

**Intrusion Detection in Homogeneous and**  
**Heterogeneous Wireless Sensor Networks**

**A Project Based Learning PROJECT Report**

**for COMPUTER NETWORKS (CSE323)**

*by*

**A PRATHEEK REDDY 14BCE0001**

**CHAPTER NO. TITLE PAGE NO.**

**ABSTRACT iii**

1. **INTRODUCTION 1**

**1.1 Homogeneous WSNs 3**

**1.2** **Heterogeneous WSNs**  **3**

**1.3 Single-sensing detection model 4**  
 **1.4 MULTI-sensing detection model 4**

**2. RELATED WORK 5**

**3. OUR INTRUSION DETECTION MODEL 7**

**4. DETECTION PROBABILITY IN** **HOMOGENEOUS WSNs**

**4.1 SINGLE- SENSING DETECTION MODEL 8**

**5. IMPLEMENTATION 10**

**5.1 Snapshots 12**

**6. RESULTS 19**

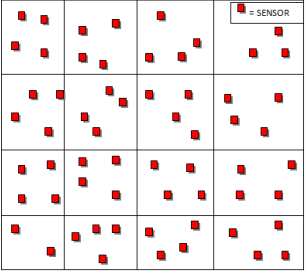
**7. CONCLUSION 20**

**ABSTRACT:**

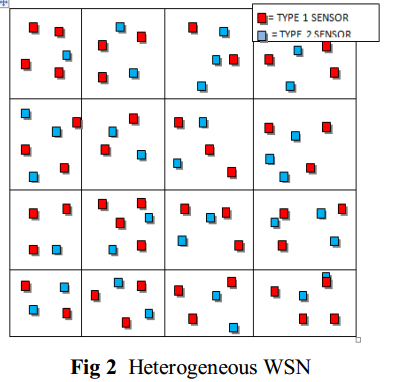
WSNs have become increasing one of the most promising and interesting area over the past few years. The various potential applications in WSNs dictate that they should be secure. Intrusion detection is a mechanism to detect inappropriate and incorrect attackers in WSNs. In this paper we present a mathematical analytical model to find the probability of intrusion detection in homogeneous as well as heterogeneous WSNs. In this paper we have considered two sensing model: single- sensing detection model and multisensing detection model. Finally the simulation results verify  
the mathematical analysis.

**1 INTRODUCTION**  
A WSN is a highly distributed network of wireless devices known as sensor nodes. Each sensor node monitors some physical phenomenon’s (e.g., humidity, temperature, pressure, light) inside the area of deployment. The monitored information is sent to base station through wireless links. The communication range of sensor nodes is limited to tens of meters, so data  
are sent hop-by-hop from one sensor node to another until they reach the base station. WSNs are used in many applications where the sensors have physical interactions with the environment and are accessible by anyone make  
them more vulnerable to security threats. The limitations of WSNs in memory, energy and other resources make the use of existing security techniques infeasible. So we need another defence mechanism “An Intrusion detection  
system” that can protect the network from attackers. There are some probabilities of intrusion detection as, an intruder can be detected as soon as it enters in the network domain or when it covers some distance in the network domain. Detection probability is defined as the probability that an object is detected in certain observation duration. Probability of intrusion detection  
heavily depends on the intrusion distance [5]. Intrusion distance denoted as D can be defined as the distance between the point where the intruder enters the network and the point where it gets detected by a sensor. Some parameters that influence the probability of intrusion detection are:-  
**Node density: -** It is defined as the total number of nodes  
present in the network.

**Transmission range: -** The maximum distance up to which a node can transmit is called transmission range.  
**Sensing range: -** The distance up to which a sensor can detect the presence of intruder is called its sensing range. We consider these three parameters in our model. Depending on sensors capability there are two types of  
WSNs:-  
**1.1 Homogeneous WSNs**  
In homogeneous WSNs (Fig.1) all the sensors have same capabilities. They all have same battery energy, hardware complexity sensing range, and transmission range [3]. Homogeneous WSNs have simple network connectivity  
because of symmetrical wireless link.



* 1. **Heterogeneous WSNs**  
     A heterogeneous WSNs (Fig. 2) consists different sensor nodes. In heterogeneous WSNs some sensors have larger capabilities than other sensors. In this network some sensors have larger sensing range, transmission range and have more battery power. Heterogeneous WSNs are most suitable for real life applications as compared to homogeneous. In heterogeneous WSNs a large no. of inexpensive nodes perform sensing, while a few nodes having comparatively more energy perform other tasks as data filtering, transport. Heterogeneous WSNs have comparatively difficult network connectivity because of asymmetrical wireless links. As packets from high capability nodes may reach the low capability nodes but low capability nodes may not be able to transmit information to high capability nodes. In real world the assumption of homogeneous sensors may not be practical because sensing application may require heterogeneous nodes in terms of sensing and communication capabilities in order to enhance network.



* 1. **Single-sensing detection model**  
     In single-sensing detection model an intruder is successfully detected by single sensor. But in some cases the information provided by singe sensor may not be correct as it can sense only a portion of the network  
     domain. In that case we use multi-sensing detection model.  
     **1.3 Multi-sensing detection model**  
     In multi-sensing system an intruder is detected by multiple collaborating sensors. The no. of sensors depends upon specific applications. For example at least sensors are required to determine the location of intruder. The rest of the paper is organised as follows: in section 2  
     we describe the related work in this field. In section 3 we present our intrusion detection model. In section 4 we present analytical model for homogeneous WSNs. In section 5 the same is done for heterogeneous WSNs. In section 6 the simulation results verify the mathematical  
     analysis. And finally we conclude our paper in section 7.

**2.RELATED WORK**  
There exist several security techniques in WSNs. Intrusion detection system is one of the most important tool in WSNs. Many solutions have been proposed in traditional networks but they cannot be applied directly to WSNs because of restricted resources. Zhang et al., studied the problem of intrusion detection in  
Wireless Ad-hoc Networks. They have proposed architecture for distributed and cooperative intrusion detection system for Ad-hoc Networks, which was based on statistical anomaly detection techniques which requires much time to detect intrusion in data and during traffic. Liu et al. have explored the effects of sensor mobility on sensing coverage and detection capability in a mobile  
WSNs. It is demonstrated that sensor mobility can improve the sensing coverage of the network, and provide fast detection of targeted events.  
Wang et al. have provided a unifying approach in relating the intrusion detection probability with respect to the intrusion distance and the network parameters (i.e., node density, sensing range and transmission range), under  
single-sensing detection and multiple-sensing detection models, in both homogeneous and heterogeneous WSNs. Xi Peng et al proposed a security management model for self organizing wireless sensor networks based on  
intrusion detection. It can prevent most of attacks. Then an analysis of each layer of networks in security model is discussed and the security management measures in the data link layer and network layer are described in detail  
especially. Such a structure is built based on the existing encryption and authentication protocols. Byunggil Lee et al., have developed management  
platform and security framework for WSNs. The proposed framework has advantages as regard secure association and intrusion detection. This also provides the background a wsn, its security issues and requirements.

Qi Wang et al., have developed a intruder detection algorithm of low complexity for static wireless sensor network. The intrusion detection model includes characteristics that determine the average frequency of execution of order. A distributed algorithm in which the sensor collects the information from the neighbouring nodes to analyses the anomalies if any from the  
neighbours. The intrusion detection algorithm on detecting anomalies packets received from its neighbours basic alarms to report the anomaly. K Shaila et al., have proposed an algorithm Secure and Energy Efficient Approach for Detection of Intruder (SEEDI) in homogeneous Wireless Sensor Networks.  
Single sensing and Multi-sensing intruder detection are considered in their algorithm. Simulation results showed that the proposed algorithm resulted in better performance. K Suresh et al., considered this issue according to  
heterogeneous WSNs models. Furthermore, they considered two sensing detection models: single-sensing detection and multiple-sensing detection. Their simulation results shows the advantage of multiple sensor heterogeneous WSNs.

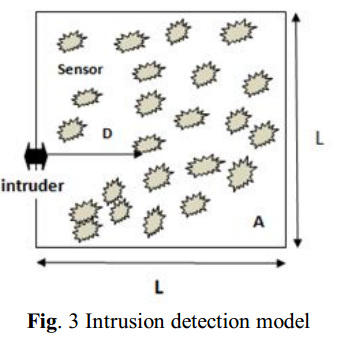
**3.OUR INTRUSION DETECTION MODEL**

In our intrusion detection model we consider a square WSN in two dimensional (2D) plane with area. The intrusion distance is denoted as D (Fig. 3). All the sensors are uniformly distributed in the deployment area. We find out the detection probability by considering three network parameters: node density, sensing range, and transmission range. We consider two types of WSNs. In  
homogeneous WSNs all the sensors have same sensing range denoted as and have same transmission range denoted as . The node density is denoted as λ. In heterogeneous WSNs we have two types of sensors:

**• Type 1** sensors with larger sensing range and longer  
transmission range .The node density is denoted as.

**• Type 2** sensors with smaller sensing range and  
shorter transmission range .The node density is  
denoted as.

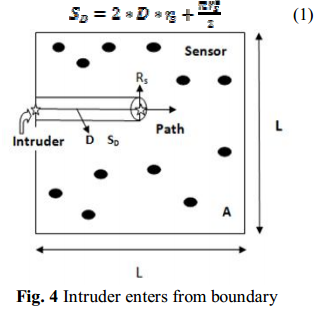
We consider two sensing models: in single-sensing model the intruder can be detected by single sensor and in multi sensor model we consider at least *k* sensors to detect the intruder.



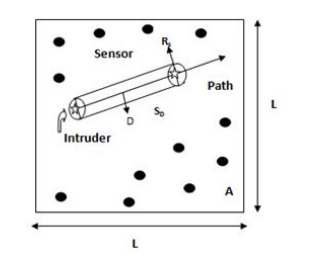
**4. DETECTION PROBABILITY IN** **HOMOGENEOUS WSNs**  
In this section we derive the intrusion detection probability for homogeneous WSNs. we present the analysis in single sensing and multi sensing detection model.

**4.1 SINGLE- SENSING DETECTION MODEL**

In the single-sensing detection model, the intruder can be recognized once it moves into the sensing coverage disk of any sensor. We consider two cases:-  
**CASE I**  
Intruder may access the network domain from any point in the network boundary:- When the intruder starts from a point of the network boundary given an intrusion distance D ≥ 0, the corresponding intrusion detection area SD is almost an oblong area. This area includes a rectangular area with length D and width 2rs and a half disk with radius rs attached to it. It has

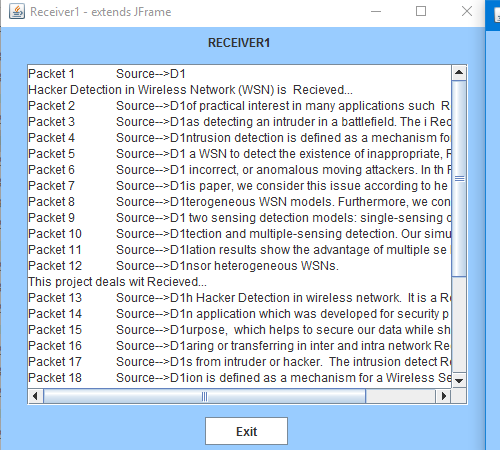


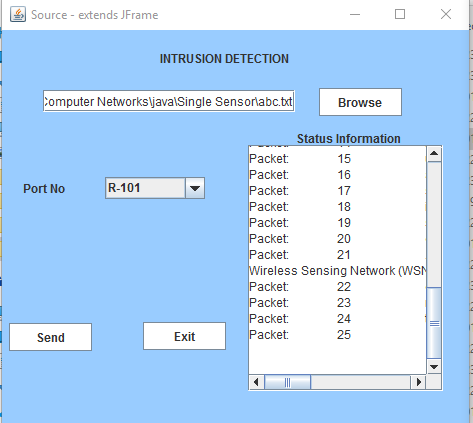
**CASE II**Intruder may access the network from a random point in the network domain (Fig. 5) for example, the intruder can be dropped from the air and starts from any point in the network domain:- When the intruder starts from a random point in the network domain, corresponding intrusion detection area is In the following analysis we focus on when the intruder starts from random point in the network domain.



**Fig. 5** Intruder enters from random point

We first consider the detection probability that the intruder can be immediately detected once it enters the network domain. In other words, it has an intrusion Distance D = 0. The corresponding intrusion detection area. We then have Theorem 1 as follows:

**Intrusion detection Java code:**



Java Source Code:

**import java.awt.\*;**

**import java.awt.event.\*;**

**import javax.swing.\*;**

**import java.io.\*;**

**import java.net.\*;**

**public class Source extends JFrame**

**{**

**// Variables declaration**

**private JLabel jLabel1;**

**private JLabel jLabel2;**

**private JLabel jLabel3;**

**private JTextField jTextField1;**

**private JComboBox jComboBox1;**

**private JTextArea jTextArea1;**

**private JScrollPane jScrollPane1;**

**private JButton jButton1;**

**private JButton jButton2;**

**private JButton jButton3;**

**private JPanel contentPane;**

**String msg="";**

**int flag=1;**

**int flag1=1;**

**Socket n1\_client;**

**String destination;**

**int limit;**

**String a[]={"Select","R-101","R-102","R-103","I-104"};**

**int len;**

**int packets;**

**int rem;**

**// End of variables declaration**

**public Source()**

**{**

**super();**

**initializeComponent();**

**//**

**// TODO: Add any constructor code after initializeComponent call**

**//**

**this.setVisible(true);**

**}**

**private void initializeComponent()**

**{**

**jLabel1 = new JLabel();**

**jLabel2 = new JLabel();**

**jLabel3 = new JLabel();**

**jTextField1 = new JTextField();**

**jComboBox1 = new JComboBox(a);**

**jTextArea1 = new JTextArea();**

**jScrollPane1 = new JScrollPane();**

**jButton1 = new JButton();**

**jButton2 = new JButton();**

**jButton3 = new JButton();**

**contentPane = (JPanel)this.getContentPane();**

**//**

**// jLabel1**

**//**

**jLabel1.setText("INTRUSION DETECTION");**

**//**

**// jLabel2**

**//**

**jLabel2.setText("Port No");**

**//**

**// jLabel3**

**//**

**jLabel3.setText("Status Information");**

**//**

**// jTextField1**

**//**

**jTextField1.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e)**

**{**

**jTextField1\_actionPerformed(e);**

**}**

**});**

**//**

**// jComboBox1**

**//**

**jComboBox1.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e)**

**{**

**jComboBox1\_actionPerformed(e);**

**}**

**});**

**//**

**// jTextArea1**

**//**

**//**

**// jScrollPane1**

**//**

**jScrollPane1.setViewportView(jTextArea1);**

**//**

**// jButton1**

**//**

**jButton1.setBackground(new Color(255, 255, 255));**

**jButton1.setText("Browse");**

**jButton1.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e)**

**{**

**jButton1\_actionPerformed(e);**

**}**

**});**

**//**

**// jButton2**

**//**

**jButton2.setBackground(new Color(255, 255, 255));**

**jButton2.setText("Send");**

**jButton2.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e)**

**{**

**jButton2\_actionPerformed(e);**

**}**

**});**

**//**

**// jButton3**

**//**

**jButton3.setBackground(new Color(255, 255, 255));**

**jButton3.setText("Exit");**

**jButton3.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e)**

**{**

**System.exit(0);**

**}**

**});**

**//**

**// contentPane**

**//**

**contentPane.setLayout(null);**

**contentPane.setBackground(new Color(153, 204,255));**

**addComponent(contentPane, jLabel1, 158,14,136,28);**

**addComponent(contentPane, jLabel2, 21,149,60,18);**

**addComponent(contentPane, jLabel3, 294,97,118,23);**

**addComponent(contentPane, jTextField1, 41,60,252,22);**

**addComponent(contentPane, jComboBox1, 103,147,100,22);**

**addComponent(contentPane, jScrollPane1, 246,115,195,246);**

**addComponent(contentPane, jButton1, 317,58,83,28);**

**addComponent(contentPane, jButton2, 7,293,83,28);**

**addComponent(contentPane, jButton3, 141,292,83,28);**

**//**

**// Source**

**//**

**this.setTitle("Source - extends JFrame");**

**this.setLocation(new Point(66, 48));**

**this.setSize(new Dimension(483, 435));**

**}**

**private void addComponent(Container container,Component c,int x,int y,int width,int height)**

**{**

**c.setBounds(x,y,width,height);**

**container.add(c);**

**}**

**//**

**// TODO: Add any appropriate code in the following Event Handling Methods**

**//**

**private void jTextField1\_actionPerformed(ActionEvent e)**

**{**

**System.out.println("\njTextField1\_actionPerformed(ActionEvent e) called.");**

**// TODO: Add any handling code here**

**}**

**private void jComboBox1\_actionPerformed(ActionEvent e)**

**{**

**System.out.println("\njComboBox1\_actionPerformed(ActionEvent e) called.");**

**Object o = jComboBox1.getSelectedItem();**

**destination=""+o;**

**flag=0;**

**}**

**private void jButton1\_actionPerformed(ActionEvent e)**

**{**

**System.out.println("\njButton1\_actionPerformed(ActionEvent e) called.");**

**// TODO: Add any handling code here**

**try**

**{**

**int b;**

**msg="";**

**FileDialog fd=new FileDialog(this,"Open",FileDialog.LOAD);**

**fd.show();**

**FileInputStream fos=new FileInputStream(fd.getDirectory()+fd.getFile());**

**jTextField1.setText(fd.getDirectory()+fd.getFile());**

**while((b=fos.read())!=-1)**

**{**

**msg+=(char)b;**

**}**

**flag1=0;**

**int len=msg.length();**

**int packets=len/48;**

**int rem=len%48;**

**packets++;**

**String source=jTextField1.getText();**

**jTextArea1.append("\n\nSource Address::"+source+"\n\n");**

**jTextArea1.append("Selected File Path "+fd.getDirectory()+fd.getFile()+"\n\n");**

**jTextArea1.append("Total Length::"+len+"\n\n");**

**}**

**catch (Exception ex)**

**{**

**ex.printStackTrace();**

**}**

**}**

**private void jButton2\_actionPerformed(ActionEvent e)**

**{**

**System.out.println("\njButton2\_actionPerformed(ActionEvent e) called.");**

**// TODO: Add any handling code here**

**String dest;**

**if(flag==0)**

**{**

**if(destination.equalsIgnoreCase(a[0]))**

**{**

**JOptionPane.showMessageDialog(null,"Select the Destination!..");**

**}**

**else if(destination.equalsIgnoreCase(a[1]))**

**{**

**JOptionPane.showMessageDialog(null,"This Is From R-101");**

**dest=setDest(a[1]);**

**sendData("localhost",111,dest);**

**}**

**else if(destination.equalsIgnoreCase(a[2]))**

**{**

**JOptionPane.showMessageDialog(null,"This Is From R-102");**

**dest=setDest(a[2]);**

**sendData("localhost",111,dest);**

**}**

**else if(destination.equalsIgnoreCase(a[3]))**

**{**

**JOptionPane.showMessageDialog(null,"This Is From R-103");**

**dest=setDest(a[3]);**

**sendData("localhost",111,dest);**

**}**

**else if(destination.equalsIgnoreCase(a[4]))**

**{**

**JOptionPane.showMessageDialog(null,"This Is From I-104");**

**dest=setDest(a[4]);**

**sendData("localhost",111,dest);**

**}**

**}**

**else**

**JOptionPane.showMessageDialog(null,"Load the File OR Select the Destination!..");**

**}**

**private void jButton3\_actionPerformed(ActionEvent e)**

**{**

**System.out.println("\njButton3\_actionPerformed(ActionEvent e) called.");**

**// TODO: Add any handling code here**

**}**

**//**

**// TODO: Add any method code to meet your needs in the following area**

**//**

**public void sendData(String name,int port,String dest)**

**{**

**try**

**{**

**n1\_client=new Socket(name,port);**

**DataOutputStream out=new DataOutputStream(n1\_client.getOutputStream());**

**if(out!=null)**

**{**

**out.flush();**

**}**

**int outgoing=0;**

**byte buffer[]=msg.getBytes();**

**int len=buffer.length;**

**int tlength=buffer.length/48;**

**int length11=buffer.length%48;**

**int len1=len;**

**if(length11!=0)**

**{**

**tlength++;**

**}**

**out.writeInt(tlength);**

**out.writeUTF(destination);**

**int st=0;**

**int end=48;**

**jTextArea1.append("Packet Length: "+len+"\n");**

**if(len<=48)**

**{**

**out.writeUTF(dest+msg);**

**jTextArea1.append("Packet: "+"\t"+(++outgoing)+"\t"+msg+"\n");**

**}**

**else**

**{**

**jTextArea1.append("Packet: "+"\t"+(++outgoing)+"\t"+msg.substring(st,end)+"\n");**

**out.writeUTF(dest+msg.substring(st,end));**

**while(len1>48)**

**{**

**len1-=48;**

**if(len1<=48)**

**{**

**jTextArea1.append("Packet: "+"\t"+(++outgoing)+"\t"+msg.substring(end,len)+"\n");**

**out.writeUTF(dest+msg.substring(end,len));**

**}**

**else**

**{**

**int sp=end+48;**

**jTextArea1.append("Packet: "+"\t"+(++outgoing)+"\t"+msg.substring(end,sp)+"\n");**

**out.writeUTF(dest+msg.substring(end,sp));**

**end=sp;**

**}**

**}**

**}**

**}**

**catch (Exception exp)**

**{**

**exp.printStackTrace();**

**}**

**}**

**public String setDest(String Destname)**

**{**

**String Destinationname="";**

**if(Destname.equalsIgnoreCase(a[1]))**

**{**

**Destinationname="Source-->D1";**

**}**

**else if(Destname.equalsIgnoreCase(a[2]))**

**{**

**Destinationname="Source-->D2";**

**}**

**else if(Destname.equalsIgnoreCase(a[3]))**

**{**

**Destinationname="Source-->D3";**

**}**

**else if (Destname.equalsIgnoreCase(a[4]))**

**{**

**Destinationname="Source-->D4";**

**}**

**return Destinationname;**

**}**

**public static void main(String[] args)**

**{**

**new Source();**

**}**

**CONCLUSION**

As the WSN becomes necessary and used frequently for many applications, the need for securing them is also increasing due to the nature of their deployment and their resource restrictions