Al-imam Mohammad Ibn Saud
Islamic University
Collage of Computer
and Information Science

XSecure

Student name:

Aljuhara Alsadoun - 440023448
Aliah Alhameed - 440020584
Rawan Alshalawi - 440018784
Sara Almashharawi - 440028560

Instructor name:

Dr. Tahani Albalawi

Section: 372

Submission Date:
8/5/2021

Table of Contents

1.0 Overview:

In this project we create XSecure application for sharing text in secure way and send it through the network. We use AES to implement encrypt/decrypt, and the application uses symmetric and asymmetric key and encrypt/decrypt the keys using RSA. The system well encrypts the text using AES algorithm with alice public key, and decrypt the text using AES algorithm with alice private key (Figure1).

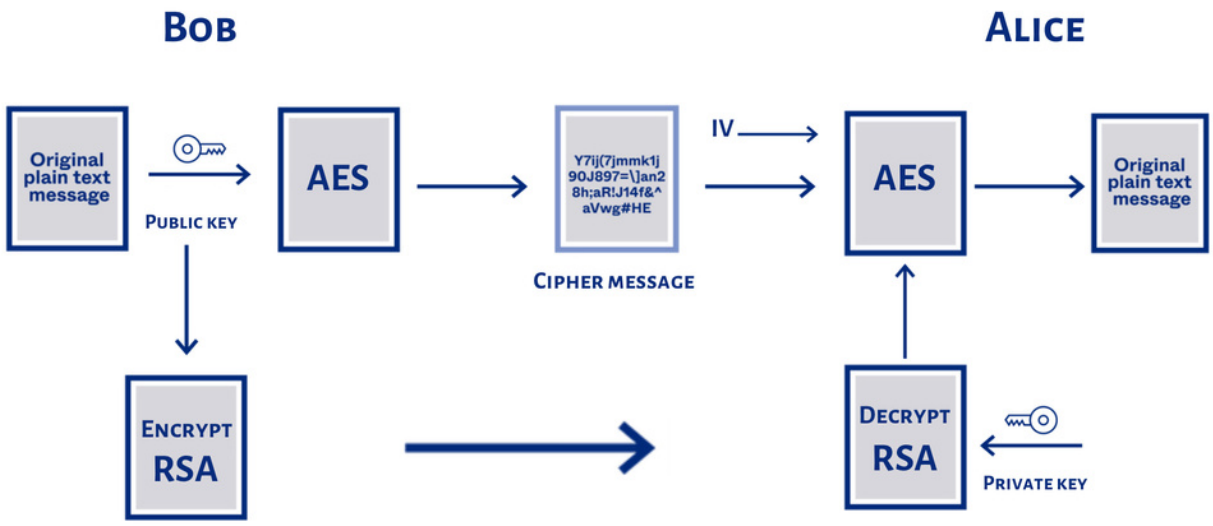


Figure 1

In this project we use java programming language. The GUI of our application contains four windows, the first window (Figure2) able the users to sign in and sign-up using username and password, the system well generates pairs of key, public key, and private key for each user and save the password after applying sha-512, after the user Enter the System window (Figure3) able the user selects the crypto type. In the encrypt window (Figure4) the sender well enters the plain text and select the public key for receiver then the system will encrypt the text using AES, and encrypt the AES key using RSA with the public key, then send the cipher text and encrypted AES key to the receiver , the receiver well receives the cipher text in the decrypt window (Figure5) then selects the private key to decrypt AES key using RSA with his private key ,then decrypt the text using AES.

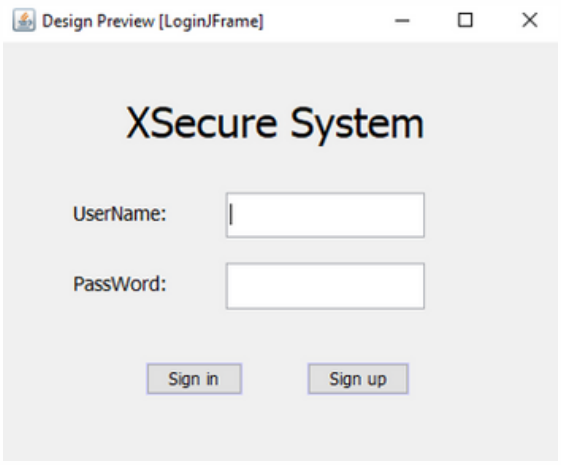


Figure 2

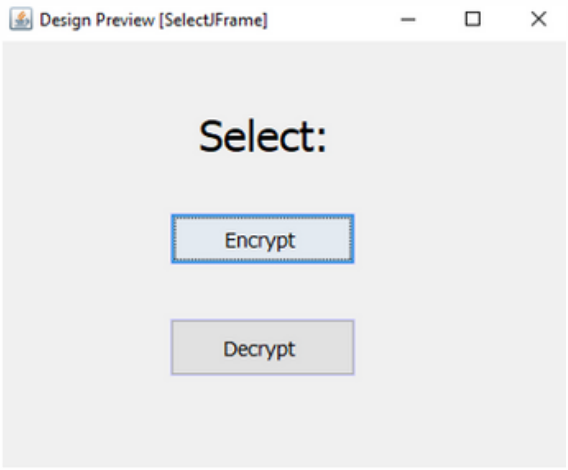


Figure 3

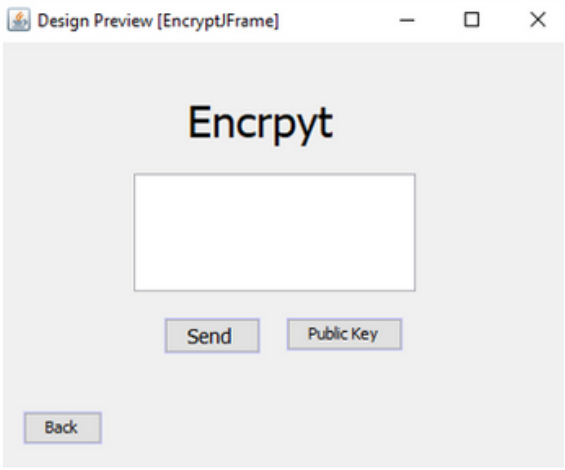


Figure 4

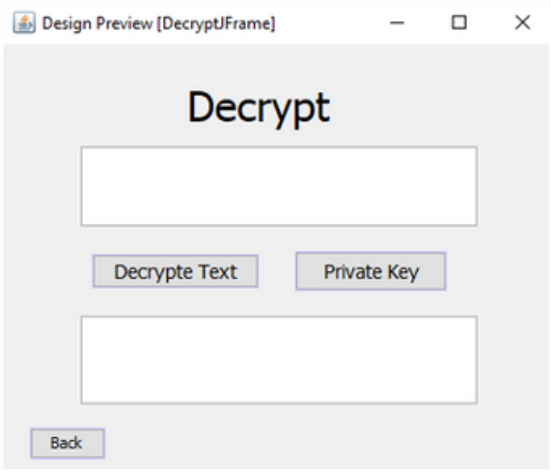


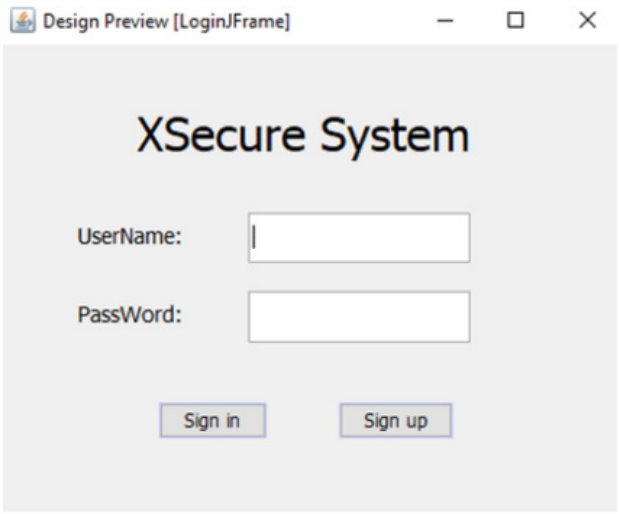
Figure 5

2.0 Implementation Steps:

Login window :
1- Log in the user by select sign in or sign up:

- If a new sign up is selected :
The username and password will be passed through method userInfo. The username will be registered directly. The password will be hashed in order to keep it encrypted by passing it to method generatHash. This method will receive the password and will hash it using the SHA-512 algorithm and will return it as String to method userInfo After the password is securely saved
Two public and private keys will be generated for each new user that has used KeyPairGeneratorin.java.security
It is an engine class capable of generating a private key and its associated public key using the RSA algorithm. After the two keys are generated, the two keys are securely stored in the file. The public key is saved using the java.security.spec.X509EncodedKeySpec library.
By using the X509EncodedKeySpec object, the getEncoded() method returns the X.509 encoded key, after which the username and public key are saved to the file using FileOutputStream.
As for the private key, it will be saved in the same way as the public key, but the library java.security.spec.PKCS8EncodedKeySpec will be used. The getEncoded() method returns an encryption for the private key
Thus, the two keys are kept securely.

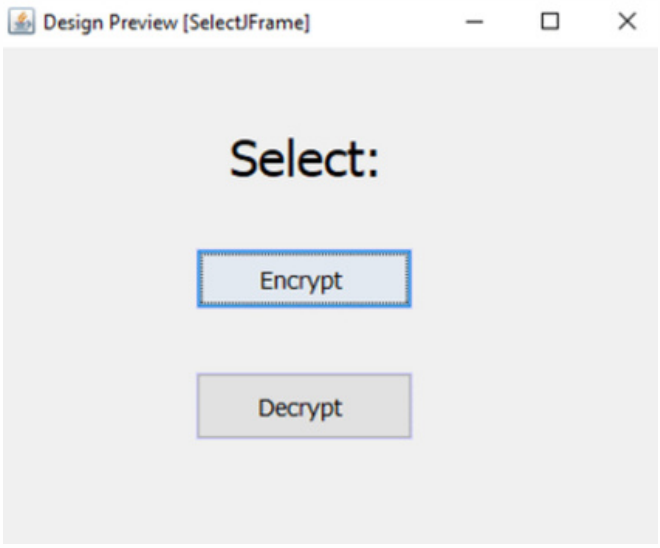
- If a login is selected, it will go directly to the encryption or decryption selection window.



Select window :
When a selection window, there are two options:

Encryption option:
Encryption field that uses the javax.swing.JFrame library and an object of class EncryptJFrame will be pushed.

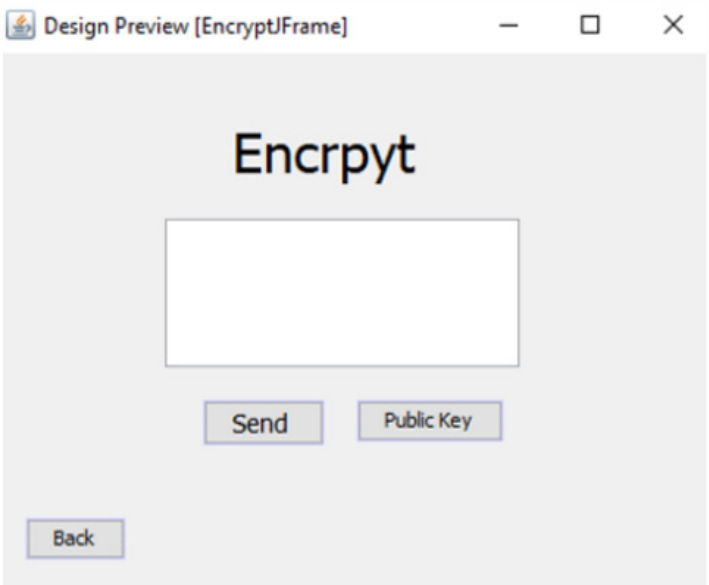
decryption option:
The decryption field that uses the javax.swing.JFrame library and an object of class DecryptJFrame will be pushed.



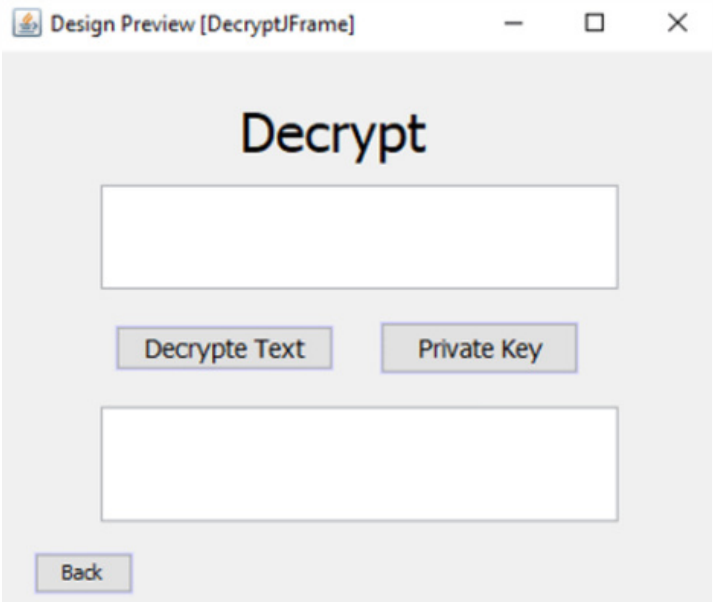
Encrypt window :
If the Encrypt button is selected:
- The plainText will be entered into the text field (jTextField1) of type String

- The public key button must be selected. An encrypted key will be generated to share between the sender and the receiver. The public key button will be selected and read and an AES key generated to generate a secure key. We used Keygenerater, which depends on the correctly classified encrypted random number generator and saved it in the key variable after saving the key. The public key will be converted from String to publickey We used the library javax.swing.filechooser.FileSystemView to save the public key file path in the variable path using the JFileChooser object and then the public key will be converted from String to publickey reading the byte of the file and then encrypting it with X509EncodedKeySpec after that convert byte To a public key using KeyFactory it returns a Key Factory object that converts public/private keys for RSA algorithm and using the generatePublic method. Then the key will be encrypted using RSA algorithm by creating an RSA variable and sending it to the Base64.getEncoder().encodeToString(). (key.getEncoded()), pubkey)Encoder using getEncoder() then get the encoded string by passing the private byte value Enter the actual string in the encodeToString() method to convert it to text. Thus, the public key was selected for the future and an encrypted key was generated using the RSA algorithm.

- After entering the text and choosing the public key, when you press the encryption button, the text will be encrypted and displayed encrypted. At first, an AES variable was created and then sent to method encrypt with the text and the encrypted key, and when you receive it, you convert the text to a byte and then choose encryption mode Then initialize the encryption using toCipher.init with key and parameters then store the IV as a string to share over socket IV and then start encrypting the text and saving it in byte using toCipher. byte to string by sending it to method encode which converts byte to string using Base64.getEncoder().encodeToString() After encoding the text we save the iv and then send it to the receiver for decoding.



Decrypt window :
If decrypt is selected:
This button receives an encrypted text and decrypts it. Initially, an RSA object will be created to decrypt the symmetric key, then store the secret key in a byte string to convert it from String to SecretKey byte. The rsa variable with key and private key is sent to method decrypt (this method decrypts). Encrypting the ciphertext using the private key The key is sent to method encode and the string is converted to a byte using Base64.getDecoder().decode()
Then the encrypted text is converted using Cipher.getInstance(algorithm) It decrypts the text using the private key and then initializes the decryption using toPlain.init with the key and parameters and then starts to decrypt the text and saves it in a byte using toPlain.doFinal it decrypts in the process From one part depending on how this encryption is configured then it is returned byte) Then the key will be re-created using secretkeySpec We used secretkey which is an interface that defines it as a secret (symmetric) key through which the decryption is done, and then start the AES object to decrypt The file using SecretKey by creating an AES variable and then sending it to method decryp with the encrypted text and secretKey and iv when decrypted the original text will be returned as a string after which the decrypted text is displayed in PlainTextField and thus the decryption process is done.



3.0 Code:

LoginJFrame :

```java
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

public class LoginJFrame extends javax.swing.JFrame {
    // store user information
    public static ArrayList <userInfo> users = new ArrayList<>();


    public LoginJFrame() {
        initComponents();

    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jPasswordField1 = new javax.swing.JPasswordField();
        jLabel4 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 30)); // NOI18N
        jLabel1.setText("XSecure System");

        jTextField1.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

        jButton2.setBackground(new java.awt.Color(204, 204, 255));
        jButton2.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        jButton2.setText("Sign in");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                SignInButton(evt);
            }
        });

        jButton3.setBackground(new java.awt.Color(204, 204, 255));
        jButton3.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        jButton3.setText("Sign up");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                try {
                    SignUpButton(evt);
                } catch (InterruptedException ex) {
                    Logger.getLogger(LoginJFrame.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        });

        jLabel2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jLabel2.setText("UserName:");

        jLabel3.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jLabel3.setText("PassWord:");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(51, 51, 51)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jLabel1)
                            .addGroup(layout.createSequentialGroup()
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel2)
                                    .addComponent(jLabel3))
                                .addGap(42, 42, 42)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                    .addComponent(jTextField1)
                                    .addComponent(jPasswordField1, javax.swing.GroupLayout.DEFAULT_SIZE, 142, Short.MAX_VALUE)))))
```

```
                    .addGroup(layout.createSequentialGroup()
                        .addGap(103, 103, 103)
                        .addComponent(jButton2)
                        .addGap(46, 46, 46)
                        .addComponent(jButton3))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(184, 184, 184)
                        .addComponent(jLabel4)))
                .addContainerGap(98, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(38, 38, 38)
                .addComponent(jLabel1)
                .addGap(32, 32, 32)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel2))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE, 33, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(20, 20, 20)
                .addComponent(jLabel4)
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton2)
                    .addComponent(jButton3))
                .addContainerGap(49, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>
    // method to signin users
    private void SignInButton(java.awt.event.ActionEvent evt) {
        try {
        String Pass = jPasswordField1.getText();
        String User = jTextField1.getText();
        // to compare intered password and hash it with user password
        for(userInfo s : users)
        if (User.equals(s.userName)&&s.generateHash(Pass).equals(s.password)){
            SelectJFrame select = new SelectJFrame();
            select.show();
            dispose();

        } else jLabel4.setText("incorrect password or username");

        } catch (Exception ex) {
            System.err.print(ex);
        }
    }
    // method to sign up new user and add user to users array
    private void SignUpButton(java.awt.event.ActionEvent evt) throws InterruptedException {

        String Pass = jPasswordField1.getText();
        String User = jTextField1.getText();
        try {
            userInfo newUser = new userInfo(User,Pass);
            users.add(newUser);
        }
        catch (Exception ex) {
            System.err.print(ex);

        }
        jLabel4.setText("sign up successfuly");
        SelectJFrame select = new SelectJFrame();
        select.show();
        dispose();
    }
    public static void main(String args[]) {



        try {

            userInfo admin = new userInfo("admin","1234");
            userInfo admin2 = new userInfo("admin2","12345");
            users.add(admin);
            users.add(admin2);

            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(LoginJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(LoginJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(LoginJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(LoginJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (Exception ex){
            System.out.println("No User");
        } //</editor-fold>
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new LoginJFrame().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPasswordField jPasswordField1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration
```

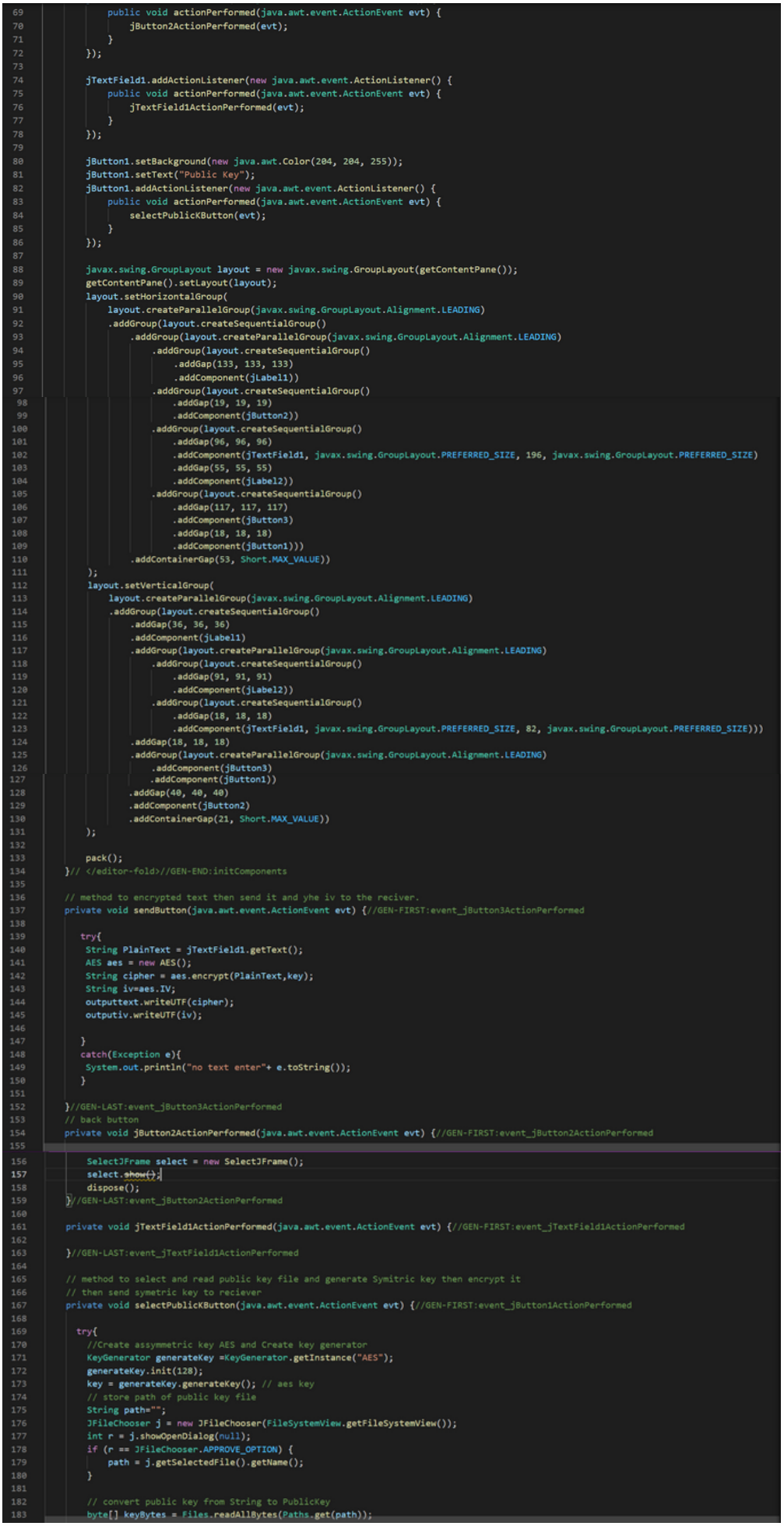
SelectJFrame class :

```
public class SelectJFrame extends javax.swing.JFrame {

    // Creates new form SelectJFrame
    public SelectJFrame() {
        initComponents();
    }
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 30)); // NOI18N
        jLabel1.setText("Select File:");

        jButton1.setBackground(new java.awt.Color(204, 204, 255));
        jButton1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jButton1.setText("Encrypt File");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setBackground(new java.awt.Color(204, 204, 255));
        jButton2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jButton2.setText("Decrypt File");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(111, 111, 111)
                        .addComponent(jLabel1))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(120, 120, 120)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_V
                            .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE, 129, Short.MAX_VALUE))))
                .addContainerGap(143, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(49, 49, 49)
                .addComponent(jLabel1)
                .addGap(34, 34, 34)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 36, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(38, 38, 38)
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(66, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
       //to open select file page
        EncryptJFrame encrypt = new EncryptJFrame();
        encrypt.show();

        //to close current page and open selected page.
        dispose();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        //to open select file page
        DecryptJFrame dncrypt = new DecryptJFrame();
        dncrypt.show();
```

```
        //to close current page and open selected page.
        dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(SelectJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(SelectJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(SelectJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(SelectJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new SelectJFrame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration
}
```

EncryptJFrame class :

```
import java.io.*;
import java.net.*;
import java.net.Socket;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PublicKey;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileSystemView;

public class EncryptJFrame extends javax.swing.JFrame {
    static ServerSocket server;
    static Socket socket;
    static DataOutputStream outputtext;
    static DataOutputStream outputiv;
    static DataOutputStream outputkey;
    static String symetricKey;
    SecretKey key;

    public EncryptJFrame() {
        initComponents();
 }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jLabel3 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        jButton3 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jButton2 = new javax.swing.JButton();
        jTextField1 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();

        jLabel3.setText("jLabel3");

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 30)); // NOI18N
        jLabel1.setText("Encrpyt");

        jButton3.setBackground(new java.awt.Color(204, 204, 255));
        jButton3.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jButton3.setText("Send ");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                sendButton(evt);
            }
        });

        jButton2.setBackground(new java.awt.Color(204, 204, 255));
        jButton2.setText("Back");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
```



```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        jTextField1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        jButton1.setBackground(new java.awt.Color(204, 204, 255));
        jButton1.setText("Public Key");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                selectPublicKButton(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(133, 133, 133)
                        .addComponent(jLabel1))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(19, 19, 19)
                        .addComponent(jButton2))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(96, 96, 96)
                        .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 196, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(55, 55, 55)
                        .addComponent(jLabel2))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(117, 117, 117)
                        .addComponent(jButton3)
                        .addGap(18, 18, 18)
                        .addComponent(jButton1)))
                .addContainerGap(53, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(36, 36, 36)
                .addComponent(jLabel1)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(91, 91, 91)
                        .addComponent(jLabel2))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(18, 18, 18)
                        .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 82, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jButton3)
                    .addComponent(jButton1))
                .addGap(40, 40, 40)
                .addComponent(jButton2)
                .addContainerGap(21, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    // method to encrypted text then send it and yhe iv to the reciver.
    private void sendButton(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton3ActionPerformed

       try{
        String PlainText = jTextField1.getText();
        AES aes = new AES();
        String cipher = aes.encrypt(PlainText,key);
        String iv=aes.IV;
        outputtext.writeUTF(cipher);
        outputiv.writeUTF(iv);

       }
       catch(Exception e){
        System.out.println("no text enter"+ e.toString());
       }

    }//GEN-LAST:event_jButton3ActionPerformed
    // back button
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton2ActionPerformed

        SelectJFrame select = new SelectJFrame();
        select.show();
        dispose();
    }//GEN-LAST:event_jButton2ActionPerformed

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jTextField1ActionPerformed

    }//GEN-LAST:event_jTextField1ActionPerformed

    // method to select and read public key file and generate Symitric key then encrypt it
    // then send symetric key to reciever
    private void selectPublicKButton(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed

      try{
        //Create assymmetric key AES and Create key generator
        KeyGenerator generateKey =KeyGenerator.getInstance("AES");
        generateKey.init(128);
        key = generateKey.generateKey(); // aes key
        // store path of public key file
        String path="";
        JFileChooser j = new JFileChooser(FileSystemView.getFileSystemView());
        int r = j.showOpenDialog(null);
        if (r == JFileChooser.APPROVE_OPTION) {
            path = j.getSelectedFile().getName();
        }

        // convert public key from String to PublicKey
        byte[] keyBytes = Files.readAllBytes(Paths.get(path));
```

```
        X509EncodedKeySpec spec =new X509EncodedKeySpec(keyBytes);
        KeyFactory kf = KeyFactory.getInstance("RSA");
        PublicKey pubkey= kf.generatePublic(spec);
        System.out.println("public key is "+Base64.getEncoder().encodeToString(pubkey.getEncoded()));
        // encrypt symetrc key
        RSA a = new RSA();
        symetricKey=a.encrypt(Base64.getEncoder().encodeToString(key.getEncoded()), pubkey);
        outputkey.writeUTF(symetricKey);
      }
    catch(Exception e){
        System.err.println("no key file"+e.toString());
       }
    }//GEN-LAST:event_jButton1ActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) throws IOException {

        /* Set the Nimbus look and feel */

        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(EncryptJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(EncryptJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(EncryptJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(EncryptJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new EncryptJFrame().setVisible(true);
            }
        });

       try{
            // create socket to send the cipher text
            server = new ServerSocket(5000);
            socket = server.accept();
            outputtext = new DataOutputStream(socket.getOutputStream());
            outputiv = new DataOutputStream(socket.getOutputStream());
            outputkey = new DataOutputStream(socket.getOutputStream());

        }
        catch(Exception e){
        System.err.println("no reciver"+e.toString());
        }



    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration//GEN-END:variables

}
```

DecryptJFrame class :

```java
import java.io.*;
import java.net.Socket;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileSystemView;


public class DecryptJFrame extends javax.swing.JFrame {

    //static ServerSocket server;
    static Socket socket;
    static DataInputStream inputkey;
    static DataInputStream inputtext;
    static DataInputStream inputiv;
    static byte[] iv;
    static String  key;
    static String CipherText;
    SecretKey secretKey ;
    PrivateKey PrivateKey;


    public DecryptJFrame() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        DecryptLable = new javax.swing.JLabel();
        DecryptButton = new javax.swing.JButton();
        ErrorLable = new javax.swing.JLabel();
        BackButton = new javax.swing.JButton();
        CipherTextField = new javax.swing.JTextField();
        PlainTextField = new javax.swing.JTextField();
        prKeyButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        DecryptLable.setFont(new java.awt.Font("Tahoma", 0, 30)); // NOI18N
        DecryptLable.setText("Decrypt");
        DecryptLable.setPreferredSize(new java.awt.Dimension(101, 37));

        DecryptButton.setBackground(new java.awt.Color(204, 204, 255));
        DecryptButton.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        DecryptButton.setText("Decrypte Text");
        DecryptButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                decryptButton(evt);
            }
        });

        ErrorLable.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

        BackButton.setBackground(new java.awt.Color(204, 204, 255));
        BackButton.setText("Back");
        BackButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                BackButton(evt);
            }
        });

        CipherTextField.setToolTipText("                                ");
        CipherTextField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        PlainTextField.setForeground(new java.awt.Color(153, 153, 153));
        PlainTextField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField2ActionPerformed(evt);
            }
        });

        prKeyButton.setBackground(new java.awt.Color(204, 204, 255));
        prKeyButton.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        prKeyButton.setText("Private Key");
```

```
        prKeyButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                selectPtivateKButton(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(134, 134, 134)
                        .addComponent(DecryptLable, javax.swing.GroupLayout.PREFERRED_SIZE, 110, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(278, 278, 278)
                        .addComponent(ErrorLable))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(20, 20, 20)
                        .addComponent(BackButton))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(57, 57, 57)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(CipherTextField)
                            .addComponent(PlainTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 287, Short.MAX_VALUE)))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(65, 65, 65)
                        addComponent(DecryptButton)
                        .addGap(26, 26, 26)
                        .addComponent(prKeyButton, javax.swing.GroupLayout.PREFERRED_SIZE, 110, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(56, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(26, 26, 26)
                .addComponent(DecryptLable, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.Group
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(CipherTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 58, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(DecryptButton)
                    .addComponent(prKeyButton, javax.swing.GroupLayout.PREFERRED_SIZE, 29, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addComponent(PlainTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 64, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(ErrorLable)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(BackButton)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>
// This button recieve cipher text and decrypt it
    private void decryptButton(java.awt.event.ActionEvent evt) {
        try{
        // initiate RSA object to decrypt Symetric Key
        RSA rsa= new RSA();
        //Store secret key in byte String to conver it from String To SecretKey
        byte[] decodedKey = Base64.getDecoder().decode(rsa.decrypt(key, PrivateKey));
        // Rebuild key using SecretKeySpec
        secretKey = new SecretKeySpec(decodedKey, 0, decodedKey.length, "AES");
        System.out.println("Symetric key afte decrypt  " +Base64.getEncoder().encodeToString(secretKey.getEncoded()));

        // initiate AES object to decrypt file using SecretKey
        AES aes = new AES();
        String plain =aes.decrypt(CipherText, secretKey,iv);
        // present decrypted text in PlainTextField
        PlainTextField.setText(PlainTextField.getText().trim()+"\n"+plain);
        }
        catch(Exception error){
        System.err.println("Error! No Key OR Text "+ error.toString());
        }

    }


    // method to go to main page
    private void BackButton(java.awt.event.ActionEvent evt) {
        SelectJFrame select = new SelectJFrame();
        select.show();

        dispose();
    }

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
        // this text filed well recive the cipher text from the sender.

    }

    private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
        // present plain text after decryption

    }

    // method to selectPtivateKButton and decrypt the symetricKey
    private void selectPtivateKButton(java.awt.event.ActionEvent evt) {
        try {
        // to select private key file and read it as String
        String path="";
        JFileChooser j = new JFileChooser(FileSystemView.getFileSystemView());
        int r = j.showOpenDialog(null);
        if (r == JFileChooser.APPROVE_OPTION) {
            path = j.getSelectedFile().getName();
            //Store private key in byte String to convert it from String To PrivateKey
            byte[] keyBytes2 = Files.readAllBytes(Paths.get(path));
            PKCS8EncodedKeySpec spec2 =new PKCS8EncodedKeySpec(keyBytes2);
            KeyFactory kf2 = KeyFactory.getInstance("RSA");
            PrivateKey= kf2.generatePrivate(spec2);
            System.out.println("private key "+Base64.getEncoder().encodeToString(PrivateKey.getEncoded()));
        }

        }
```

```
        catch(Exception ex){
            System.err.println("No Private  File"+ex.toString());
            }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {

        /* Set the Nimbus look and feel */

        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(DecryptJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(DecryptJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(DecryptJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(DecryptJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new DecryptJFrame().setVisible(true);
            }
        });


        try{
            // to establish connection
            socket = new Socket("localhost", 5000);
            // object to read from socket
            inputkey = new DataInputStream(socket.getInputStream());
            inputtext = new DataInputStream(socket.getInputStream());
            inputiv = new DataInputStream(socket.getInputStream());
            // read key and use it in decryptButton method
            key=inputkey.readUTF();
            // read text and display it in cipher text field
            CipherText= inputtext.readUTF();
            // read IV as string then convert it as byte [] and use it in decryptButton method
            iv= Base64.getDecoder().decode(inputiv.readUTF());
            // display cipher text
            CipherTextField.setText(CipherTextField.getText().trim()+"\n"+CipherText);


        }
        catch(Exception e){
        System.err.println("no text recive"+e.toString());
       }

    }
    // Variables declaration - do not modify
    private javax.swing.JButton prKeyButton;
    private javax.swing.JButton BackButton;
    private static javax.swing.JButton DecryptButton;
    private javax.swing.JLabel DecryptLable;
    private javax.swing.JLabel ErrorLable;
    static javax.swing.JTextField CipherTextField;
    private javax.swing.JTextField PlainTextField;
    // End of variables declaration
}
```

RSA class :

```java
import java.security.*;
import java.util.Base64;
import javax.crypto.Cipher;
public class RSA {
    // method to encrypt massage by public key
    public String encrypt ( String text , PublicKey pub) throws Exception {
        byte [] TextByte =text.getBytes();
        Cipher toCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        toCipher.init(Cipher.ENCRYPT_MODE, pub);
        byte [] cipherByte = toCipher.doFinal(TextByte);
        return encode(cipherByte);
    }
    // method to decrypt cipher by private key
    public String decrypt (String m , PrivateKey pr )throws Exception{
        byte [] cipherByte = decode(m);
        Cipher toPlain = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        toPlain.init(Cipher.DECRYPT_MODE,pr);
        byte [] TextByte = toPlain.doFinal(cipherByte);
        return new String(TextByte, "UTF8");
    }
    // method to convert byte[] to String
    private String encode (byte [] s){
        return Base64.getEncoder().encodeToString(s);
    }
    // method to convert String to Byte[]
    private byte [] decode (String s){
        return Base64.getDecoder().decode(s);
    }
}
```

AES class :

```java
import java.util.Base64;
import javax.crypto.*;
import javax.crypto.spec.GCMParameterSpec;
public class AES {
    Cipher toCipher;
    String IV;
    //this method to encrypt the plain text for the sender.
    public String encrypt (String Text, SecretKey key) throws Exception{
        // convert text to byte[]
        byte [] TextByte =Text.getBytes();
        // to select encryption mode
        toCipher = Cipher.getInstance("AES/GCM/NoPadding");
        // Initialize cipher with secret key passed from parameter
        toCipher.init(Cipher.ENCRYPT_MODE, key);
        // store IV as String to share it through socket
        IV=Base64.getEncoder().encodeToString(toCipher.getIV());
        // start encryption
        byte [] cipherByte = toCipher.doFinal(TextByte);
        // convert cipher text to String
        return encode(cipherByte);
    }
        //this method to decrypt the chipher text for the reciver.
    public String decrypt (String cipherText ,SecretKey key, byte[] iv)throws Exception{
        // convert text to byte[]
        byte [] cipherByte = decode(cipherText);
        // to select decryption mode
        Cipher toPlain = Cipher.getInstance("AES/GCM/NoPadding");
        GCMParameterSpec spec = new GCMParameterSpec(128,iv);
        toPlain.init(Cipher.DECRYPT_MODE, key,spec);
        byte [] TextByte = toPlain.doFinal(cipherByte);
        return new String(TextByte);
    }
    // method to convert byte[] to String
    private String encode (byte [] s){
        return Base64.getEncoder().encodeToString(s);
    }
    // method to convert String to Byte[]
    private byte [] decode (String s){
        return Base64.getDecoder().decode(s);
    }
}

```

userInfo class :

```java
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.security.NoSuchAlgorithmException;
import java.security.*;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;

public class userInfo {
    String userName ;
    String password ;
    private PublicKey pubKey ;
    private PrivateKey prKey;
    // to create instance from the class
    public userInfo(String userName, String password) throws Exception {
        this.userName = userName;
        // send password to save hashed password
        this.password = generateHash(password);
        // generate private and public keys
        KeyPairGenerator kgenerator = KeyPairGenerator.getInstance("RSA");
        kgenerator.initialize(1024);
        KeyPair pair=kgenerator.generateKeyPair();
        prKey=pair.getPrivate();
        pubKey=pair.getPublic();
        // Store Public Key.
        X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(pubKey.getEncoded());
        FileOutputStream keysFile = new FileOutputStream(this.userName + "public.key");
        keysFile.write(x509EncodedKeySpec.getEncoded());
        keysFile.close();
        // Store Private Key.
        PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(prKey.getEncoded());
        keysFile = new FileOutputStream(this.userName + "private.key");
        keysFile.write(pkcs8EncodedKeySpec.getEncoded());
        keysFile.close();
    }
    public String getUserName() {
        return userName;
    }
    public String getPassword() {
        return password;
    }
    public PublicKey getPubKey() {
        return pubKey;
    }
    public PrivateKey getPrKey() {
        return prKey;
    }
    // method to hash the password using SHA-512
    public String generateHash(String password_text) throws Exception {
    MessageDigest md = MessageDigest.getInstance ("SHA-512") ;
    byte[] hashedPassword = md.digest(password_text.getBytes(StandardCharsets.UTF_8));
    // convert hashed password to String
    return Base64.getEncoder().encodeToString(hashedPassword);
    }
}
```

4.0 Challenges:

1. One of the challenges that we faced, we did not know how to make the user send a file to the other user ‘receiver’, so we found an alternative solution Make the user send a text massage.

2. We could not be able to read the cipher ,symmetric key and IV separately. So, we create input stream objects for each one.

3. We had also a problem with method decryption in the AES class that didn't work .To solve this problem, we saved the IV in a variable and sent it to the receiver.

4. To send the key trough the socket it has to be converted to string. So, we face a problem to return it back as a SecretKey. Also reading private and public key from a file will return a String , so we had to convert the to PublicKey or PrivateKey.

5.0 References:

1. https://www.youtube.com/watch?v=kqBmsLvWU14
2. https://www.youtube.com/watch?v=8wlE6DgOWBs
3. https://youtu.be/R9eerqP78PE
4. https://youtu.be/J1RmZZEkN0k
5. https://stackoverflow.com/questions/5355466/converting-secret-key-into-a-string-and-vice-versa
6. https://snipplr.com/view/18368/saveload--private-and-public-key-tofrom-a-file
7. https://www.sneppets.com/java/util/java-convert-string-to-rsa-public-key/
8. https://www.geeksforgeeks.org/sha-512-hash-in-java/