A HIGHLY SCALABLE KEY PRE-DISTRIBUTION SCHEME IN WIRELESS SENSOR NETWORK

Name of Student - AAYUSH MITTAL

Enrollment No. -101319

Name of supervisor - 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

CONTENTS

TABLE OF FIGURES

CERTIFICATE

This is to certify that the work titled "**A HIGHLY SCALABLE KEY PRE-DISTRIBUTION SCHEME IN WIRELESS SENSOR NETWORK**" submitted by "**AAYUSH MITTAL**" in partial fulfillment for the award of degree of B.Tech of Computer Science Engineering from 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 Student	Signature of Supervisor
Name: Aayush Mittal	Name: Dr. Yashwant Singh
Date	Date

ACKNOWLEDGEMENT

On the very outset of this report, I would like to extend my sincere & heartfelt obligation towards all the personages who have helped me in this endeavor. Without their active guidance, help, cooperation & encouragement, I would not have made headway in the project.

I would like to show my greatest appreciation to **Dr. Yashwant Singh (Assistant Professor)**. I feel motivated every time I get his encouragement. For his coherent guidance throughout the tenure of the project, I feel fortunate to be taught by **Dr. Yashwant Singh**, who gave me his unwavering support. Besides being my mentor, he taught me that there is no substitute for hard work.

Signature of the student ……..

Name of Student Aayush Mittal

Date ……..

ABSTRACT

Given the sensitivity of the potential WSN applications and because of resource limitations, key management emerges as a challenging issue for WSNs. One of the main concerns when designing a key management scheme is the network scalability. Indeed, the protocol should support a large number of nodes to enable a large scale deployment of the network. In this paper, we propose a new scalable key management scheme for WSNs which provides a good secure connectivity coverage. For this purpose, we make use of the unital design theory. We show that the basic mapping from unitals to key pre-distribution allows us to achieve high network scalability. Nonetheless, this naive mapping does not guarantee a high key sharing probability. Therefore, we propose an enhanced unital-based key pre-distribution scheme providing high network scalability and good key sharing probability approximately lower bounded by $1 - e^{-1} \approx 0.632$. We conduct approximate analysis and simulations and compare our solution to those of existing methods for different criteria such as storage overhead, network scalability, network connectivity, average secure path length and network resiliency. Our results show that the proposed approach enhances the network scalability while providing high secure connectivity coverage and overall improved performance. Moreover, for an equal network size, our solution reduces significantly the storage overhead compared to those of existing solutions.

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 wireless sensor network (WSN) 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.

1.2 Genesis of Problem

Wireless sensor networks (WSNs) are increasingly used in critical applications within several fields including military, medical and industrial sectors. Given the sensitivity of these applications, sophisticated security services are required. Key management is a corner stone for many security services such as confidentiality and authentication which are required to secure communications in WSNs. The establishment of secure links between nodes is then a challenging problem in WSNs. Because of resource limitations, symmetric key establishment is

one of the most suitable paradigms for securing exchanges in WSNs. On the other hand, because of the lack of infrastructure in WSNs, we have usually no trusted third party which can attribute pair wise secret keys to neighboring nodes, that is why most existing solutions are based on key pre-distribution.

1.3 Problem Statement

A host of research work dealt with symmetric key pre-distribution issue for WSNs and many solutions have been proposed In the existing system many disadvantages occur: the design of key rings (blocks of keys) is strongly related to the network size, these solutions either suffer from low scalability (number of supported nodes), or degrade other performance metrics including secure connectivity, storage overhead and resiliency in the case of large networks.

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 my project is:

- To achieve a naive mapping from unital design to key pre-distribution and analyse the high scalability of the network.
- To enhance unital based key pre-distribution scheme that maintains a good key sharing probability while enhancing the network scalability.

1.5 Proposed System

In this proposed system, our aim is to tackle the scalability issue without degrading the other network performance metrics. For this purpose, we target the design of a scheme which ensures a good secure coverage of large scale networks with a low key storage overhead and a good network resiliency. To this end, we make use, of the unital design theory for efficient WSN key pre-distribution. We conduct approximate analysis and simulations and compare our solution to those of existing methods for different criteria such as storage overhead, network scalability, network connectivity, average secure path length and network resiliency. Our results show that the proposed approach enhances the network scalability while providing high secure connectivity coverage and overall improved performance.

1.6 Approach Used

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).

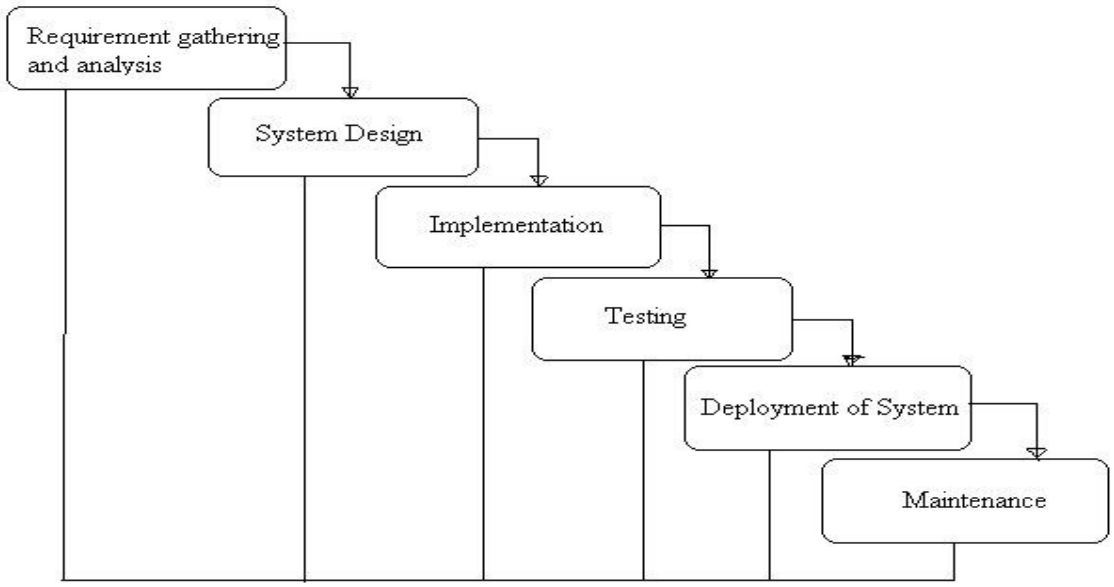

Figure 1.1: Software Development Life Cycle

1.7 Organisation of Thesis

In Chapter 2, we have discussed the literature survey in which we have seen the various features of WSN and its applications. We also discuss various key distribution scheme which are present and their drawbacks.

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 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.

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 Securing wireless sensor networks: a survey

The significant advances of hardware manufacturing technology and the development of efficient software algorithms make technically and economically feasible a network composed of numerous, small, low-cost sensors using wireless communications, that is, a wireless sensor network. WSNs have attracted intensive interest from both academia and industry due to their wide application in civil and military scenarios. In hostile scenarios, it is very important to protect WSNs from malicious attacks. Due to various resource limitations and the salient features of a wireless sensor network, the security design for such networks is significantly challenging. In this article, a comprehensive survey of WSN security issues was presented that were investigated by researchers in recent years and that shed light on future directions for WSN security.

2.5 A key-management scheme for distributed sensor networks

Distributed Sensor Networks (DSNs) are ad-hoc mobile networks that include sensor nodes with limited computation and communication capabilities. DSNs are dynamic in the sense that they allow addition and deletion of sensor nodes after deployment to grow the network or replace failing and unreliable nodes. DSNs may be deployed in hostile areas where communication is monitored and nodes are subject to capture and surreptitious use by an adversary. Hence DSNs require cryptographic protection of communications, sensor capture detection, key revocation and sensor disabling. In this paper, a key-management scheme designed to satisfy both operational and security requirements of DSNs were presented.

2.6 A key management scheme for wireless sensor networks using deployment knowledge

To achieve security in wireless sensor networks, it is important to he able to encrypt messages sent among sensor nodes. Keys for encryption purposes must he agreed upon by communicating nodes. Due to resource constraints, achieving such key agreement in wireless sensor networks is nontrivial. Many key agreement schemes used in general networks, such as Diffie-Hellman and public-key based schemes, are not suitable for wireless sensor networks. Pre-distribution of secret keys for all pairs of nodes is not viable due to the large amount of memory used when the network size is large. Recently, a random key pre-distribution scheme and its improvements have been proposed. A common assumption made by these random key pre-distribution schemes is that no deployment knowledge is available. Noticing that in many practical scenarios, certain deployment knowledge may be available a priori, we propose a novel random key pre-distribution scheme that exploits deployment knowledge and avoids unnecessary key assignments. It was shown that the performance (including connectivity, memory usage, and network resilience against node capture) of sensor networks can he substantially improved with the use of our proposed scheme. The scheme and its detailed performance evaluation are presented in this paper.

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
- Social 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.

3.1.3 Social feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

3.2 Modules

1) Node Deployment
2) Key Generation
3) Key Pre-distribution Technique
4) Secure Transmission with Energy

Node Deployment

The first module is Node deployment, where the node can be deployed by specifying the number of nodes in the network. After specifying the number of nodes in the network, the nodes are deployed. The nodes are deployed with unique ID (Identity) number so that each can be differentiated. And also nodes are deployed with their energy levels.

Key Generation

After the Node deployment module, the key generation module is developed, where the number of nodes and number of blocks should be specified, so that the key will be generated. The key is symmetric key and the key is displayed in the text area given in the node.

Key Pre-distribution Technique:

In this module, we generate blocks of m order initial design, where each block corresponds to a key set. We pre-load then each node with t completely disjoint blocks where t is a protocol parameter that we will discuss later in this section. In lemma 1, we demonstrate the condition of existence of such t completely disjoint blocks among the unital blocks. In the basic approach each node is pre-loaded with only one unital block and we proved that each two nodes share at most one key. Contrary to this, pre-loading each two nodes with t disjoint unital blocks means that each two nodes share between zero and keys since each two unitals blocks share at most one element. After the deployment step, each two neighbors exchange the identifiers of their keys in order to determine the common keys. This approach enhances the network resiliency since the attackers have to compromise more overlap keys to break a secure link. Otherwise, when neighbors do not share any key, they should find a secure path composed of successive secure link

Secure Transmission with Energy

In this module, the node distance is configured and then the nodes with their neighbor information are displayed. So the nodes which is near by the node, is selected and the energy level is first calculated to verify the secure transmission. After that the data is uploaded and sent to the destination node. Where in the destination node, the key is verified and then the data is received

3.3 Block Diagram

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

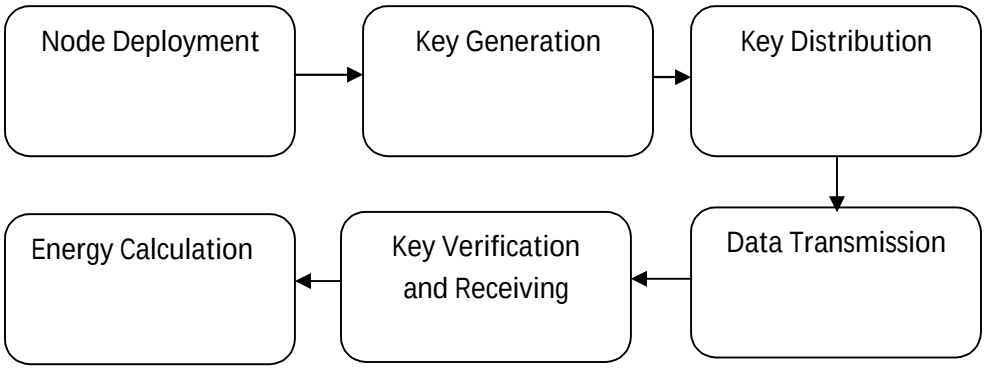

Figure 3.1: Block Diagram

3.4 UML Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

3.4.1 Activity Diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

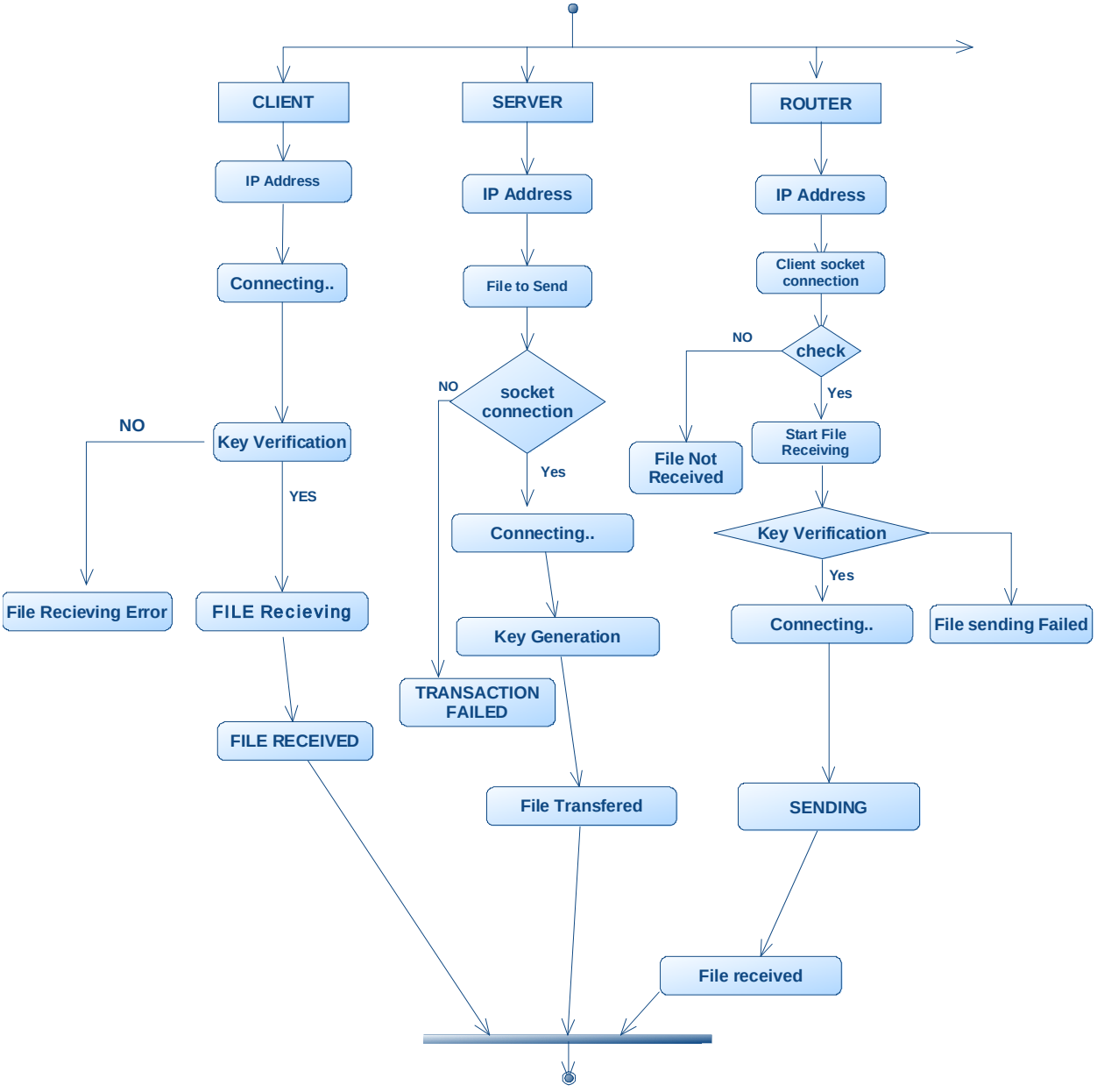

Figure 3.2: Activity Diagram

3.4.2 Sequence Diagram:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

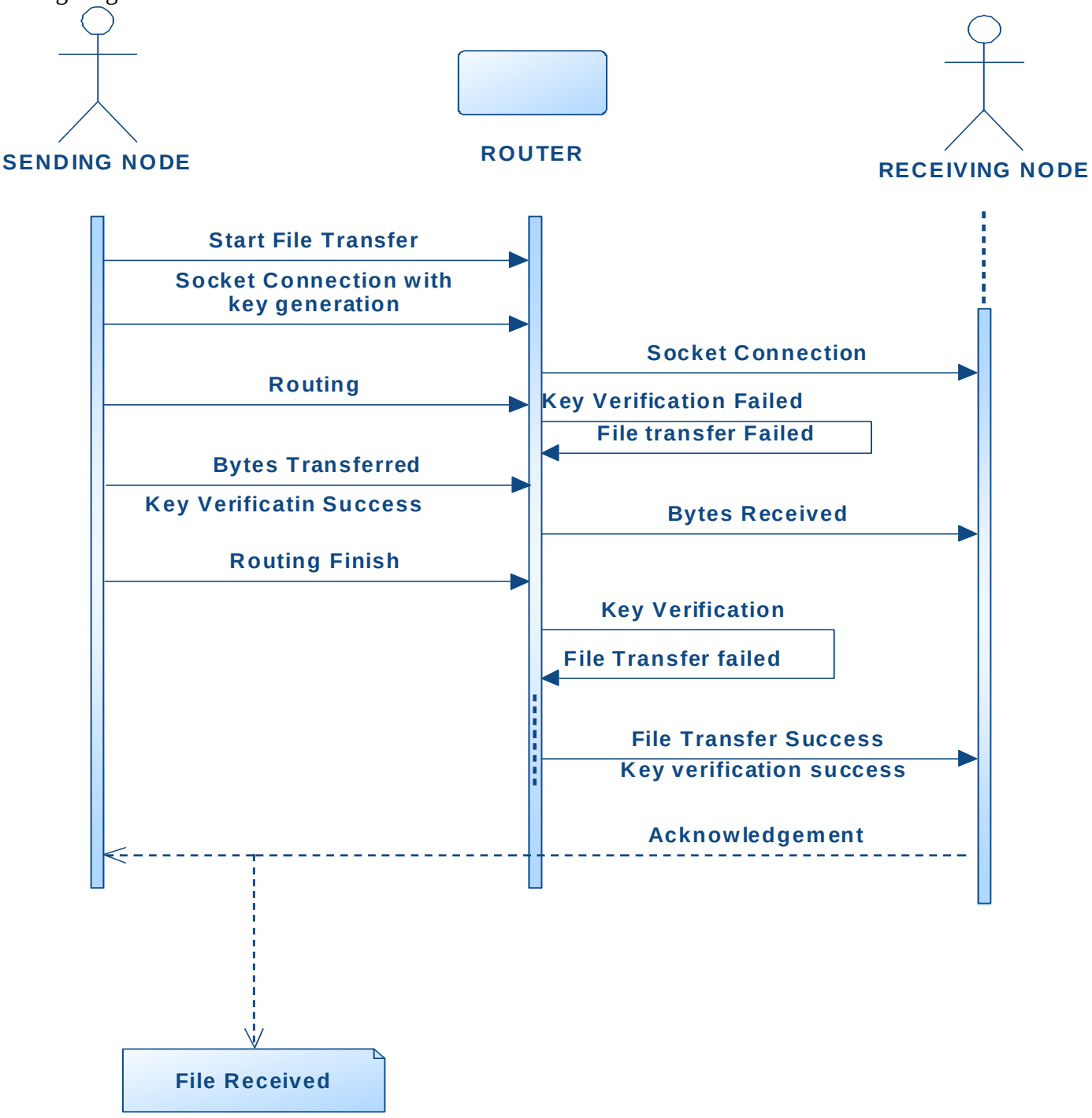

Figure 3.3: Sequence Diagram

3.4.3 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

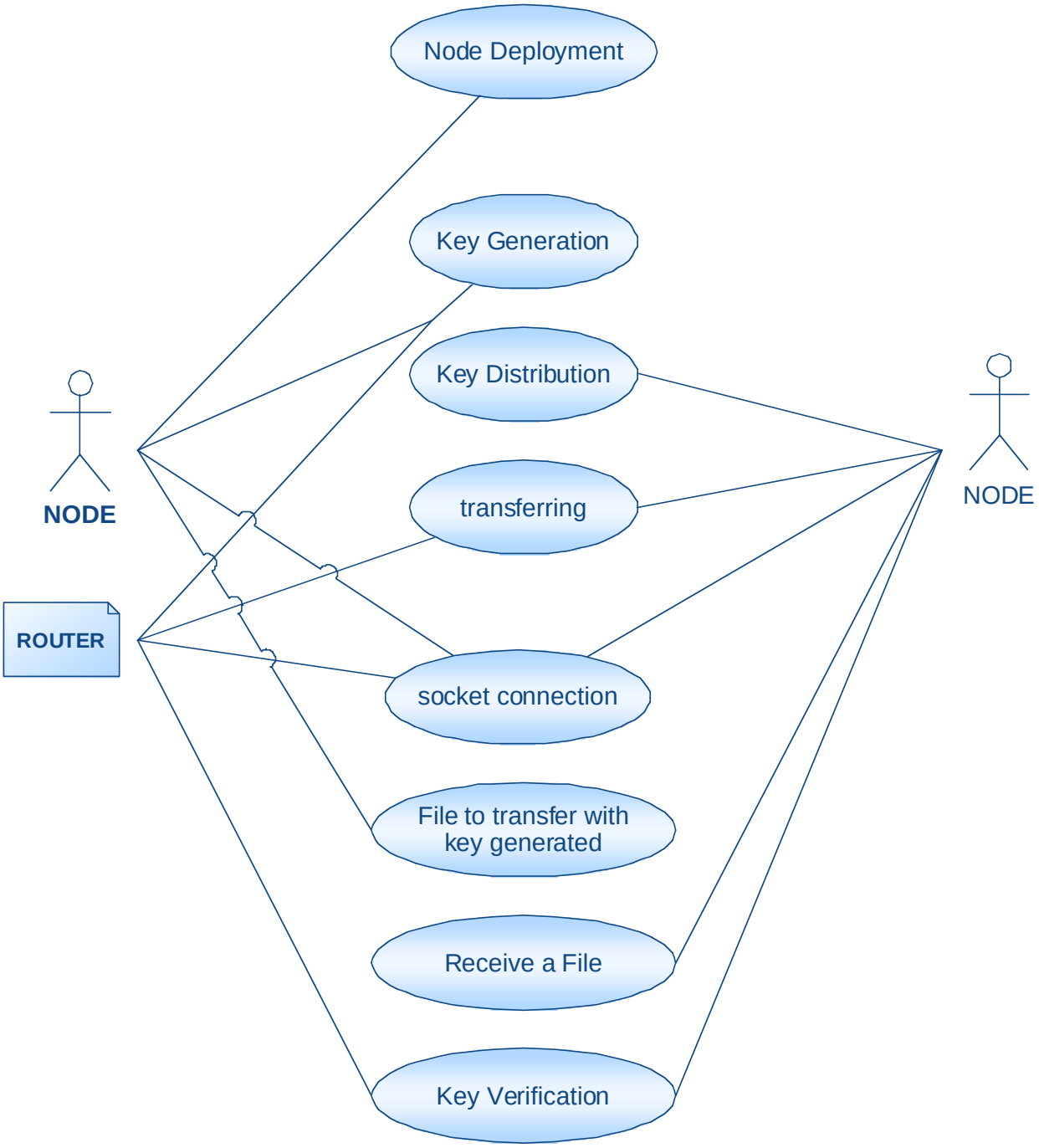

Figure3.4: Use Case diagram

3.4.4 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

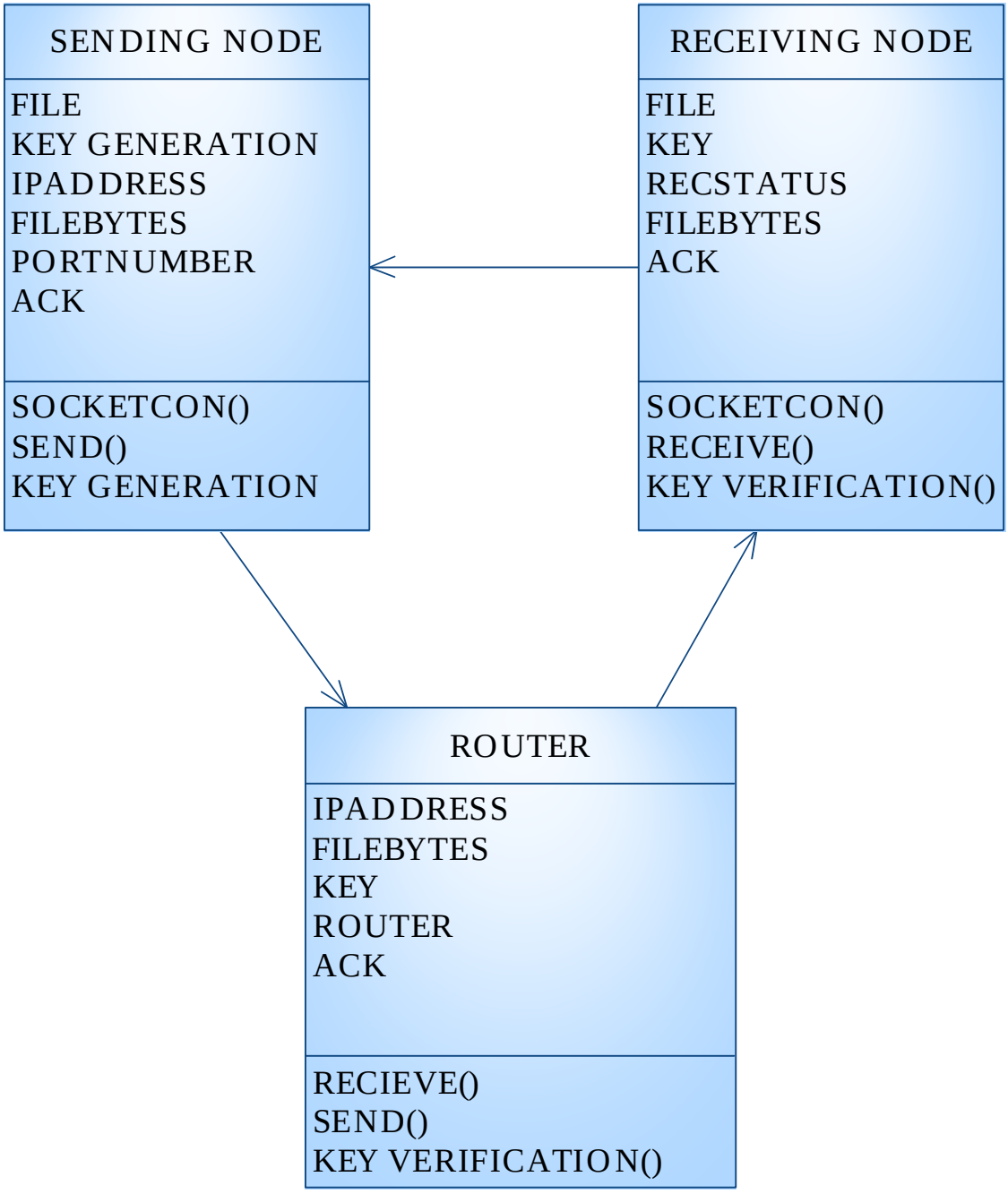

Figure 3.5 Class Diagram

3.5 Data Flow Diagram:

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

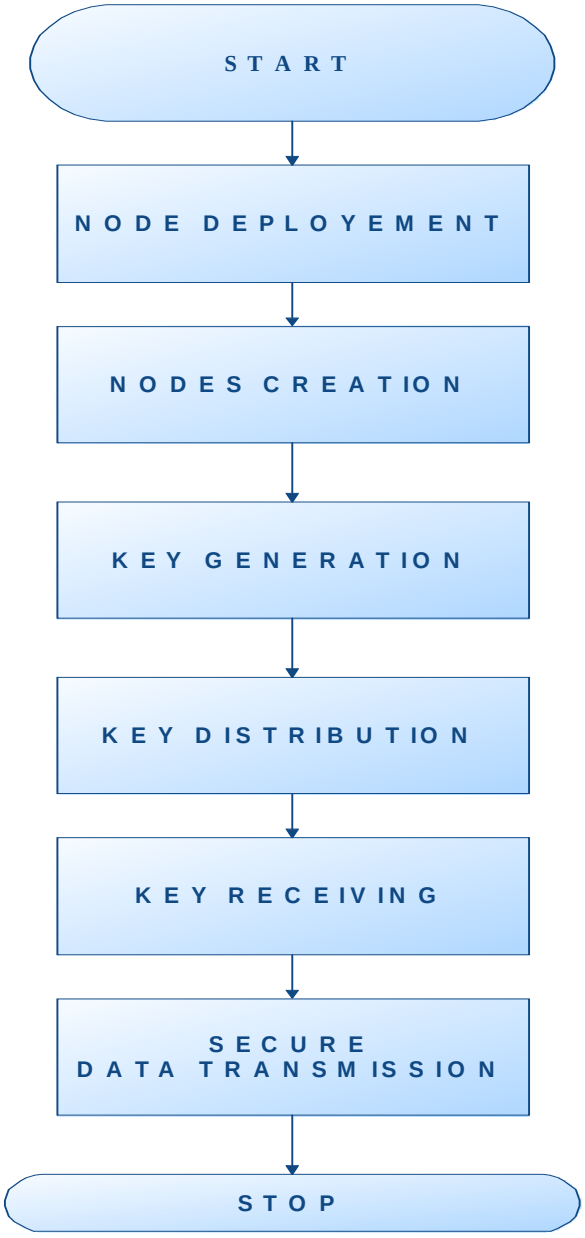

Figure 3.6: Data Flow Diagram

3.6 Source code

```
package com.mycompany.design;
import com.multicast.MulticastRx;
import com.multicast.TrippleDes;
import java.awt.Color;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.security.NoSuchAlgorithmException;
import java.util.Iterator;
import java.util.Random;
import java.util.Set;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.swing.JFileChoose;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;



public class Node_Design extends javax.swing.JFrame
 {

    private static final long serialVersionUID = 2L;

 public Action action;

 public String source;

    int port;

   public int enc;

  public int energy;
```

```
    public String dis;

   public MulticastRx mrx;

 Receive receive;

String mpcrPath;

 public String path;

 static Random rr = new Random();

        int ii = rr.nextInt(108945000);

         String hhh = ii+"";

      String ekey ;


   public Node_Design()
throws UnknownHostException
 {

       initComponents();


        jTextArea2.setEditable(false);

      jTextArea3.setEditable(false);

      jTextArea5.setEditable(false);

    jTextField3.setEditable(false);

     jTextField6.setEditable(false);

    jButton2.setVisible(false);


        init();


  }


    public Node_Design(String  key)
```

```
throws UnknownHostException
 {
        initComponents();
  this.getContentPane().setBackground(Color.PINK);
    ekey = key;
  jButton2.setVisible(false);
       jTextArea2.setEditable(false);
      jTextArea3.setEditable(false);
 jTextArea5.setEditable(false);
     jTextField3.setEditable(false);
   jTextField6.setEditable(false);
   jButton5.setVisible(false);
        jPanel1.setVisible(false);
      jPanel3.setVisible(false);
      jPanel2.setVisible(false);

        init();

    }


    private void init()
{
      action = new Action();
       source = action.getSource();
   setTitle(source);
    energy = action.getEnergy();
```

```
    jTextField3.setText("" + energy);
     jLabel5.setText(source);
    port = action.getPort();
   receive = new Receive(this, port, action);

    }
private void initComponents()
{
  jLabel1 = new javax.swing.JLabel();
     jLabel2 = new javax.swing.JLabel();
      jLabel5 = new javax.swing.JLabel();
      jTextField3 = new javax.swing.JTextField();
     jLabel9 = new javax.swing.JLabel();
      jLabel10 = new javax.swing.JLabel();
   jLabel11 = new javax.swing.JLabel();
    jLabel12 = new javax.swing.JLabel();
    jTextField5 = new javax.swing.JTextField();
      jButton2 = new javax.swing.JButton();
       jTextField6 = new javax.swing.JTextField();
      jLabel13 = new javax.swing.JLabel();
        jTextField7 = new javax.swing.JTextField();
  jTextField8 = new javax.swing.JTextField();
       jButton4 = new javax.swing.JButton();
       jButton1 = new javax.swing.JButton();
```

```
    jButton5 = new javax.swing.JButton();
    jPanel3 = new javax.swing.JPanel();
  send = new javax.swing.JButton();
        get_energy = new javax.swing.JButton();
    jLabel6 = new javax.swing.JLabel();
       jScrollPane5 = new javax.swing.JScrollPane();
    jTextArea5 = new javax.swing.JTextArea();
        jTextField2 = new javax.swing.JTextField();
      get_Neighbor = new javax.swing.JButton();
     file_browse = new javax.swing.JButton();
      cancel = new javax.swing.JButton();
     jScrollPane1 = new javax.swing.JScrollPane();
       jTextArea1 = new javax.swing.JTextArea();
    jPanel1 = new javax.swing.JPanel();
     jPanel2 = new javax.swing.JPanel();
     jScrollPane3 = new javax.swing.JScrollPane();
     jTextArea3 = new javax.swing.JTextArea();
      jScrollPane2 = new javax.swing.JScrollPane();
        jTextArea2 = new javax.swing.JTextArea();
      jLabel3 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
      distance_config = new javax.swing.JButton();
     jTextField1 = new javax.swing.JTextField();
```

```
    jLabel7 = new javax.swing.JLabel();
     jLabel4 = new javax.swing.JLabel();

     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
   setTitle("Node");
  setMinimumSize(new java.awt.Dimension(930, 700));
    getContentPane().setLayout(null);

      jLabel1.setFont(new java.awt.Font("Times New Roman", 1, 24));

 jLabel1.setForeground(new java.awt.Color(0, 0, 153));
 jLabel1.setText("A HIGHLY SCALABLE KEY PRE-DISTRIBUTION SCHEME FOR WSN");
    getContentPane().add(jLabel1);
    jLabel1.setBounds(80, 10, 800, 40);

    jLabel2.setFont(new java.awt.Font("Times New Roman", 1, 14));

  jLabel2.setText("Energy");
  getContentPane().add(jLabel2);
     jLabel2.setBounds(570, 57, 60, 20);

    jLabel5.setFont(new java.awt.Font("Times New Roman", 1, 14));

   getContentPane().add(jLabel5);
      jLabel5.setBounds(660, 90, 90, 0);
```

```
      jTextField3.addActionListener(new java.awt.event.ActionListener()
{
       public void actionPerformed(java.awt.event.ActionEvent evt)
{
           jTextField3ActionPerformed(evt);
         }
      });
   getContentPane().add(jTextField3);
  jTextField3.setBounds(640, 50, 120, 30);

   jLabel9.setFont(new java.awt.Font("Tahoma", 3, 14));

  jLabel9.setText("NODE DEPLOYEMENT");
   getContentPane().add(jLabel9);
  jLabel9.setBounds(50, 80, 180, 30);

     jLabel10.setFont(new java.awt.Font("Tahoma", 3, 14));

    jLabel10.setText("Node");
    getContentPane().add(jLabel10);
      jLabel10.setBounds(20, 140, 80, 30);

jLabel11.setFont(new java.awt.Font("Tahoma", 3, 14));

     jLabel11.setText("BLOCK GENERATION");
  getContentPane().add(jLabel11);
  jLabel11.setBounds(470, 100, 180, 30);
```

```
  jLabel12.setFont(new java.awt.Font("Tahoma", 3, 12));

  jLabel12.setText("Node Size");
     getContentPane().add(jLabel12);
    jLabel12.setBounds(370, 150, 90, 30);
   getContentPane().add(jTextField5);
    jTextField5.setBounds(480, 150, 190, 30);

     jButton2.setFont(new java.awt.Font("Tahoma", 1, 12));

jButton2.setText("Distribute Key");
     jButton2.addActionListener(new java.awt.event.ActionListener()
{
       public void actionPerformed(java.awt.event.ActionEvent evt)
{
           jButton2ActionPerformed(evt);
    }
      }
);
      getContentPane().add(jButton2);
   jButton2.setBounds(810, 150, 140, 30);
      getContentPane().add(jTextField6);
    jTextField6.setBounds(690, 260, 240, 40);

    jLabel13.setFont(new java.awt.Font("Tahoma", 1, 14));

    jLabel13.setText(" Block size");
```

```
    getContentPane().add(jLabel13);

   jLabel13.setBounds(370, 200, 80, 30);

        getContentPane().add(jTextField7);

    jTextField7.setBounds(480, 200, 190, 30);


      jTextField8.addKeyListener(new java.awt.event.KeyAdapter()
{

       public void keyTyped(java.awt.event.KeyEvent evt)
{

            jTextField8KeyTyped(evt);

          }

    }
);

   getContentPane().add(jTextField8);

     jTextField8.setBounds(90, 140, 180, 30);


     jButton4.setFont(new java.awt.Font("Tahoma", 1, 12));


  jButton4.setText("Generate Key");

   jButton4.addActionListener(new java.awt.event.ActionListener()
{

 public void actionPerformed(java.awt.event.ActionEvent evt)
{

      jButton4ActionPerformed(evt);

        }

     });

   getContentPane().add(jButton4);
```

```
 jButton4.setBounds(680, 150, 120, 30);

       jButton1.setText("Received key");

      jButton1.addActionListener(new java.awt.event.ActionListener()
{

        public void actionPerformed(java.awt.event.ActionEvent evt)
{

         jButton1ActionPerformed(evt);

       }

       });

    getContentPane().add(jButton1);

   jButton1.setBounds(550, 270, 110, 30);

      jButton5.setText("DEPLOY");

    jButton5.addActionListener(new java.awt.event.ActionListener()
 {

   public void actionPerformed(java.awt.event.ActionEvent evt)
{

  jButton5ActionPerformed(evt);

         }

     });

   getContentPane().add(jButton5);

 jButton5.setBounds(120, 190, 100, 30);

      jPanel3.setBackground(new java.awt.Color(255, 204, 204));

       jPanel3.setLayout(null);
```

```
      send.setFont(new java.awt.Font("Times New Roman", 1, 14));


 send.setText("Send To");

      send.addActionListener(new java.awt.event.ActionListener()
{

   public void actionPerformed(java.awt.event.ActionEvent evt)
 {

     sendActionPerformed(evt);

        }

    });

   jPanel3.add(send);

     send.setBounds(130, 410, 100, 25);


       get_energy.setFont(new java.awt.Font("Times New Roman", 1, 14));


  get_energy.setText("get Energy");
       get_energy.addActionListener(new java.awt.event.ActionListener()
{

  public void actionPerformed(java.awt.event.ActionEvent evt)
 {
              get_energyActionPerformed(evt);

          }

      });

  jPanel3.add(get_energy);

   get_energy.setBounds(20, 410, 100, 25);

       jLabel6.setFont(new java.awt.Font("Times New Roman", 1, 14));


   jLabel6.setText(".");
```

```
    jPanel3.add(jLabel6);
      jLabel6.setBounds(240, 420, 180, 30);


    jTextArea5.setColumns(20);

     jTextArea5.setRows(5);

      jScrollPane5.setViewportView(jTextArea5);


     jPanel3.add(jScrollPane5);

  jScrollPane5.setBounds(10, 300, 370, 96);

      jPanel3.add(jTextField2);
       jTextField2.setBounds(0, 230, 200, 30);


       get_Neighbor.setFont(new java.awt.Font("Times New Roman", 1, 14));

       get_Neighbor.setText("get Neighbor Route");

      get_Neighbor.addActionListener(new java.awt.event.ActionListener()
{

        public void actionPerformed(java.awt.event.ActionEvent evt)
{

       get_NeighborActionPerformed(evt);

       }

   });

 jPanel3.add(get_Neighbor);
       get_Neighbor.setBounds(220, 240, 160, 30);


    file_browse.setFont(new java.awt.Font("Times New Roman", 1, 14));

       file_browse.setText("Browse File");

    file_browse.addActionListener(new java.awt.event.ActionListener()
{
```

```
 public void actionPerformed(java.awt.event.ActionEvent evt)
{

      file_browseActionPerformed(evt);

        }

     });

    jPanel3.add(file_browse);
       file_browse.setBounds(50, 170, 120, 25);


     cancel.setFont(new java.awt.Font("Times New Roman", 1, 14));


   cancel.setText("Cancel");

  cancel.addActionListener(new java.awt.event.ActionListener()
{

      public void actionPerformed(java.awt.event.ActionEvent evt)
{

    cancelActionPerformed(evt);

         }

       });

      jPanel3.add(cancel);

     cancel.setBounds(210, 170, 100, 25);

       jTextArea1.setColumns(20);

      jTextArea1.setRows(5);

    jScrollPane1.setViewportView(jTextArea1);


      jPanel3.add(jScrollPane1);
       jScrollPane1.setBounds(20, 40, 380, 120);
```

```
        getContentPane().add(jPanel3);

     jPanel3.setBounds(0, 250, 410, 470);

        getContentPane().add(jPanel1);

     jPanel1.setBounds(440, 270, 10, 10);


  jPanel2.setBackground(new java.awt.Color(255, 204, 204));
        jPanel2.setLayout(null);


   jTextArea3.setColumns(20);

    jTextArea3.setRows(5);

    jScrollPane3.setViewportView(jTextArea3);

        jPanel2.add(jScrollPane3);

     jScrollPane3.setBounds(140, 220, 300, 100);


      jTextArea2.setColumns(20);

       jTextArea2.setRows(5);
        jScrollPane2.setViewportView(jTextArea2);


        jPanel2.add(jScrollPane2);

    jScrollPane2.setBounds(140, 120, 300, 70);


        jLabel3.setFont(new java.awt.Font("Times New Roman", 1, 14));


 jLabel3.setText("Received Data");

 jPanel2.add(jLabel3);
        jLabel3.setBounds(10, 240, 110, 17);


    jLabel8.setFont(new java.awt.Font("Times New Roman", 1, 14));
```

```
    jLabel8.setText("Available Nodes");

   jPanel2.add(jLabel8);

  jLabel8.setBounds(20, 130, 120, 17);


 distance_config.setFont(new java.awt.Font("Times New Roman", 1, 14));


distance_config.setText("Distance Configure");

   distance_config.addActionListener(new java.awt.event.ActionListener()
{

         public void actionPerformed(java.awt.event.ActionEvent evt)
{

distance_configActionPerformed(evt);

        }

    });

    jPanel2.add(distance_config);

       distance_config.setBounds(270, 70, 170, 25);


      jTextField1.setFont(new java.awt.Font("Times New Roman", 1, 14));

        jPanel2.add(jTextField1);

     jTextField1.setBounds(150, 70, 100, 30);


       jLabel7.setFont(new java.awt.Font("Times New Roman", 1, 14));


  jLabel7.setText("Enter the Distance");

     jPanel2.add(jLabel7);

    jLabel7.setBounds(20, 70, 120, 20);
```

```
      jLabel4.setFont(new java.awt.Font("Times New Roman", 1, 14));

jLabel4.setText("Source Name :");

      jPanel2.add(jLabel4);

  jLabel4.setBounds(110, 20, 100, 17);


       getContentPane().add(jPanel2);

    jPanel2.setBounds(440, 320, 490, 390);


      setSize(new java.awt.Dimension(987, 790));

    setLocationRelativeTo(null);

  }


  private void distance_configActionPerformed(java.awt.event.ActionEvent evt)
{

    dis = jTextField1.getText();

    try {
          Integer.parseInt(dis);

      }
 catch (NumberFormatException f)
{

  JOptionPane.showMessageDialog(null, f + "\n" + "Type only Numbers.....", "Error", 2);


       }

   if (dis.equals(""))
{

       JOptionPane.showMessageDialog(null, "Enter the Distance.");

     }
```

```
 else
{
       mrx = new MulticastRx(this);
        new com.multicast.MulticastTx(source, dis, port);
        distance_config.setEnabled(false);
    }

   }
    private void file_browseActionPerformed(java.awt.event.ActionEvent evt)
{
     StringBuffer buffer;
       String strLine;
     File file = null;
     JFileChooser Chooser = new JFileChooser();
   Chooser.setMultiSelectionEnabled(true);
       Chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
    int result = Chooser.showDialog(this, "Open");
     if (result == JFileChooser.APPROVE_OPTION)
{
file = Chooser.getSelectedFile();

        }
       try
{
    FileInputStream fstream = new FileInputStream(file);
    DataInputStream ins = new DataInputStream(fstream);
      BufferedReader br = new BufferedReader(new InputStreamReader(ins));
```

```
          buffer = new StringBuffer();
            while ((strLine = br.readLine()) != null)
{

buffer.append(strLine + "\n");

        }

       jTextArea1.setText(buffer.toString() + "\n");
        }
catch (Exception e)
{

    System.out.println(e);

        }

}
  private void get_NeighborActionPerformed(java.awt.event.ActionEvent evt)
 {
if
(jTextArea2.getText().equals(""))
 {

     JOptionPane.showMessageDialog(null, "Neighbour Nodes are not there.");

   }
else if (jTextField2.getText().equals(""))
 {

   JOptionPane.showMessageDialog(null, "Enter the Destination");

 }
 else
 {

       Vector<String> path = new Vector<String>();

       path.add(source);

        action.routing(mrx, path, jTextField2.getText());

        }

   }
  private void cancelActionPerformed(java.awt.event.ActionEvent evt)
```

```
{
    jTextArea1.setText(null);

   }

  private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt)
 {
   }
  private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{


       String kk = null;
         jTextField6.setText(ekey);


   }
  private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
 {

      }


   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt, SecretKey key)
{
   }
   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
 {
      try {

          String dec = new TrippleDes().decrypt(ekey);

        jTextField6.setText(ekey);

        jPanel3.setVisible(true);

      jPanel2.setVisible(true);
       }
 catch (Exception ex)
 {

    Logger.getLogger(Node_Design.class.getName()).log(Level.SEVERE, null, ex);

       }
```

```
    }
   private void jButton5ActionPerformed(java.awt.event.ActionEvent evt)
{
     try
 {
           String gg = jTextField8.getText();
         ekey = new TrippleDes().encrypt(gg);


       int a = Integer.parseInt(gg);
           for (int i = 1; i < a; i++) {

try
 {

         new Node_Design(ekey).setVisible(true);
        }
catch (UnknownHostException ex)
 {

         Logger.getLogger(Node_Design.class.getName()).log(Level.SEVERE, null, ex);

         }
              JOptionPane.showMessageDialog(null, "Node Deployed.", "", 2);

       }

    }
catch (Exception ex)
{

         Logger.getLogger(Node_Design.class.getName()).log(Level.SEVERE, null, ex);

      }


    }
  private void jTextField8KeyTyped(java.awt.event.KeyEvent evt)
{
     jButton5.setVisible(true);

    }
private void get_energyActionPerformed(java.awt.event.ActionEvent evt)
{

        mpcrPath = receive.allPaths.get(receive.allPaths.firstKey());
```

```
    jLabel6.setText(mpcrPath + " [" + receive.allPaths.firstKey()
          + "]");

}

 private void sendActionPerformed(java.awt.event.ActionEvent evt)
 {
    String text = jTextArea1.getText();

  if (text.equals(""))
 {

   JOptionPane.showMessageDialog(null, "Enter the Message.");

     }
else
 {

       Vector<String> path = action.getPath(mpcrPath);

      action.sendData(mrx, path, text);
         JOptionPane.showMessageDialog(null, "Data Sending.....");

        JOptionPane.showMessageDialog(null, "Data Sent Successfully.....");

      }
   }
  public static void main(String args[])
{

     for
(javax.swing.UIManager.LookAndFeelInfojavax.swing.UIManager.getInstalledLookAndFeels())
 {

      if ("Nimbus".equals(info.getName()))
 {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());

           break;

         }

         }

     }
```

```
 catch (ClassNotFoundException ex)
{

java.util.logging.Logger.getLogger(Node_Design.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);

       }
 catch (InstantiationException ex)
 {


java.util.logging.Logger.getLogger(Node_Design.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);

 }
catch (IllegalAccessException ex)
 {


java.util.logging.Logger.getLogger(Node_Design.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);

  }
catch (javax.swing.UnsupportedLookAndFeelException ex)
{


java.util.logging.Logger.getLogger(Node_Design.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
       }


java.awt.EventQueue.invokeLater(new Runnable() {

public void run()
{

          try {
                    KeyGenerator keyGen = KeyGenerator.getInstance("DES");

            SecretKey key = keyGen.generateKey();
                 new Node_Design("").setVisible(true);

             }
catch (UnknownHostException ex)
{
```

```
            Logger.getLogger(Node_Design.class.getName()).log(Level.SEVERE, null, ex);
          }
        }
      });
  }

    private javax.swing.JButton cancel;
private javax.swing.JButton distance_config;
  private javax.swing.JButton file_browse;
    private javax.swing.JButton get_Neighbor;
 private javax.swing.JButton get_energy;
private javax.swing.JButton jButton1;
  private javax.swing.JButton jButton2;
   private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
  private javax.swing.JLabel jLabel11;
   private javax.swing.JLabel jLabel12;
 private javax.swing.JLabel jLabel13;
  private javax.swing.JLabel jLabel2;
 private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
```

```
  private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
     private javax.swing.JPanel jPanel3;
 private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
   private javax.swing.JScrollPane jScrollPane3;
 private javax.swing.JScrollPane jScrollPane5;
   private javax.swing.JTextArea jTextArea1;
 public javax.swing.JTextArea jTextArea2;
 public javax.swing.JTextArea jTextArea3;
    public javax.swing.JTextArea jTextArea5;
 private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
 private javax.swing.JTextField jTextField3;
 private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
private javax.swing.JTextField jTextField7;
   private javax.swing.JTextField jTextField8;
 private javax.swing.JButton send;
```

3.7 Output:

Figure 3.7: Screen shot 1

Figure 3.8: Screen shot 2

Figure 3.9: Screen shot 3

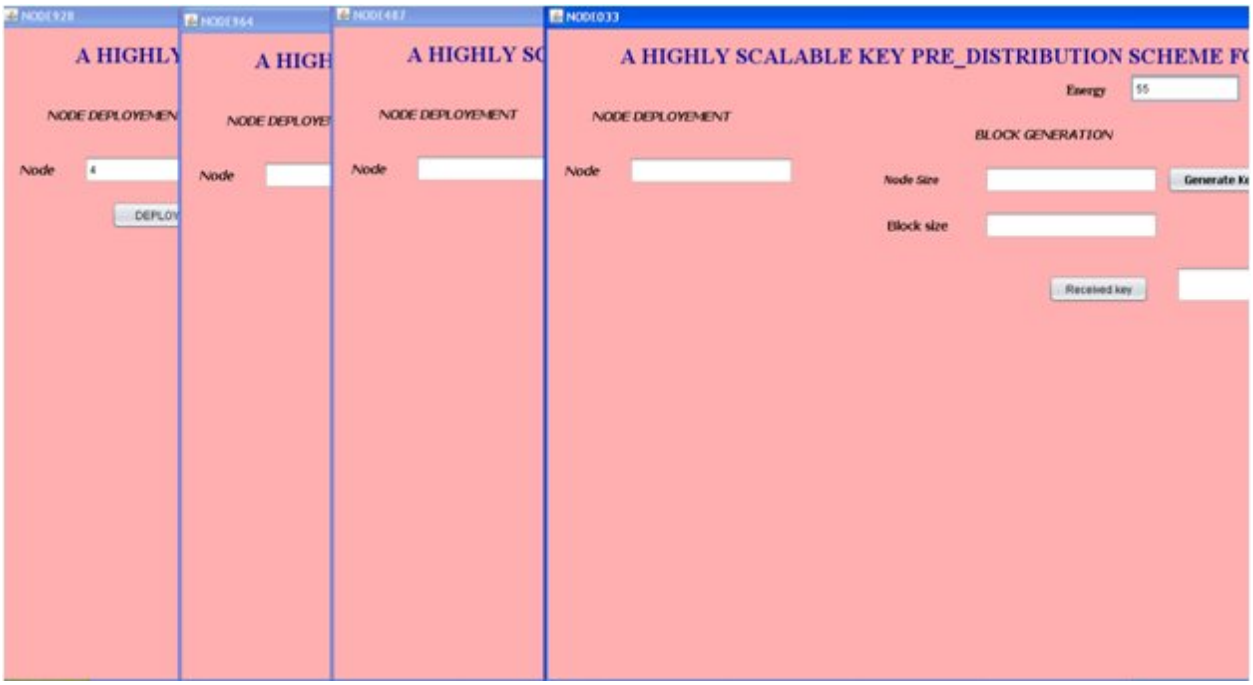

Figure 3.10: Screen shot 4

Figure 3.11: Screen shot 5

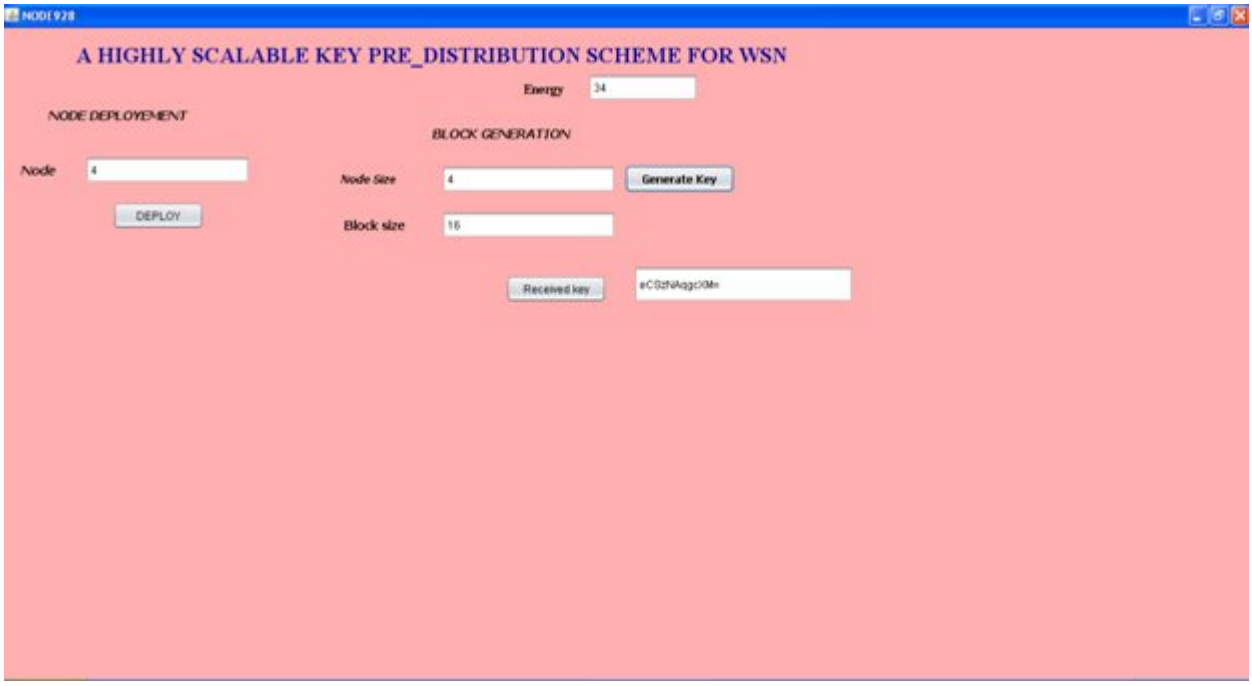

Figure 3.12: Screen shot 6

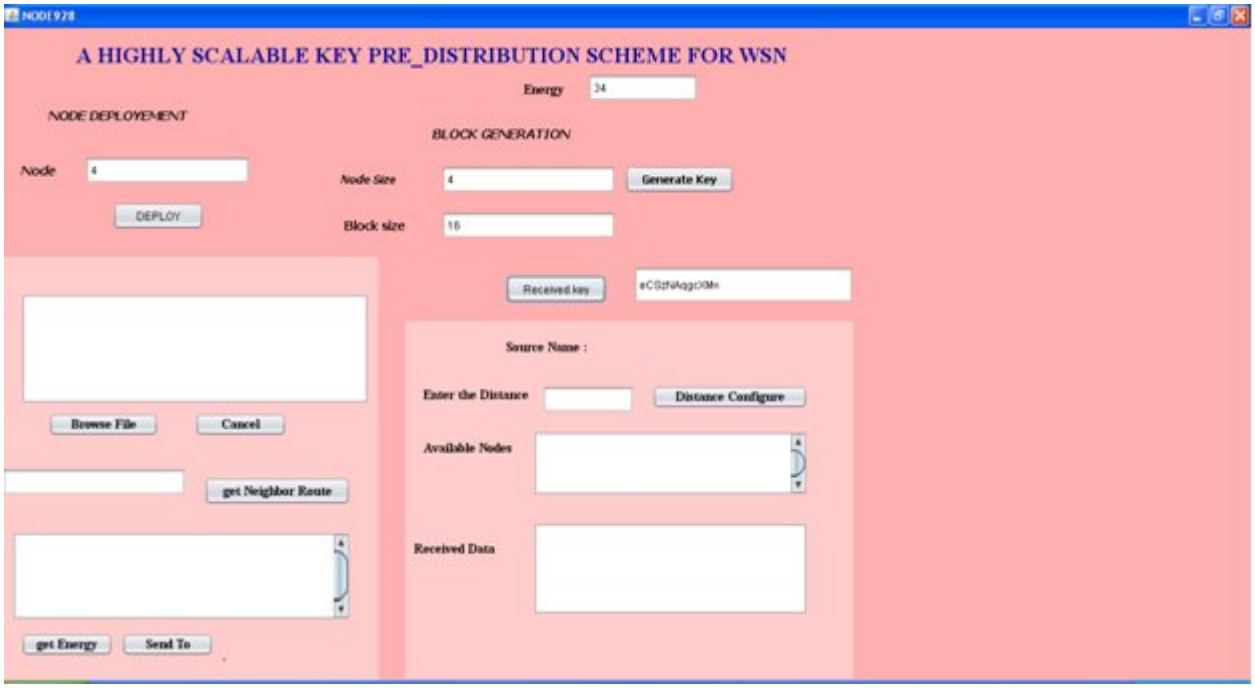

Figure 3.13: Screen shot 7

Chapter 4: CONCLUSION

We proposed, in this work, a scalable key management scheme which ensures a good secure coverage of large scale WSN with a low key storage overhead and a good network resiliency. We make use of the unital design theory. We showed that a basic mapping from unitals to key pre-distribution allows achieving high network scalability while giving a low direct secure connectivity coverage. We proposed then an efficient scalable unital-based key pre-distribution scheme providing high network scalability and good secure connectivity coverage. We discuss the solution parameter and we propose adequate values giving a very good trade-off between network scalability and secure connectivity. We conducted analytical analysis and simulations to compare our new solution to existing ones, the results showed that our approach ensures a high secure coverage of large scale networks while providing good overall performances.

References

[1] Y. Zhou, Y. Fang, and Y. Zhang, “Securing wireless sensor networks: a survey,” *IEEE Commun. Surv. Tuts.*, vol. 10, no. 1–4, pp. 6–28, 2008.

[2] L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks,” in *Proc. 2002 ACM CCS*, pp. 41–47.

[3] H. Chan, A. Perrig, and D. Song, “Random key predistribution schemes for sensor networks,” in *IEEE SP*, pp. 197–213, 2003.

[4] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney, “A key management scheme for wireless sensor networks using deployment knowledge,” in *Proc. 2004 IEEE INFOCOM*, pp. 586–597.

[5] C. Castelluccia and A. Spognardi, “A robust key pre-distribution protocol for multi-phase wireless sensor networks,” in *Proc. 2007 IEEE Securecom*, pp. 351–360.

[6] D. Liu and P. Ning, “Establishing pairwise keys in distributed sensor networks,” in *Proc. 2003 ACM CCS*, pp. 52–61.

[7] Z. Yu and Y. Guan, “A robust group-based key management scheme for wireless sensor networks,” in *Proc. 2005 IEEE WCNC*, pp. 1915–1920.

[8] S. Ruj, A. Nayak, and I. Stojmenovic, “Fully secure pairwise and triple key distribution in wireless sensor networks using combinatorial designs,” in *Proc. 2011 IEEE INFOCOM*, pp. 326–330.

[9] S. Zhu, S. Setia, and S. Jajodia, “Leap: efficient security mechanisms for large-scale distributed sensor networks,” in *Proc. 2003 ACM CCS*, pp. 62–72.

[10] S. A. C¸ amtepe and B. Yener, “Combinatorial design of key distribution mechanisms for wireless sensor networks,” *IEEE/ACM Trans. Netw.*, vol. 15, pp. 346–358, 2007.

[11] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar, “Spins: security protocols for sensor netowrks,” in *Proc. 2001 ACM MOBICOM*, pp. 189–199.

[12] B. Maala, Y. Challal, and A. Bouabdallah, “Hero: hierarchcal key management protocol for heterogeneous WSN,” in *Proc. 2008 IFIP WSAN*, pp. 125–136.

[13] W. Bechkit, Y. Challal, and A. Bouabdallah, “A new scalable key predistribution scheme for WSN,” in *Proc. 2012 IEEE ICCCN*, pp. 1–7.

[14] J. Zhang and V. Varadharajan, “Wireless sensor network key management survey and taxonomy,” *J. Netw. Comput. Appl.*, vol. 33, no. 2, pp. 63–75, 2010.

[15] S. A. C¸ amtepe and B. Yener, “Key distribution mechanisms for wireless sensor networks: a survey,” Technical Report TR-05-07, Mar. 2005.