requirements.

References:-

1. I.F. Akyildiz, S.M. Joseph and Yi-Bing Lin, Movement-based location update and selective paging for PCS networks, IEEE/ACM Transactions on Networking 4 (1996) 94–104.

2. C. Alaettinoglu, K. Dussa-Zieger, I. Matta and A.U. Shankar, MaRS user's manual – version 1.0, Technical Report TR 91-80, The University of Maryland (June 1991).

3. S. Basagni, I. Chlamtac, V.R. Syrotiuk and B.A. Woodward, A distance routing effect algorithm for mobility (DREAM), in: *Proc. Of ACM/IEEE MOBICOM '98* (1998).

4. J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu and J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: *Proc. of MOBICOM '98* (1998). Wireless Sensor Networks: Applications and Challenges of Ubiquitous Sensing
   By- Daniele Puccinelli and Martin Haenggi

5. Outlier Detection Techniques for Wireless Sensor Networks: A Survey
   By- Yang Zhang, Nirvana Meratnia, and Paul Havinga

6. Coverage Problem in Wireless Sensor Networ By- GaoJun Fan and ShiYao

7. Location-Aided Routing (LAR) in mobile ad hoc networks , Young-Bae Ko and Nitin H. Vaidya

8. http://wiki.media-ulture.org.au/index.php/Main_Page

9. http://webnotations.com/sensor-networks/

10. http://www.sensor-networks.org

11. http://www.intechopen.com/books/wireless-sensor-networks-technology-and-protocols/asymmetric-encryption-in-wireless-sensor-networks#article-front

12. FatemehDeldar and Mohammad HossienYaghmaee, "Energy Efficient Prediction-based Clustering Algorithm for Target Tracking in Wireless Sensor Networks", International Conference on Intelligent Networking and Collaborative Systems,2010.

13. An Energy Efficient Link-Layer Security Protocol for Wireless Sensor Networks Leonard E. Lighfoot Jian Ren Tongtong Li Department of Electrical & Computer Engineering Michigan State University East Lansing, MI 48824 email: {lightf10, renjian, tongli}@egr.msu.edu

14. A. Perrig, R. Szewczyk, V, Wen, D. Culler, and J.D. Tygar, "SPINS: Security Protocols for Sensors Networks," Proceedings of Seventh Annual International Conference on Mobile Computing and Networks,2001.

15. J. Ren, T. Li and D. Aslam, “A Power Efficient Link Layer Protocol (LLSP) for Wireless Sensor Network,” Military Communication Conference, 2005, pp 1002-1007.

16. P. Levis, N. Lee, M. Welsh and D. Culler, “TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications,” Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, 2003, pp 126-137.

17. R. Venugopalan, P. Ganesan, P. Peddabachagari, A. Dean, F. Mueller and M. Sichitiui, “Encryption Overhead in Embedded Systems and Sensor Network Nodes: Modeling and Analysis,” 2003 International Conference on Compliers, Architectures and Synthesis for Embedded Systems, 2003, pp 188-197.

18. V. Scnayder, M. Hempstead, B. Chen, G. Allen and M. Welsh, “Simulating the Power Consumption of Large-Scale Sensor Network Applications,” Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, 2004, pp 188-200