# PROTECTION OF MEDICAL DATA USING CRYPTOSYSTEM

**PROJECT REPORT**

**Submitted in fulfilment for**

# INFORMATION AND SYSTEM SECURITY

**By**

**V RITHIK - 19MIS1189**

**Under the Guidance of**

**PROF. RAJESH KUMAR**

**School of Information Technology & Engineering**



**TABLE OF CONTENT**

1. **Title**
   1. **Introduction**
   2. **Aim**
   3. **Existing System**
   4. **Proposed System**
2. **Problem statement of the application**
3. **System Planning And Design**
   1. **Need for project**
   2. **Overall system Design Architecture**
   3. **Module Description**
   4. **Sequence Diagram**
4. **System Implementation**
   1. **Description**
   2. **Methodology**
   3. **Sample Code**
5. **Reports**
   1. **Sample Screens**
      1. **Doctor Node**
      2. **Key Generation**
      3. **Server 1**
      4. **Server 2**
      5. **Server 3**
      6. **Sensor Node**
      7. **Server-1 Encryption**
      8. **Server-2 Encryption**
      9. **Server-3 Encryption**
      10. **Upload File**
      11. **Elgamal Decryption**
      12. **Paillier Decryption**
6. **Conclusion**

# TITLE “PROTECTION OF MEDICAL DATA USING

**CRYPTOSYSTEM”**

**1.1. Introduction —**

To monitor the environment conditions in the wireless sensor networks use of large number of sensors and this can pass the information to the main location and this wireless sensor networks are mainly motivated by military applications and this sensor networks became very famous today and also used in many consumer and industrial areas, and now a days the promising fields like healthcare applications are maintained very well by these wireless sensor networks. now a days Wireless medical sensor networks certainly improve quality-of-care, so privacy is ensured.

* 1. **Aim —**

The objective of ‘PRIVACY PROTECTION OF MEDICAL DATA USING

CRYPTOSYSTEMS’ is to protect the patient details using some cryptographic algorithms. It also preserve and protect data not only attacks from outside but also the inside attacks.

* 1. **Existing System –**

To protect the wireless medical sensor networks against various attacks, a lot of work has been done. But the system is not cover secure healthcare monitoring using wireless sensor networks. Current solutions are built on either secret-key encryption or public-key encryption as Secret-key based solutions assume that the secret keys for encryption and authentication are deployed in the medical sensors and the servers in advance. . A secret key cryptosystem, such as Advanced Encryption Standard (AES) , is used for encryption, while the message authentication code (MAC) is used for authentication. Typical examples of secret key based solutions include. These solutions are usually efficient.

However, the distribution of the secret keys are less efficient than the public-key based solution.

Public-key based solutions assume that a public-key cryptosystem, such as Difﬁe- Hellman key exchange protocol , is used to establish a secret key for encryption on the basis of the public keys.

These solutions facilitate key distribution and update. However, they are usually inefficient and not directly applicable to the wireless medical sensor networks, where the sensors have limited computation and communication and also

* + - Current WMSN healthcare research trends focus on patient reliable communication, patient mobility and energy-efficient routing.
    - Wireless medical sensor networks are more vulnerable to eavesdropping, modification, impersonation and replaying attacks than the wired networks.
    - So protection is an important task for patient data. In existing Symmetric key and Asymmetric key cryptography techniques used for protection.
  1. **Proposed System**
     + To prevent the patient data from the inside attacks, this propose a new data collection technique, where a sensor splits the sensitive patient data into three components according to a random number generator based on hash function and sends them to three servers, respective, via secure channels.
     + To keep the privacy of the patient data in data access, proposed a new data access technique on the basis of the Paillier cryptosystem. The protocol allows the user (e.g., physician) to access the patient data without revealing it to any data server.
     + To preserve the privacy of the patient data in statistical analysis, proposed some new privacy-preserving statistical analysis protocols on the basis of the Paillier and ElGamal cryptosystems. These protocols allow the user (e.g., medical researcher) to perform statistical analysis on the patient data without compromising the patient data privacy.

## PROBLEM STATEMENT OF THE APPLICATION

Wireless Sensor Networks (WSN) is an emerging technology that has the potential to transform the way of human life. Healthcare applications are considered promising fields for Wireless Medical Sensor Network, where patient’s health can be monitored using Medical Sensors. Wireless Medical Sensor Networks (WMSNs) are the key enabling technology in healthcare applications that allows the data of a patient’s vital body parameters to be collected by wearable biosensors. Current WMSN healthcare research trends focus on patient reliable communication, patient mobility and energy-efficient routing. Wireless medical sensor networks are more vulnerable to eavesdropping

,modification, impersonation and replaying attacks than the wired networks. So protection is an important task for patient data.

* 1. **Need for Project**

## SYSTEM PLANNING AND DESIGN

The basic need of this project is now a days data has been undergoing attacks and also number of attackers and also the attacking techniques have been improvised and also medical information is more than 10 times useful than credit card details on black market. Attackers discovers more methods so health care becoming a target because it contains personal data like number, address etc. so there is a need of privacy protection for that patient data so in this project we are protecting the patient data not only from outside but also from inside .

**Front-end / Client Side**

"Front-end" typically means the parts of the project a user interacts with--such as the graphical user interface or command line.

The top most visible layer is what’s called the Frontend. This is usually written in java using NetBeans IDE.

**Back-end / Server side**

"Back-end" means the parts that do the work, but the user is unaware of or cannot see. Databases, services, etc

* 1. **Overall System Design Architecture –**



Elgamal Encryption

Upload

Paillier Decryption

**Server 3**

Paillier Encryption

Elgamal Decryption

**Server 2**

**Sensor Nodes**

**Doctor**

**Server 1**

### MODULE DESCRIPTION

* + - Network Formation, Paillier & Elgamal Key Generation
    - Paillier Encryption & Upload Sensed Data
    - Elgamal Encryption
    - Elgamal & Paillier Decryption

### NETWORK FORMATION ,PALLIER & ELGAMAL GENERATION

* + - * In this module, generation of wireless medical sensor network is done.
      * It contains n sensor nodes, 3 servers and one doctor node.
      * These sensor nodes worn by a patient which provide physiological sensing to 3 servers. Then doctor can access these sensed data from servers.
      * In this module, first sensor nodes and servers are connect with doctors. At the connection time, doctor generate the Paillier and Elgamal Public & Private Keys. Then send the Paillier Public Key to nodes for Paillier Encryption and send the Elgamal Public Key to Servers for Elgamal Encryption.

### PAILLIER ENCRYPTION AND UPLOAD SENSED DATA

* + - * In this module, Sensor nodes sense physiological data. Then split these data to 3 blocks for 3 servers.
      * Then it encrypts these blocks based on Paillier Public Key.
      * Followed by, it forward these encrypted blocks to 3 servers respectively.

### ELGAMAL ENCRYPTION

* + - * To protect uploaded data against inside attacks, servers can encrypt once.
      * Here servers apply Elgamal encryption using Elgamal Public Key.
      * So these uploaded sensed data have more security against inside attacks.
    1. ELGAMAL & PAILLIER DECRYPTION
       - In this module, doctor wants to access these uploaded sensed data.
       - So he downloads all sensed data from all 3 servers.
       - First he decrypts the sensed data based on Elgamal decryption using Elgamal Private Key.
       - After Elgamal Decryption, doctor decrypts the sensed data based on Paillier Decryption using Paillier Private Key.
       - Finally the doctor gets the original sensed dat

1. **SYSTEM IMPLEMENTATION**

The implementation of the system mainly consists of many methodologies and the techniques used for the system to perform the overall process in the project.

* 1. **DESCRIPTION**

It contains n sensor nodes, 3 servers and one doctor node. These sensor nodes worn by a patient which provide physiological sensing to 3 servers. Then doctor can access these sensed data from servers.In this , first sensor nodes and servers are connect with doctors. At the connection time, doctor generate the Paillier and Elgamal Public & Private Keys. Then send the Paillier Public Key to nodes for Paillier Encryption and send the Elgamal Public Key to Servers for Elgamal Encryption. and Sensor nodes sense physiological data. Then split these data to 3 blocks for 3 servers .Then it encrypts these blocks based on Paillier Public Key. Followed by, it forward these encrypted blocks to 3 servers respectively.To protect uploaded data against inside attacks, servers can encrypt once.

Here servers apply Elgamal encryption using Elgamal Public Key.

So these uploaded sensed data have more security against inside attacks.

In this , doctor wants to access these uploaded sensed data. So he downloads all sensed data from all 3 servers. First decrypts the sensed data based on Elgamal decryption using Elgamal Private Key. After Elgamal Decryption, doctor decrypts the sensed data based on Paillier Decryption using Paillier Private Key. Finally gets the original sensed data.

* 1. **METHODOLOGY**

In this usage of cryptosystems like Elgamal and paillier this can be shown and El\_Gamal is a public-key cryptosystem technique was designed by Dr. Taher Elgamal El\_Gamal depends on the one way function, means that the encryption and decryption are done in separate functions. The encryption process requires two modular exponentiations (extra time).

**Key generation**

* + - Generate a large random prime number (p)
    - Choose a generator number (a)
    - Choose an integer (x) less than (p-2) ,as secret number.
    - Compute (d) where
      * d= a**x** mod p
    - Determine the public key (p, a, d), and the private key (x)

**Encryption**

* + - Obtain the public key (p , a , d ) from
    - the receiver A.
    - Choose an integer k such that :
      * 1 < k < p-2
    - Represent the plaintext as an integer m where 0 < m < p-1
    - compute (y) as follows :
      * y = ak mod p
    - compute (z) as follows :
      * z = (dk \* m ) mod p
    - Find the ciphertext (C) as follows :
      * C= ( y , z )
    - The sender B send C to The receiver A .

**Decryption**

* Obtain the ciphertext (C) from B .
* compute (r) as follows :
  + r = yp-1-x mod p
* Recover the plaintext as follows:
  + m = ( r \* z ) mod p
  1. **CODE**

SENSOR DATA:

package privacymedicalsensornodes; import java.io.BufferedReader; import java.io.BufferedWriter; import java.io.DataInputStream; import java.io.File;

import java.io.FileInputStream; import java.io.FileWriter;

import java.io.InputStreamReader; import java.math.BigInteger; import java.net.DatagramPacket; import java.net.DatagramSocket; import java.net.InetAddress;

import java.security.SignatureException; import java.sql.ResultSet;

import java.util.ArrayList; import java.util.Random; import java.util.Scanner; import java.util.Vector; import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec; import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

public class SensorNodeFrame extends javax.swing.JFrame { int snid;

ArrayList path=new ArrayList(); String inputfile;

public SensorNodeFrame(int id)

{ initComponents(); snid=id;

}

private void initComponents() {

jPanel1 = new javax.swing.JPanel(); jLabel1 = new javax.swing.JLabel();

jTabbedPane1 = new javax.swing.JTabbedPane(); jPanel2 = new javax.swing.JPanel();

jButton1 = new javax.swing.JButton(); jLabel2 = new javax.swing.JLabel(); jTextField1 = new javax.swing.JTextField(); jPanel3 = new javax.swing.JPanel(); jScrollPane1 = new javax.swing.JScrollPane(); jTable1 = new javax.swing.JTable();

jPanel5 = new javax.swing.JPanel(); jLabel4 = new javax.swing.JLabel();

jTextField3 = new javax.swing.JTextField(); jButton3 = new javax.swing.JButton(); jScrollPane2 = new javax.swing.JScrollPane(); jTextArea1 = new javax.swing.JTextArea(); jButton4 = new javax.swing.JButton(); jButton5 = new javax.swing.JButton(); jButton6 = new javax.swing.JButton(); jButton7 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE); jPanel1.setBackground(new java.awt.Color(0, 102, 0));

jLabel1.setFont(new java.awt.Font("Andalus", 0, 36)); // NOI18N

jLabel1.setForeground(new java.awt.Color(255, 255, 255)); jLabel1.setText("Sensor Node");

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1); jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup( jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(200, 200, 200)

.addComponent(jLabel1)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

jPanel1Layout.setVerticalGroup( jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel1))

);

jButton1.setText("Connect");

jButton1.addActionListener(new java.awt.event.ActionListener() { public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jLabel2.setText("Paillier Public Key");

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2); jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup( jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap(42, Short.MAX\_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jTextField1)

.addComponent(jButton1, javax.swing.GroupLayout.DEFAULT\_SIZE,512, Short.MAX\_VALUE))

.addGap(42, 42, 42))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()

.addComponent(jLabel2)

.addGap(259, 259, 259))))

);

jPanel2Layout.setVerticalGroup( jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(42, 42, 42)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE,50, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(44, 44, 44)

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,47, Short.MAX\_VALUE)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(78, 78, 78))

);

jTabbedPane1.addTab("Connect", jPanel2); jTable1.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] { "Server Id"

}

));

jScrollPane1.setViewportView(jTable1);

javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3); jPanel3.setLayout(jPanel3Layout);

jPanel3Layout.setHorizontalGroup( jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()

.addGap(28, 28, 28)

.addComponent(jScrollPane1,javax.swing.GroupLayout.PREFERRED\_SIZE,541, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(27, Short.MAX\_VALUE))

);

jPanel3Layout.setVerticalGroup( jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()

.addGap(24, 24, 24)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE,253, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(22, Short.MAX\_VALUE))

);

jTabbedPane1.addTab("Server Details", jPanel3); jLabel4.setText("Enter the Filename:"); jButton3.setText("Browse");

jButton3.addActionListener(new java.awt.event.ActionListener() { public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

jTextArea1.setColumns(20); jTextArea1.setRows(5); jScrollPane2.setViewportView(jTextArea1); jButton4.setText("Encrypt");

jButton4.addActionListener(new java.awt.event.ActionListener() { public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton4ActionPerformed(evt);

}

});

jButton5.setText("Upload");

jButton5.addActionListener(new java.awt.event.ActionListener() { public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton5ActionPerformed(evt);

}

});

jButton6.setText("Clear");

jButton6.addActionListener(new java.awt.event.ActionListener() { public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton6ActionPerformed(evt);

}

});

jButton7.setText("Split");

jButton7.addActionListener(new java.awt.event.ActionListener() { public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton7ActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5); jPanel5.setLayout(jPanel5Layout);

jPanel5Layout.setHorizontalGroup( jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel5Layout.createSequentialGroup()

.addGap(31, 31, 31)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(jPanel5Layout.createSequentialGroup()

.addComponent(jButton7, javax.swing.GroupLayout.PREFERRED\_SIZE,124, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jButton4, javax.swing.GroupLayout.PREFERRED\_SIZE,130, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jButton5, javax.swing.GroupLayout.PREFERRED\_SIZE,114, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton6, javax.swing.GroupLayout.PREFERRED\_SIZE,103, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel5Layout.createSequentialGroup()

.addComponent(jLabel4)

.addGap(35, 35, 35)

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE,273, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(27, 27, 27)

.addComponent(jButton3, javax.swing.GroupLayout.PREFERRED\_SIZE,85, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(jScrollPane2))

.addContainerGap(32, Short.MAX\_VALUE))

);

jPanel5Layout.setVerticalGroup( jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel5Layout.createSequentialGroup()

.addGap(21, 21, 21)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel4)

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE,

javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton3))

.addGap(18, 18, 18)

.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE,158, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,29, Short.MAX\_VALUE)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton4)

.addComponent(jButton5)

.addComponent(jButton6)

.addComponent(jButton7))

.addGap(23, 23, 23))

);

jTabbedPane1.addTab("Encrypt & Upload", jPanel5);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane()); getContentPane().setLayout(layout);

layout.setHorizontalGroup( layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(layout.createSequentialGroup()

.addGap(36, 36, 36)

.addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED\_SIZE,601, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(32, Short.MAX\_VALUE))

);

layout.setVerticalGroup( layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,32, Short.MAX\_VALUE)

.addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED\_SIZE,329, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(29, 29, 29))

);

pack();

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) String msg="ConnectSensor#"+snid;

int pt=4000;

packetTransmission(msg,pt); JOptionPane.showMessageDialog(this,"Connected Successfully!"); jButton1.setEnabled(false);

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) try

{

JFileChooser fc=new JFileChooser(); int result=fc.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION)

{

File file=fc.getSelectedFile(); jTextField3.setText(file.getName()); inputfile=file.getAbsolutePath().trim(); String fname=file.getAbsolutePath(); File fe=new File(fname);

FileInputStream fis=new FileInputStream(fe); byte data[]=new byte[fis.available()]; fis.read(data);

fis.close();

String str=new String(data); jTextArea1.setText(str.trim().replaceAll("#",""));

}

else if (result == JFileChooser.CANCEL\_OPTION)

{

JOptionPane.showMessageDialog(this,"Open Command cancelled by user.");

}

}

catch(Exception e)

{

e.printStackTrace();

}

} private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) try

{

int nof=jTable1.getRowCount(); File file = new File(inputfile);

Scanner scanner = new Scanner(file); int count = 0;

String filname=file.getName();

String fi=filname.substring(0,filname.lastIndexOf('.')); while (scanner.hasNextLine())

{

scanner.nextLine(); count++;

}

System.out.println("Lines in the file: " + count);

if(nof<=count)

{

double temp = (count/nof); double nol=temp+1;

System.out.println("No. of lines to be generated for each file:"+nol); FileInputStream fstream = new FileInputStream(inputfile); DataInputStream in = new DataInputStream(fstream);

BufferedReader br = new BufferedReader(new InputStreamReader(in)); String strLine="";

for (int j=1;j<=nof;j++)

{

if(j==nof)

{

nol=(count-nol\*nof)+nol; System.out.println("Last "+nol);

}

FileWriter fstream1 = new FileWriter("FILE/"+fi+""+j+".txt"); if(!(path.contains("FILE/"+fi+""+j+".txt")))

{

path.add("FILE/"+fi+""+j+".txt");

}

BufferedWriter out = new BufferedWriter(fstream1); for (int i=1;i<=nol;i++)

{

strLine = br.readLine(); if (strLine!= null)

{

out.write(strLine); if(i!=nol)

{

out.newLine();

}

}

}

out.close();

}

in.close();

}

else

{

JOptionPane.showMessageDialog(this,"No of Lines in the file "+count+" So don't ask for too much!");

}

System.out.println("paths: "+path);

String ablck="";

for(int p=0;p<path.size();p++)

{

String p1=path.get(p).toString(); String fname=p1;

File fe=new File(fname);

FileInputStream fis=new FileInputStream(fe); byte data[]=new byte[fis.available()]; fis.read(data);

fis.close();

String str1=new String(data); ablck=ablck+"\n=======================\nBlock -

"+(p+1)+"\n=======================\n"+str1.trim()+"\n";

}

JOptionPane.showMessageDialog(this,"Splitted Successfully!"); jTextArea1.setText(ablck.trim());

}

catch(Exception e)

{

e.printStackTrace();

}

}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) jTextField3.setText("");;

jTextArea1.setText("");

}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt)

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)

}

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

for(javax.swing.UIManager.LookAndFeelInfoinfo : javax.swing.UIManager.getInstalledLookAndFeels()) {

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

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(SensorNodeFrame.class.getName()).log(java.util.logging.Level. SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(SensorNodeFrame.class.getName()).log(java.util.logging.Level. SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(SensorNodeFrame.class.getName()).log(java.util.logging.Level. SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(SensorNodeFrame.class.getName()).log(java.util.logging.Level. SEVERE, null, ex);

}

java.awt.EventQueue.invokeLater(new Runnable() { public void run() {

//new SensorNodeFrame().setVisible(true);

}

});

}

private javax.swing.JButton jButton1; private javax.swing.JButton jButton3; private javax.swing.JButton jButton4; private javax.swing.JButton jButton5; private javax.swing.JButton jButton6; private javax.swing.JButton jButton7; public javax.swing.JLabel jLabel1; private javax.swing.JLabel jLabel2; private javax.swing.JLabel jLabel4; private javax.swing.JPanel jPanel1; private javax.swing.JPanel jPanel2; private javax.swing.JPanel jPanel3; private javax.swing.JPanel jPanel5;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2; private javax.swing.JTabbedPane jTabbedPane1; public javax.swing.JTable jTable1;

private javax.swing.JTextArea jTextArea1; public javax.swing.JTextField jTextField1; private javax.swing.JTextField jTextField3;

private void packetTransmission(String msg, int pt) { try

{

byte data1[]=msg.getBytes(); DatagramSocket ds1=new DatagramSocket();

DatagramPacket dp1=new

DatagramPacket(data1,0,data1.length,InetAddress.getByName("127.0.0.1"),pt); ds1.send(dp1);

System.out.println("Port is "+pt+"\n");

}

catch(Exception e)

{

e.printStackTrace();

}

}

public static class Signature

{

private static final String HMAC\_ALGORITHM = "HmacSHA1";

public static String calculateRFC2104HMAC(String data, String key) throws java.security.SignatureException

{

String result; try

{

SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), HMAC\_ALGORITHM);

Mac mac = Mac.getInstance(HMAC\_ALGORITHM); mac.init(signingKey);

byte[] rawHmac = mac.doFinal(data.getBytes());

result = new String(encode(rawHmac));

}

catch (Exception e)

{

throw new SignatureException("Failed to generate HMAC : " + e.getMessage());

}

return result;

}

private static char[] encode(byte[] bytes)

{

final int amount = bytes.length; char[] result = new char[2 \* amount]; int j = 0;

for (int i = 0; i < amount; i++)

{

result[j++] = HEX[(0xF0 & bytes[i]) >>> 4]; result[j++] = HEX[(0x0F & bytes[i])];

}

return result;

}

private static final char[] HEX = {

'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'

};

}}

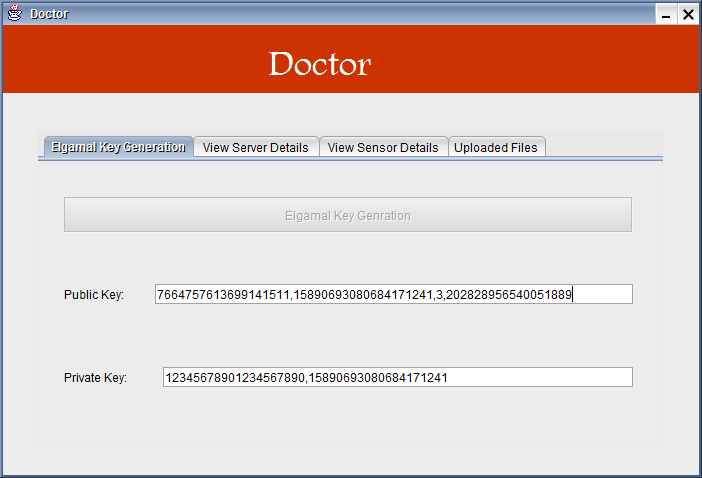
1. **REPORTS**
   1. **SAMPLE SCREENS**

Snapshot is nothing but every moment of the application while running. It gives the clear elaboration of application. It will be useful for the new user to understand the future steps.

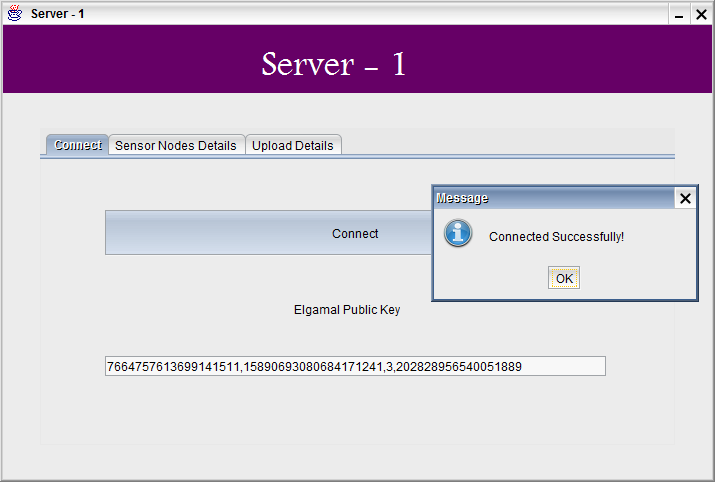
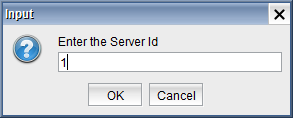
* + 1. **DOCTOR NODE**



* + 1. **KEY GENERATION**



* + 1. **SERVER 1**



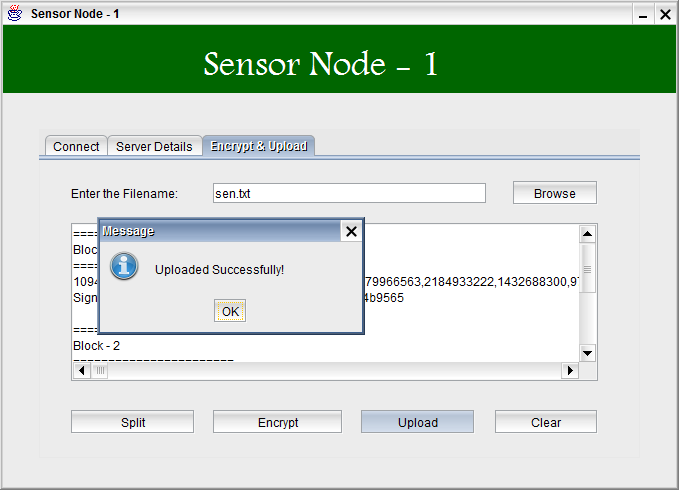
* + 1. **SERVER 2**



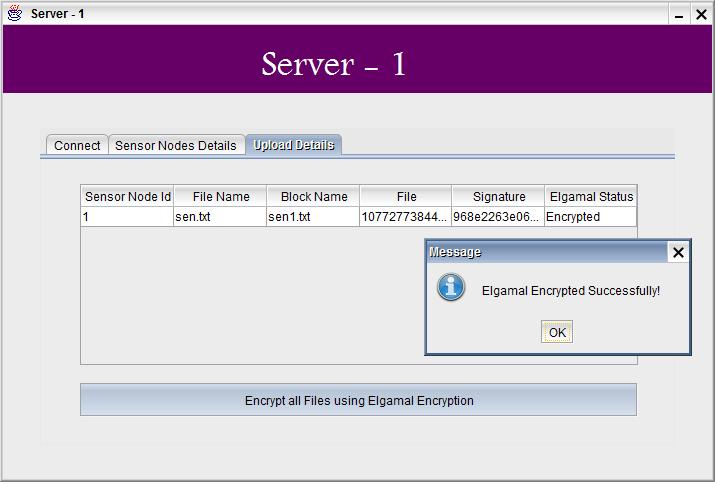
* + 1. **SERVER 3**



* + 1. **SENSOR NODE**

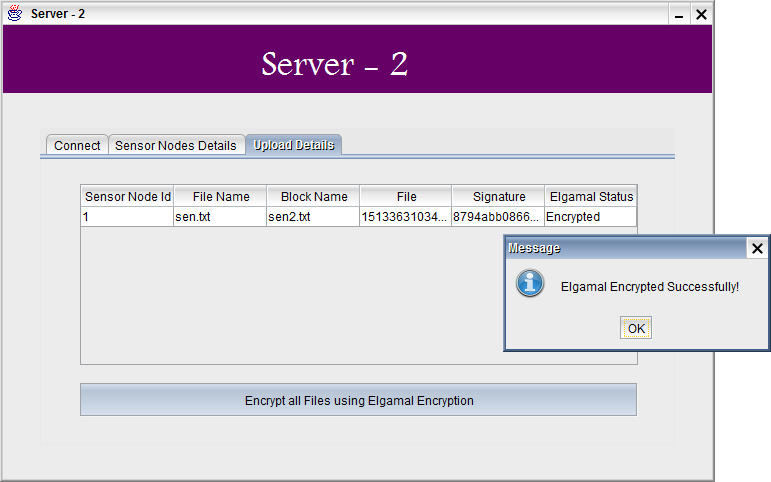


* + 1. **SERVER -1 ENCRYPTION**

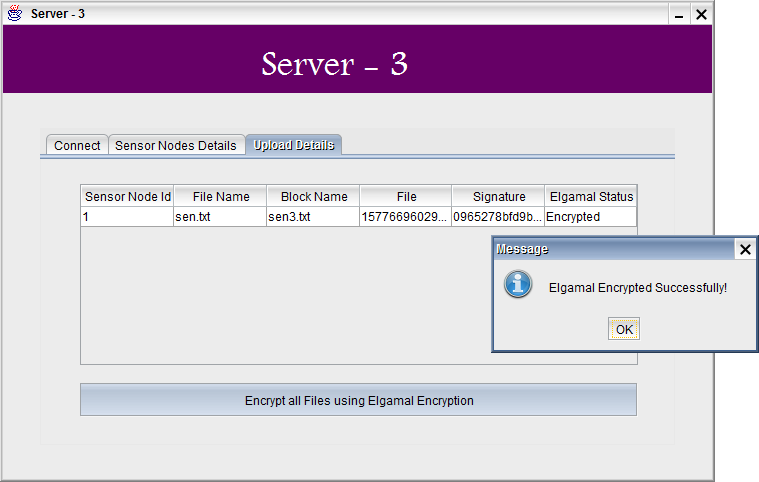


**0**

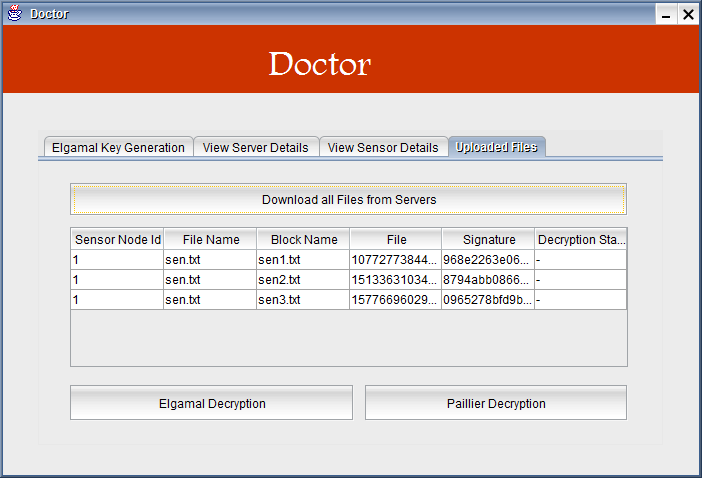
* + 1. **SERVER-2 ENCRYPTION**



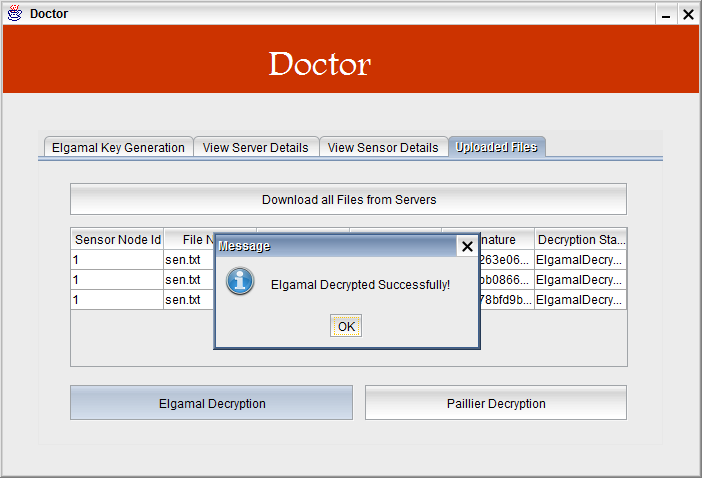
* + 1. **SERVER-3 ENCRYPTION**



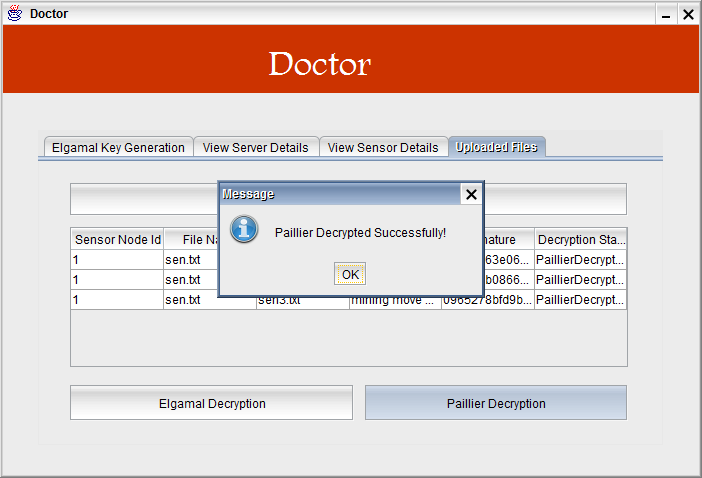
* + 1. **UPLOADED FILES**



* + 1. **ELGAMAL DECRYPTION**



* + 1. **PAILLIER DECRYPTION**



1. **CONCLUSION**

To secure the communication between medical sensors and data servers, used the lightweight encryption scheme and MAC generation scheme based on SHA-3 proposed. To keep the privacy of the patient data, proposed a new data collection protocol which splits the patient data into three numbers and stores them in three data servers, respectively. As long as one data server is not compromised, the privacy of the patient data can be preserved. For the legitimate user (e.g., physician) to access the patient data, proposed an access control protocol, where three data servers cooperate to provide the user with the patient data, but do not know what it is. For the legitimate user (e.g., medical researcher) to perform statistical analysis on the patient data, proposed some new protocols for average, correlation, variance and regression analysis, where the three data servers cooperate to process the patient data without disclosing the patient privacy and then provide the user with the statistical analysis results. Security and privacy analysis has shown that protocols are secure against both outside and inside attacks as long as one data server is not compromised. Performance analysis has shown that our protocols are practical as well. Unlike, the solution can preserve the patient data privacy as long as one of three data server is not compromised.