RSA-based Public Key Crypto System Chatting Program

14146325 ITM Hongbeom Choi

Contents

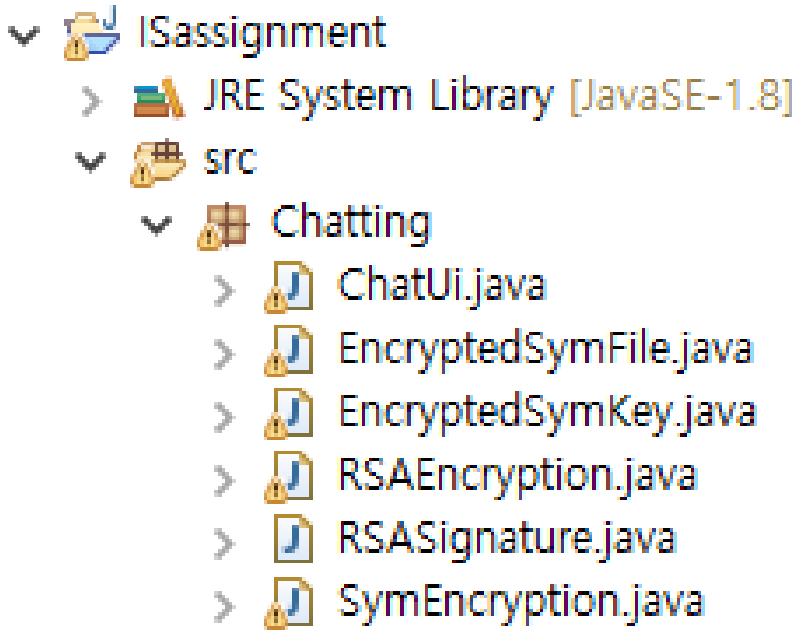
Source Code

01

02

My program consists of six classes

- ChatUi class - contains main method
- EncryptedSymFile
- EncryptedSymKey
- RSAEncryption
- RSASignature
- SymEncryption



Source code – ChatUi class

```java
package Chatting;

import java.awt.BorderLayout;

public class ChatUi extends JFrame {

    private JPanel contentPane;
    private JTextField Chattextfield;
    private JTextField Idtextfield;
    private JTextField IPtextfield;
    private JTextField Porttextfield;
    private String Id;
    private String IPAddress;
    private int PortNumber;
    private Sender sender;
    private StyledDocument Chattingdoc;
    private StyledDocument keyInfodoc;
    private StyledDocument Filetransferdoc;
    private Receiver receiver;
    private String mes;
    private byte[] inputobj = null;
    private byte[] message = null;
    private byte[] receivedFile = null;
    private RSAEncryption RSA;
    private RSASignature RSASig;
    private PublicKey mypublicKey;
    private PublicKey clientpublicKey;
    private PrivateKey privateKey;
    private KeyPair keyPair;
    private Key symKey;
    private Object receiveobj= null;
    private File file = null;
    private FileOutputStream stream = null;
    private String fileName = null;
    private String filePath = null;
    ServerSocket serverSocket = null;
    private EncryptedSymKey ensobj = null;
    private SymEncryption sym = null;
    private SecretKey sck = null;
    Socket socket = null;
    private EncryptedSymKey keyobj;
    private JTextField filepathtextfield = null;
    private JTextField filenametextfield = null;
    private JButton btnSetFileOption = null;

    /**
     *
     * Launch the application.
     */
    public static void main(String[] args) {        // main method
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    ChatUi frame = new ChatUi();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
```

Source code – ChatUi class

```
                }
            }
        });
    }


    // Sender class - The role of sending objects to receiver class such as byte[], Public Key, EncrpytedSymKey, EncryptedSymFile
    public class Sender{
        ObjectOutputStream obs = null;      // I use the ObjectOutputStream object in sending objects to ObjectInputStream.

        public Sender(Socket socket){       // When you create an object, you need a socket object that initializes the obs(ObjectOutputStream) in the sender class member variable.
            try {
                obs = new ObjectOutputStream(socket.getOutputStream());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }


        public void sendmessage(byte[] messa) { // method of sending byte[] of message. messa is that byte[] comes from message.getEncoded()
            try{
                    byte[] encryptedmessa = RSA.RSAencryption(messa, clientpublicKey);  // Encrypt messa using clientpublicKey. encryptedemessa is encrypted byte[] of message.
                    obs.writeObject(encryptedmessa);                                    // write encryptedmessa to obs
                    obs.flush();                                                        // send encryptedmessa to Opponent ObjectInputStream.

            }catch(IOException e){
                System.out.println("ServerSender Error");
                e.printStackTrace();
            }

        }

        public void sendPublicKey(PublicKey key) { // method of seneding my public key to opponent. key comes from method of RSAKeygeneration.
            try {
                obs.writeObject(key);               // Write my public key(key object) to obs
                obs.flush();                        // send key to opponent

            } catch (IOException e) {
                e.printStackTrace();
            }

        }

        public void sendFile(byte[] file) {     // method of sending encrypted file using symmetric Key to opponent
            try {
                EncryptedSymFile ensFile = new EncryptedSymFile(sym.AESkeyEncrpytion(file, symKey));    //create a object of EncryptedSymFile with initializing byte[] variable of the object.
                //sym.AESkeyEncryption method carrying out encryption file using symKey(Symmetric Key). it returns the encrypted byte[] of file.
                obs.writeObject(ensFile);       // write ensFile object to obs
                obs.flush();                    // send ensFile to opponent
            } catch (IOException e) {
                e.printStackTrace();
            }


        }

```

Source code – ChatUi class

```java
        public void sendFileSignature(byte[] signaturebyte) {   // method of sending signature of my file to opponent. signaturebyte is that signature of transferred file.
            try {
                obs.writeObject(signaturebyte);                 // write signature of transferred file to obs
                obs.flush();                                    // send signautre to opponent
            } catch (IOException e) {
                e.printStackTrace();
            }

        }

        public void sendSymKey(Key key) {                   // method of sending Symmetric key(key) to opponent. key is the symmetric key using at encrypt file to transfer.
            ensobj = new EncryptedSymKey(RSA.RSAencryption(key.getEncoded(), clientpublicKey));     // create a object of EncryptedSymKey with initializing byte[] variable of the object.
            // RSA.RSAencryption(key.getEncoded(), clientpublicKey) means that encrypt symmetric key using opponent public key. it returns encrypted byte[] of symmetric key.
            try {
                obs.writeObject(ensobj);                    // write ensobj to obs
                obs.flush();                                // send ensobj to opponent's ObjectInputStream
            } catch (IOException e) {
                e.printStackTrace();
            }


        }


    }

    // Receiver class - The role of receiving objects from sender class such as byte[], Public Key, EncrpytedSymKey, EncryptedSymFile
    public class Receiver extends Thread{
        private Socket socket;
        ObjectInputStream ois = null;       // I use the ObjectInputStream object in receiving objects from ObjectOutputStream
        public Receiver(Socket socket){     // When you create an object, you need a socket object that initializes the ois(ObjectOutputStream) in the receiver class member variable.
            this.socket=socket;
        }

        public Receiver(){}
        String abc=null;
        String clientpbk = null;
        String def=null;
        @Override
        public void run() {     // method of receiving. receiver class extends Thread class because receiver should always read obejects in ObjectInputstream from opponent's objectOutputStream.

            try{
            ois = new ObjectInputStream(socket.getInputStream());

            while(true){
                try {   // 1st case that object from opponent sender is object of PublicKey
                    receiveobj = ois.readObject();                      // read object in ois. it returns object of type "Object"
                    clientpublicKey = (PublicKey)receiveobj;            // casting receiveobj to PublicKey. if error occurs(receiveobj is not the opponet's public key from sender), go to catch clause.

                    byte[] clientpubk = clientpublicKey.getEncoded();                                                   //These codes carry out
                    clientpbk = "";                                                                                     //appearing opponent's public
                    for(byte b: clientpubk) clientpbk = clientpbk+Integer.toString((b & 0xff)+0x100, 16).substring(1)+" ";  //key information in chatting UI
                    keyInfodoc.insertString(keyInfodoc.getLength(), "\nOpponent Public Key: \n"+ clientpbk, keyInfodoc.getStyle("black"));  //
                    keyInfodoc.insertString(keyInfodoc.getLength(), "\nOpponent Public Key Length : "+clientpubk.length+ " byte" , keyInfodoc.getStyle("black"));  //

```

Source code – ChatUi class

```
229             }catch(Exception e) {
230                 e.printStackTrace();
231                 System.out.println(e);
232                 try {   // 2st case that object from opponent sender is object of EncrpytedSymKey
233                     keyobj = (EncryptedSymKey)receiveobj;           // casting receiveobj to EncryptedSymkey. if error occurs(receiveobj is not the symmetric key from sender), go to catch clause.
234
235                     byte[] receivedKeyobj = keyobj.getEncrpytedSymKey();                                    //Theses codes carry out saving symmetric key from sender into class variable sck.
236                     byte[] decryptedSymKeyByte = RSA.RSAdecryption(receivedKeyobj, privateKey);             //
237                     sck = new SecretKeySpec(decryptedSymKeyByte, 0, decryptedSymKeyByte.length, "AES");     //
238
239
240
241
242                 } catch (Exception e1) {
243                     System.out.println(e1);
244                     e1.printStackTrace();
245                     try {   //3rd case that object from opponent sender is byte[] of encrypted message
246                         inputobj = (byte[]) receiveobj;             // casting receiveobj to byte[]. if error occurs(receiveobj is not byte[] of encrypted message from sender), go to catch clause.
247                         byte[] decryptedByte = RSA.RSAdecryption(inputobj, privateKey); // decrypt inputobj(byte[] of encrypted message) using private key.
248                         def = new String(inputobj, 0, inputobj.length, "utf-8");                        //These codes carry out appearing encrypted message and decrpyted message
249                         abc = new String(decryptedByte, 0, RSA.RSAdecryption(inputobj, privateKey).length, "utf-8");   //in chatting UI.
250                         Chattingdoc.insertString(Chattingdoc.getLength(), "\n dec: "+def, Chattingdoc.getStyle("black"));   //
251                         Chattingdoc.insertString(Chattingdoc.getLength(), "\n enc:"+abc, Chattingdoc.getStyle("black"));    //
252                         inputobj = null;
253
254                     }catch(Exception e2) {
255                         System.out.println(e2);
256                         e2.printStackTrace();
257
258                         try { // 4th case that object from opponent sender is file signature.
259                                 RSASignature verify = new RSASignature();
260                                 if(verify.verifyRSASignature(receivedFile, (byte[]) receiveobj, clientpublicKey)) {     // This case is that the file has not been corrupted of changed,
261                                     Filetransferdoc.insertString(Filetransferdoc.getLength(),                       // because verify.verifyRSASignature(receivedFile, (byte[]) receiveobj, clientpublicKey)
262                                             "\nFile Verification: True \nFile has not been corrupted or changed!!"  ,   // return true. And appears true message in chatting UI
263                                             Filetransferdoc.getStyle("black"));
264
265                                 }else if(verify.verifyRSASignature(receivedFile, (byte[]) receiveobj, clientpublicKey)){// This case is that the file has been corrupted of changed,
266                                     Filetransferdoc.insertString(Filetransferdoc.getLength(),                       // because verify.verifyRSASignature(receivedFile, (byte[]) receiveobj, clientpublicKey)
267                                             "\nFile Verification: False \nFile has been corrpted or changed."  ,        // return false. And appears false message in chatting UI
268                                             Filetransferdoc.getStyle("black"));
269                                 }
270
271                                 receivedFile=null;      // It is important to set null in receivedFile because receivedFile is the class member variable.
272
273
274
275                         } catch(Exception e3){
276                             System.out.println(e3);
277                             e3.printStackTrace();
278                             try {   // 5th case that object from opponent sender is object of EncryptedSymFile.
279                                 SymEncryption dec = new SymEncryption();
280                                 EncryptedSymFile recensFileobj = (EncryptedSymFile) receiveobj; // casting receiveobj to EncryptedSymFile.
281                                                                                                 // if error occurs(receiveobj is not file encrypted by symmetric key from sender), go to catch clause.
282                                 byte[] encFile = recensFileobj.getEncryptedFile();              // byte[] of encrypted file
283                                 byte[] decFile = dec.AESdecryption(encFile, sck);               // decrypt byte[] of encrypted file using AESdecrpytion method. sck is symmetric key received from opponent.
284
285                                 if(filePath==null) {        // case that user does not set the file path to receive
```

Source code – ChatUi class

```
286                     stream = new FileOutputStream(new File("C:\\Users\\idd74\\Desktop\\Encrypted_Recevied_File"));
287                     // Initialize member variable 'stream' with default path(C:\\Users\\idd74\\Desktop\\Encrypted_Recevied_File)
288                     stream.write(encFile);                                                          // create encrypted file to default path
289                     Filetransferdoc.insertString(Filetransferdoc.getLength(),
290                             "\n****** File received !!******\nEncrypted File Name: Encrypted_Recevied_File\nEncrypted_File Path:"   // These codes carry out appearing encrypted file
291                             + " C:\\Users\\idd74\\Desktop\nEncrypted File Size: "+encFile.length +"bytes",                          // information in chatting UI
292                             Filetransferdoc.getStyle("black"));                                                                    //
293
294                     byte[] secretKeybb = sck.getEncoded();                                          // These codes carry out appearing symmetric key
295                     String secretKeystring = "";                                                    // information in chatting UI
296                     for(byte b: secretKeybb) secretKeystring = secretKeystring+Integer.toString((b & 0xff)+0x100, 16).substring(1)+" "; //
297                     Filetransferdoc.insertString(Filetransferdoc.getLength(),                       //
298                             "\nSymmetric Key: "+secretKeystring +"Symmetric Key Size: " + secretKeybb.length +"byte",              //
299                             Filetransferdoc.getStyle("black"));
300                 }
301                 else if(filePath != null) { // case that user set the file path to receive before file transfer
302                     stream = new FileOutputStream(new File(filePath+"\\"+"Encrypted_"+fileName));
303                     // Initialize member variable 'stream' with specified path(filePath)
304                     stream.write(encFile);                                                          // create encrypted file to specified path
305
306                     Filetransferdoc.insertString(Filetransferdoc.getLength(),                       // These codes carry out appearing encrypted file
307                             "\n****** File received!! ******\nEncrypted File Name: " +"Encrypted_"+fileName+"\nEncrypted File Path: "   // information in chatting UI
308                                     + filePath+"\nEncrypted File Size: "+encFile.length +"bytes",                                  //
309                             Filetransferdoc.getStyle("black"));                                                                    //
310
311                     byte[] secretKeybb = sck.getEncoded();                                          // These codes carry out appearing symmetric key
312                     String secretKeystring = "";                                                    // information in chatting UI
313                     for(byte b: secretKeybb) secretKeystring = secretKeystring+Integer.toString((b & 0xff)+0x100, 16).substring(1)+" "; //
314                     Filetransferdoc.insertString(Filetransferdoc.getLength(),                       //
315                             "\nSymmetric Key: "+secretKeystring +"\nSymmetric Key Size: " + secretKeybb.length +"byte",            //
316                             Filetransferdoc.getStyle("black"));                                                                    //
317                 }
318
319
320             if(filePath==null) {    // case that user does not set the file path to receive
321                 receivedFile = decFile;                                                             // save byte[] of decrpyted file to receivedFile
322                 stream = new FileOutputStream(new File("C:\\Users\\idd74\\Desktop\\Recevied_File"));
323                 // Initialize member variable 'stream' with default path(C:\\Users\\idd74\\Desktop\\Recevied_File)
324                 stream.write(decFile);                                                              // create decrypted file to default path
325                 filepathtextfield.setText(null);            // These codes carry out controlling UI's contents
326                 filenametextfield.setName(null);            //
327                 filepathtextfield.setEnabled(true);         //
328                 filenametextfield.setEnabled(true);         //
329                 btnSetFileOption.setEnabled(true);          //
330                 Filetransferdoc.insertString(Filetransferdoc.getLength(),                           // These codes carry out appearing decrypted file
331                         "\nDecrypted_File Name: Recevied_File\nDecrypted File Path:"                // information in chatting UI
332                         + " C:\\Users\\idd74\\Desktop" + "\nDecrypted File Size: "+decFile.length +"bytes" + "\n******************",   //
333                         Filetransferdoc.getStyle("black"));                                         //
334             }
335             else if(filePath != null) { // case that user set the file path to receive before file transfer
336                 receivedFile = decFile;                                                             // save byte[] of decrpyted file to receivedFile
337                 stream = new FileOutputStream(new File(filePath+"\\"+fileName));
338                 // Initialize member variable 'stream' with user specified path(filePath)
339                 stream.write(decFile);                                                              // create decrypted file to specified path(filePath)
340                 filepathtextfield.setText(null);            // These codes carry out controlling UI's contents
341                 filenametextfield.setText(null);            //
```

Source code – ChatUi class

```
342                                 filepathtextfield.setEnabled(true);         //
343                                 filenametextfield.setEnabled(true);         //
344                                 btnSetFileOption.setEnabled(true);          //
345                                 Filetransferdoc.insertString(Filetransferdoc.getLength(),                                    // These codes carry out appearing decrypted file
346                                         "\nDecrypted_File Name: " +fileName+"\nDecrypted_File Path: "                        // information in chatting UI
347                                                 + filePath+"\nFile Size: "+decFile.length +"bytes" + "\n******************",  //
348                                         Filetransferdoc.getStyle("black"));
349                                 filePath = null;                            // set filePath and fileName null.
350                                 fileName = null;                            // for next file.
351                             }
352
353
354                         }catch(Exception e4) {
355                             System.out.println(e4);
356                             e4.printStackTrace();
357                         }finally {
358                             stream.close();     // close filestream
359                         }
360                             }
361
362
363
364                     }
365                 }
366
367             }
368
369
370
371
372
373             }
374         }catch(NullPointerException | SocketException e){
375             System.out.println("Chatting ended");
376         }catch(IOException e){
377             System.out.println("ServerReader Error");
378             e.printStackTrace();
379         }
380     }
381
382 }
383
384
385
386     /**
387      * Create the frame.
388      */
389     public ChatUi() {
390         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
391         setBounds(100, 100, 597, 1017);
392         contentPane = new JPanel();
393         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
394         setContentPane(contentPane);
395         contentPane.setLayout(null);
396
397         JCheckBox chckbxClient = new JCheckBox("Client");           // client check box
```

Source code – ChatUi class

```java
398         chckbxClient.setBounds(12, 47, 71, 23);
399         contentPane.add(chckbxClient);
400
401         JCheckBox chckbxServer = new JCheckBox("Server");           // server check box
402         chckbxServer.setBounds(87, 47, 71, 23);
403         contentPane.add(chckbxServer);
404
405
406         JLabel lblMode = new JLabel("Mode");
407         lblMode.setBounds(12, 10, 126, 20);
408         contentPane.add(lblMode);
409
410         JLabel lblNewLabel = new JLabel("Communication Info");
411         lblNewLabel.setBounds(10, 141, 126, 15);
412         contentPane.add(lblNewLabel);
413
414         JLabel lblKeyInfo = new JLabel("Key Info");                 // Key info area
415         lblKeyInfo.setBounds(12, 239, 57, 15);
416         contentPane.add(lblKeyInfo);
417
418         JButton btnkeygeneration = new JButton("Key generation");   // Key generation button
419         btnkeygeneration.setBounds(103, 414, 172, 23);
420         contentPane.add(btnkeygeneration);
421
422         JButton sendPublicKey = new JButton("Send public key");     // Send Public Key button
423         sendPublicKey.setBounds(326, 414, 172, 23);
424         contentPane.add(sendPublicKey);
425
426         JLabel lblChatting = new JLabel("Chatting");
427         lblChatting.setBounds(12, 447, 57, 15);
428         contentPane.add(lblChatting);
429
430         JButton btnSend = new JButton("Send");                      // Send button
431         btnSend.setBounds(475, 680, 97, 23);
432         contentPane.add(btnSend);
433
434         Chattextfield = new JTextField();                           // Chatting text field
435         Chattextfield.setBounds(12, 680, 451, 23);
436         contentPane.add(Chattextfield);
437         Chattextfield.setColumns(10);
438
439         JLabel lblFileTransfer = new JLabel("File transfer");
440         lblFileTransfer.setBounds(12, 738, 71, 15);
441         contentPane.add(lblFileTransfer);
442
443         JButton sendFile = new JButton("Send file");                // Send File button
444         sendFile.setBounds(475, 915, 97, 23);
445         contentPane.add(sendFile);
446
447         JButton btnFindfile = new JButton("Find file");             // Find File button
448         btnFindfile.setBounds(475, 870, 97, 23);
449         contentPane.add(btnFindfile);
450
451         Idtextfield = new JTextField();                             // ID text field
452         Idtextfield.setBounds(266, 34, 166, 21);
453         contentPane.add(Idtextfield);
454         Idtextfield.setColumns(10);
```

```java
455
456         JButton btnConnect = new JButton("Connect");                // Connect button
457         btnConnect.setBounds(444, 34, 97, 114);
458         contentPane.add(btnConnect);
459
460         JLabel lblId = new JLabel("ID");
461         lblId.setBounds(266, 10, 97, 15);
462         contentPane.add(lblId);
463
464         IPtextfield = new JTextField();                             // IP Address text field
465         IPtextfield.setBounds(266, 80, 166, 21);
466         contentPane.add(IPtextfield);
467         IPtextfield.setColumns(10);
468
469         JLabel lblIpAddress = new JLabel("IP Address");
470         lblIpAddress.setBounds(266, 58, 97, 15);
471         contentPane.add(lblIpAddress);
472
473         JLabel lblPortNumber = new JLabel("Port Number");
474         lblPortNumber.setBounds(266, 106, 97, 15);
475         contentPane.add(lblPortNumber);
476
477         Porttextfield = new JTextField();                           // Port Number text field
478         Porttextfield.setColumns(10);
479         Porttextfield.setBounds(266, 131, 166, 21);
480         contentPane.add(Porttextfield);
481
482         JScrollPane commInfoScroll = new JScrollPane();
483         commInfoScroll.setBounds(12, 166, 545, 63);
484         contentPane.add(commInfoScroll);
485
486         JTextPane CommInfo = new JTextPane();                       // Communication Info area
487         commInfoScroll.setViewportView(CommInfo);
488         StyledDocument CommInfodoc = CommInfo.getStyledDocument();
489         Style def = StyleContext.getDefaultStyleContext().getStyle(StyleContext.DEFAULT_STYLE);
490         Style s = CommInfodoc.addStyle("black", def);
491         StyleConstants.setForeground(s,Color.black);
492
493         JScrollPane FiletransferScroll = new JScrollPane();
494         FiletransferScroll.setBounds(12, 848, 451, 120);
495         contentPane.add(FiletransferScroll);
496
497         JTextPane Filetransfer = new JTextPane();                   // File transfer area
498         FiletransferScroll.setViewportView(Filetransfer);
499         Filetransferdoc = Filetransfer.getStyledDocument();
500         Style fsd = Filetransferdoc.addStyle("black", def);
501         StyleConstants.setForeground(fsd, Color.black);
502
503         JScrollPane ChattiingScroll = new JScrollPane();
504         ChattiingScroll.setBounds(6, 470, 558, 200);
505         contentPane.add(ChattiingScroll);
506
507         JTextPane Chatting = new JTextPane();                       // Chatting area
508         ChattiingScroll.setViewportView(Chatting);
509         Chattingdoc = Chatting.getStyledDocument();
510         Style cs = Chattingdoc.addStyle("black", def);
```

Source code – ChatUi class

```
511         StyleConstants.setForeground(cs,Color.black);
512
513         JScrollPane keyinfoscrollPane = new JScrollPane();
514         keyinfoscrollPane.setBounds(12, 264, 556, 144);
515         contentPane.add(keyinfoscrollPane);
516
517         JTextPane keyInfo = new JTextPane();                        // Key Info area
518         keyinfoscrollPane.setViewportView(keyInfo);
519
520         JLabel lblFilePathTo = new JLabel("File Path to Receive");
521         lblFilePathTo.setBounds(12, 763, 136, 15);
522         contentPane.add(lblFilePathTo);
523
524         JLabel lblFileNameTo = new JLabel("File Name to Receive");
525         lblFileNameTo.setBounds(12, 801, 132, 15);
526         contentPane.add(lblFileNameTo);
527
528         filepathtextfield = new JTextField();                       // File Path text field
529         filepathtextfield.setBounds(151, 760, 254, 21);
530         contentPane.add(filepathtextfield);
531         filepathtextfield.setColumns(10);
532
533         filenametextfield = new JTextField();                       // File Name text field
534         filenametextfield.setBounds(151, 798, 254, 21);
535         contentPane.add(filenametextfield);
536         filenametextfield.setColumns(10);
537
538         btnSetFileOption = new JButton("Set File Option");          // Set File Option button
539         btnSetFileOption.setBounds(427, 759, 137, 63);
540         contentPane.add(btnSetFileOption);
541         keyInfodoc = keyInfo.getStyledDocument();
542         Style ks = keyInfodoc.addStyle("black", def);
543         StyleConstants.setForeground(ks, Color.black);
544
545
546
547         //action listener
548         chckbxClient.addActionListener(new ActionListener() {       // Client check box action listener
549             public void actionPerformed(ActionEvent e) {
550                 if(chckbxClient.isSelected()) {
551                     chckbxServer.setEnabled(false);
552
553
554                 }
555                 else {
556                     chckbxServer.setEnabled(true);
557
558                 }
559
560             }
561         });
562
563
564         chckbxServer.addActionListener(new ActionListener() {       // Server check box action listener
565             public void actionPerformed(ActionEvent e) {
566                 if(chckbxServer.isSelected()) {
```

Source code – ChatUi class

```
                chckbxClient.setEnabled(false);
                lblIpAddress.setEnabled(false);
                IPtextfield.setEnabled(false);
            }
            else {
                chckbxClient.setEnabled(true);
                lblIpAddress.setEnabled(true);
                IPtextfield.setEnabled(true);
                }
        }
    });


    btnConnect.addActionListener(new ActionListener() {         // Connect button action listener
        public void actionPerformed(ActionEvent e) {
            IPAddress = IPtextfield.getText();
            PortNumber = Integer.parseInt(Porttextfield.getText());
            Id=Idtextfield.getText();
            if(chckbxServer.isSelected()){                      // Connection in Server
            try{
                serverSocket = new ServerSocket();                                                      // Initialize serverSocket
                serverSocket.bind(new InetSocketAddress(PortNumber));                                   // Binding port number (comes from user) to serverSocket
                socket = serverSocket.accept();                                                         // connect with client

                CommInfodoc.insertString(CommInfodoc.getLength(), "Connected!", CommInfodoc.getStyle("black"));   // These codes carry out
                CommInfodoc.insertString(CommInfodoc.getLength(), "\n"+"Client IP Address: "            // appearing connection information
                +socket.getInetAddress().toString(), CommInfodoc.getStyle("black"));                    // in chatting UI

                Receiver receiver = new Receiver(socket);                                               // create object of Receiver class
                receiver.start();                                                                       // running thread method of Receiver class


                sender = new Sender(socket);                                                            // create object of Sender class


            }catch(IOException e1){
                System.out.println("Server Error occured");
                e1.printStackTrace();
            } catch (BadLocationException e1) {
                e1.printStackTrace();
            }finally{
                try{
                    if(serverSocket!=null) serverSocket.close();
                }catch(IOException e1){
                    System.out.println("");
                    e1.printStackTrace();
                }
            }
            }
            else if(chckbxClient.isSelected()){     // Connection in Client

                try{
                    socket =new Socket();                                                               // initialize socket.
                    socket.connect(new InetSocketAddress(IPAddress, PortNumber));                       // connect with server using IP address and port number.
                    CommInfodoc.insertString(CommInfodoc.getLength(), "Connected!", CommInfodoc.getStyle("black"));   // appearing connection information in chatting UI

```

Source code – ChatUi class

```
                    sender = new Sender(socket);                                                    // create object of Sender class


                    receiver = new Receiver(socket);                                                // create object of Receiver class
                    receiver.start();                                                               // running thread method of Receiver class

                }catch(IOException e1){
                    try {
                        CommInfodoc.insertString(CommInfodoc.getLength(), "Client Error!", CommInfodoc.getStyle("black"));
                    } catch (BadLocationException e2) {
                        e2.printStackTrace();
                    }
                    e1.printStackTrace();
                } catch (BadLocationException e1) {
                    e1.printStackTrace();
                }
            }
        }
    });


    btnSend.addActionListener(new ActionListener() {    // Send button actionListener
        public void actionPerformed(ActionEvent e) {
            mes = "["+Id+"]: " +Chattextfield.getText();                                            // message is that [ID]:~~~
            Chattextfield.setText(null);
            try {
                message=mes.getBytes("utf-8");                                                      // convert message to byte[]
                Chattingdoc.insertString(Chattingdoc.getLength(), "\n"+mes, Chattingdoc.getStyle("black"));   // show my message in chatting UI
                sender.sendmessage(message);                                                        // send message using sender.sendmessage() method
                message = null;
            } catch (BadLocationException e1) {
                e1.printStackTrace();
            } catch (UnsupportedEncodingException e1) {
                e1.printStackTrace();
            }

        }
    });

    btnFindfile.addActionListener(new ActionListener() {    // Find File button actionListener
        public void actionPerformed(ActionEvent arg0) {

            Frame f = new Frame();
            FileDialog fileOpen = new FileDialog(f , "Open File", FileDialog.LOAD);     // I use filedialog to select file

            f.setVisible(false);

            fileOpen.setDirectory("c:\\jdk1.5");

            fileOpen.setVisible(true);
            if(fileOpen.getFile() == null) return;

            file = new File(fileOpen.getDirectory() +"/" +fileOpen.getFile());          // Initialize file using selected file in filedialog
```

Source code – ChatUi class

```
679
680
681
682
683
684                 try {
685                     Filetransferdoc.insertString(Filetransferdoc.getLength(), "\n***File to transfer Information***", Filetransferdoc.getStyle("black"));       // These codes carrying out
686                     Filetransferdoc.insertString(Filetransferdoc.getLength(), "\nFile Path: "+file.getAbsolutePath(), Filetransferdoc.getStyle("black"));       // showing information of
687                     Filetransferdoc.insertString(Filetransferdoc.getLength(), "\nFile Name: "+file.getName(), Filetransferdoc.getStyle("black"));               // file to transfer
688                     Filetransferdoc.insertString(Filetransferdoc.getLength(), "\n**********************************", Filetransferdoc.getStyle("black"));       //
689                 } catch (BadLocationException e) {
690                     e.printStackTrace();
691                 }
692
693
694
695
696
697
698
699             }
700         });
701
702
703         btnkeygeneration.addActionListener(new ActionListener() {       // Key generation button actionListener
704             public void actionPerformed(ActionEvent arg0) {
705
706                 RSA = new RSAEncryption();                              // Initialize RSA
707                 keyPair = RSA.RSAkeygenerate();                         // Initialize keyPair using RSAkeygenerate() method. this method returns keyPair.
708                 mypublicKey = keyPair.getPublic();                      // Initialize mypublicKey using keyPair
709                 privateKey = keyPair.getPrivate();                      // Initialize privateKey using keyPair
710                 byte[] mypubk = mypublicKey.getEncoded();                                                                                   // These codes carry out showing information of
711                 byte[] prik = privateKey.getEncoded();                                                                                      // my public key(RSA) and private key(RSA) in
712                 String pbk = "";                                                                                                            // chatting UI
713                 for(byte b: mypubk) pbk = pbk+Integer.toString((b & 0xff)+0x100, 16).substring(1)+" ";                                      //
714                 String pik = "";                                                                                                            //
715                 for(byte b: prik) pik = pik+Integer.toString((b & 0xff)+0x100, 16).substring(1)+" ";                                        //
716                                                                                                                                             //
717                                                                                                                                             //
718                 try {                                                                                                                       //
719                     keyInfodoc.insertString(keyInfodoc.getLength(), "Publick Key: \n"+pbk, keyInfodoc.getStyle("black"));                   //
720                     keyInfodoc.insertString(keyInfodoc.getLength(), "\nPublic Key Length : "+mypubk.length+ " byte" , keyInfodoc.getStyle("black"));       //
721                     keyInfodoc.insertString(keyInfodoc.getLength(), "\n"+ "Private Key: \n"+pik, keyInfodoc.getStyle("black"));                //
722                     keyInfodoc.insertString(keyInfodoc.getLength(), "\nPrivate Key Length : "+prik.length+ " byte", keyInfodoc.getStyle("black"));       //
723
724                 } catch (BadLocationException e) {
725                     e.printStackTrace();
726                 }
727
728
729
730             }
731         });
732
733         sendFile.addActionListener(new ActionListener() {       // Send File button actionListner
734             public void actionPerformed(ActionEvent arg0) {
```

Source code – ChatUi class

```
735                 sym = new SymEncryption();                      // initialize sym
736                 symKey = sym.AESkeygeneration();                // initialize symKey using sym. AESkeygeneration() method generate symmetric key and returns it
737                 sender.sendSymKey(symKey);                      // send symmetric key first
738                 RSASig = new RSASignature();                    // initialize RSASig
739                 Path path = Paths.get(file.getAbsolutePath());  // create path using file path comes from file.getAbsolutePath()
740                 byte[] data = null;
741                 try {
742                     data = Files.readAllBytes(path);            // initialize data with byte[] of file.
743                 } catch (IOException e) {
744                     e.printStackTrace();
745                 }
746                 byte[] signature = RSASig.generationRSASignature(data, privateKey);     // initialize signature using RSASig.generationRSASignature method.
747                                                                                         // this method create AES signature of file using private key and returns the signature
748
749                 sender.sendFile(data);                          // send byte[] of file secondly
750
751
752
753
754
755
756                 try {
757                     Filetransferdoc.insertString(Filetransferdoc.getLength(), "\n****** Transfer completed!! ******", Filetransferdoc.getStyle("black"));              // These codes carry out showing
758                     byte[] symKeybb = symKey.getEncoded();                                                                                                          // information of symmetric key and
759                     String symKeystring = "";                                                                                                                       // "Transfer completed" in chatting UI
760                     for(byte b: symKeybb) symKeystring = symKeystring+Integer.toString((b & 0xff)+0x100, 16).substring(1)+" ";                                      //
761                     Filetransferdoc.insertString(Filetransferdoc.getLength(), "\nSymmetric Key: " + symKeystring, Filetransferdoc.getStyle("black"));                 //
762                     Filetransferdoc.insertString(Filetransferdoc.getLength(), "\nSymmetric Key Size: " + symKeybb.length + "byte", Filetransferdoc.getStyle("black"));  //
763                     Filetransferdoc.insertString(Filetransferdoc.getLength(), "\n**********************************", Filetransferdoc.getStyle("black"));              //
764                 } catch (BadLocationException e) {
765                     // TODO Auto-generated catch block
766                     e.printStackTrace();
767                 }finally {
768                     sender.sendFileSignature(signature);            // finally, send signature of file that transferred.
769                 }
770
771
772             }
773         });
774
775         btnSetFileOption.addActionListener(new ActionListener() {   // Set File Option button actionListener
776             public void actionPerformed(ActionEvent arg0) {
777                 filePath = filepathtextfield.getText();             // initialize filePath
778                 fileName = filenametextfield.getText();             // initialize fileName
779
780                 filepathtextfield.setEnabled(false);
781                 filenametextfield.setEnabled(false);
782                 btnSetFileOption.setEnabled(false);
783             }
784         });
785
786
787         sendPublicKey.addActionListener(new ActionListener() {      // Send Public Key button actionListener
788             public void actionPerformed(ActionEvent e) {
789                 sender.sendPublicKey(mypublicKey);                  // send my public key to opponent
790
```

```
791             }
792         });
793     }
794 }
795
```

Source code – RSAEncryption class

```java
package Chatting;

import java.security.*;

public class RSAEncryption {
    private KeyPairGenerator generator;
    private KeyPair keyPair;
    byte[] pubk;
    byte[] prik;

    public KeyPair RSAkeygenerate() {  // This method create RSA Keypair(public key, private key)
        try {
            generator = KeyPairGenerator.getInstance("RSA");
            generator.initialize(1024);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return generator.generateKeyPair();
    }

    public byte[] RSAencryption(byte[] plaintext, PublicKey cpKey) { // This method carries out RSA encryption of byte[] using opponent public key and returns encrypted byte[]
        Cipher cipher;
        byte[] b0 = null;
        try {
            cipher = Cipher.getInstance("RSA");
            cipher.init(Cipher.ENCRYPT_MODE, cpKey);
            b0 = cipher.doFinal(plaintext);
        } catch (Exception e) {
            e.printStackTrace();
        }

        return b0;
    }

    public byte[] RSAdecryption(byte[] ciphertext, PrivateKey myprik) { // This method carries out RSA decryption of encrypted byte[] using my private key and returns decrypted byte[]
            Cipher deccipher;
            byte[] b1 = null;
            try {
                deccipher = Cipher.getInstance("RSA");
                deccipher.init(Cipher.DECRYPT_MODE, myprik);
                b1 = deccipher.doFinal(ciphertext);
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }


        return b1;

    }


}
```

Source code – RSASignature class

```java
package Chatting;

import java.security.*;

public class RSASignature {

    public byte[] generationRSASignature(byte[] file, PrivateKey prik){  // This method create signature of file using private key and byte[] of file and returns the signature of file
        byte[] signatureBytes = null;
        Signature sig1;
        try {
            sig1 = Signature.getInstance("SHA512WithRSA");
            sig1.initSign(prik);
            sig1.update(file);
            signatureBytes = sig1.sign();

        } catch (NoSuchAlgorithmException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InvalidKeyException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SignatureException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return signatureBytes;

    }


    public boolean verifyRSASignature(byte[] file, byte[] signature, PublicKey pbk){ // this method verify that file has been corrupted or changed using byte[] of file and opponent public key. returns boolean value
        Signature sig2 ;
        boolean returns = false;


        try {
            sig2 = Signature.getInstance("SHA512WithRSA");
            sig2.initVerify(pbk);
            sig2.update(file);
            returns = sig2.verify(signature);


        } catch (InvalidKeyException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SignatureException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (NoSuchAlgorithmException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return returns;

    }
```

Source code – SymEncryption class

```java
package Chatting;

import java.security.*;
import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;

public class SymEncryption {
    private KeyGenerator keyGen;
    public Key AESkeygeneration() {          // This method generate Symmetric key and returns it
        try {
            keyGen = KeyGenerator.getInstance("AES");
            keyGen.init(128);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        Key key = keyGen.generateKey();
        return key;
    }
    public byte[] AESkeyEncrpytion(byte[] plaintext, Key key) { // This method carries out AES encryption using byte[] of plaintext and symmetric key and returns encrypted byte[]
        Cipher cipher;
        byte[] cipherbyte = null;
        try {
            cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, key);
            try {
                cipherbyte = cipher.doFinal(plaintext);
            } catch (Exception e) {
                e.printStackTrace();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return cipherbyte;
    }

    public byte[] AESdecryption(byte[] cipherbyte, SecretKey key) { // This method carries out AES decryption of encrypted byte[] using symmetric key
        Cipher cipher2;
        byte[] decryptbyte = null;
        try {
            cipher2 = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher2.init(Cipher.DECRYPT_MODE, key);
            decryptbyte = cipher2.doFinal(cipherbyte);
        } catch (NoSuchAlgorithmException | NoSuchPaddingException e) {

            e.printStackTrace();
        } catch (InvalidKeyException e) {

            e.printStackTrace();
        } catch (IllegalBlockSizeException e) {

            e.printStackTrace();
        } catch (BadPaddingException e) {

            e.printStackTrace();
        }
        return decryptbyte;
    }
    }
```

Source code – EncryptedSymKey class

```java
package Chatting;

import java.io.Serializable;

public class EncryptedSymKey implements Serializable{  // This class was created to send RSA encrypted symmetric keys.
    private byte[] encrpytedSymKey;     // this variable is used for carry encrypted symmetric key

    public EncryptedSymKey(byte[] encryptedSymKey) { // When generate object of this class, you need byte[] of encrypted symmetric key.
        this.encrpytedSymKey=encryptedSymKey;

    }

    public byte[] getEncrpytedSymKey() {
        return encrpytedSymKey;
    }

    public void setEncrpytedSymKey(byte[] encrpytedSymKey) {
        this.encrpytedSymKey = encrpytedSymKey;
    }
}
```

Source code – EncryptedSymFile class

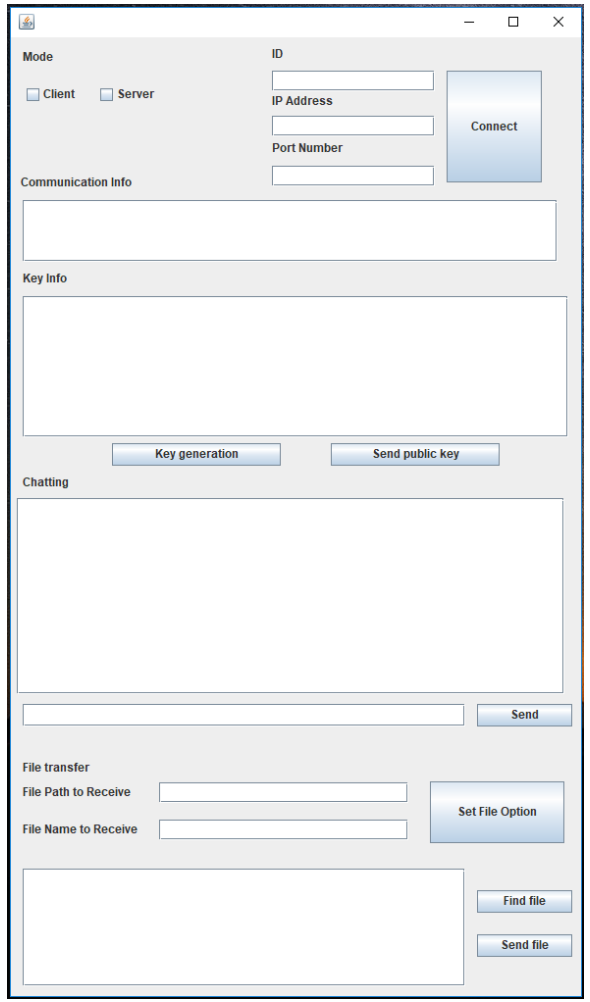
```java
package Chatting;

import java.io.Serializable;

public class EncryptedSymFile implements Serializable{  // This class was created to transfer files with AES encoded byte arrays.
    private byte[] encryptedFile;       // this variable is used for carry encrypted byte[] of file

    public EncryptedSymFile(byte[] encryptedFile) { // When generate object of this class, you need byte[] of encrypted file.
        this.encryptedFile=encryptedFile;

    }

    public byte[] getEncryptedFile() {
        return encryptedFile;
    }

    public void setEncryptedFile(byte[] encryptedFile) {
        this.encryptedFile = encryptedFile;
    }


}
```

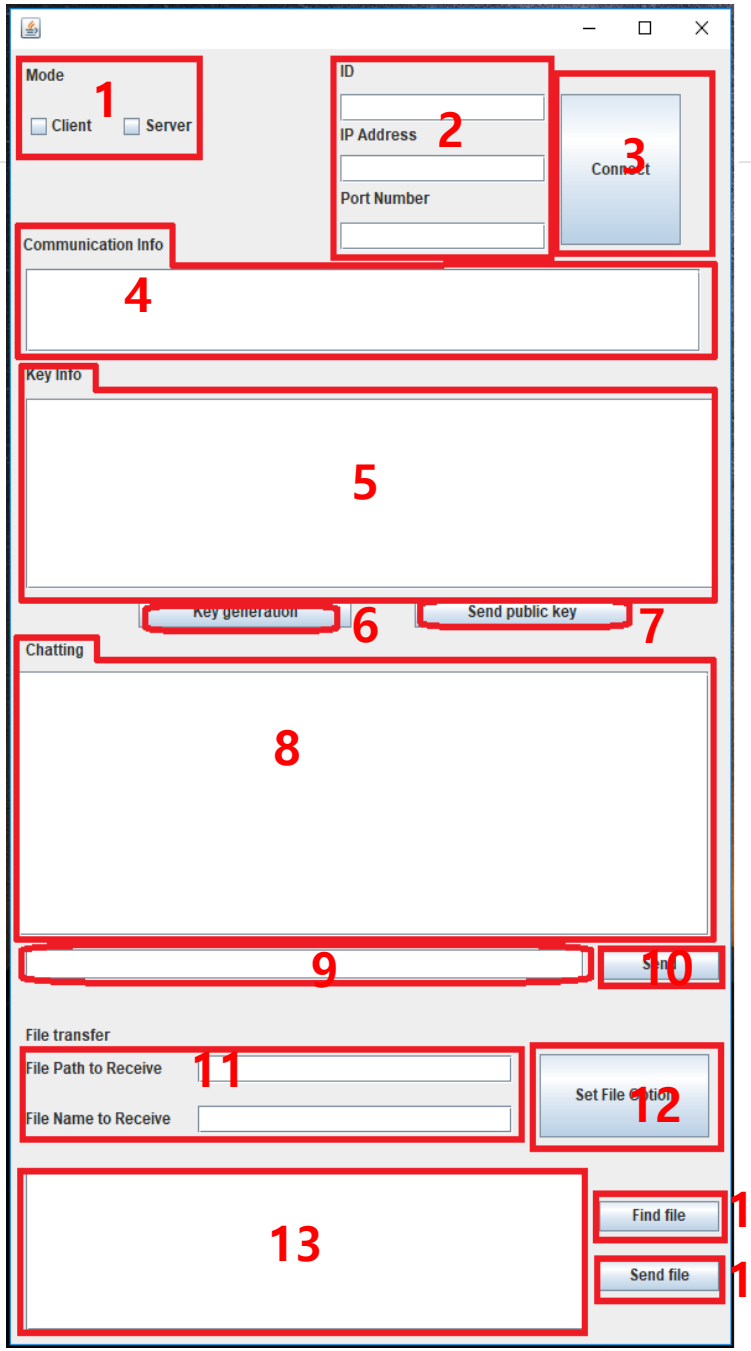
How to use

01

02

First view of the program





Functional Classifiation

1. Mode Select
2. Input Connection data
3. Connect
4. Connection Information
5. Key Information(private, public, opponent's public key)
6. Key Generation(private, public)
7. Send Public Key
8. Chat Window
9. Message Text Field
10. Send Message
11. Input File Data to Receive
12. Set File Data
13. File Transfer Information
14. Find File to Transfer
15. Send File

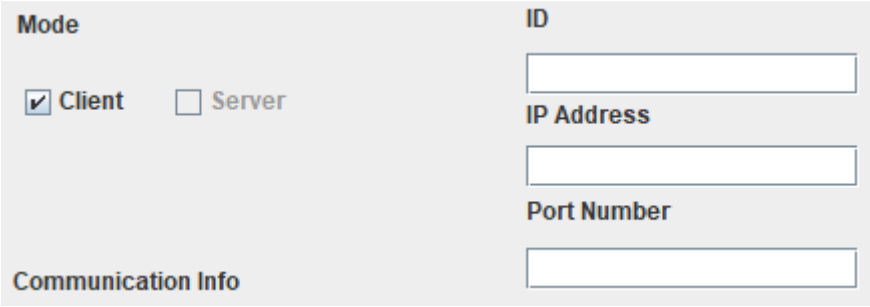
How to use

01

02

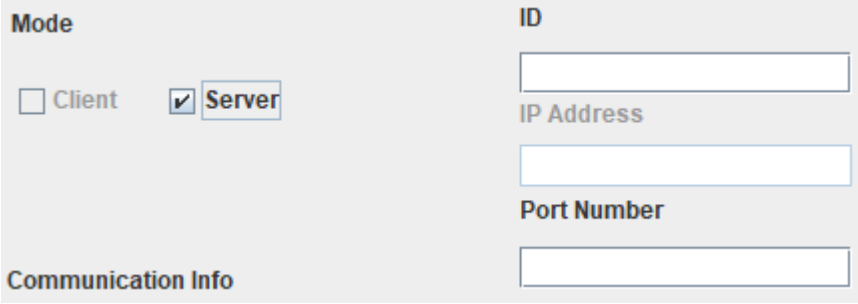
1. Mode Selection

Server Mode



- User can select the mode of connection (Client or Server). The variables for the connection and communication will be set depending on this selection.

Client Mode

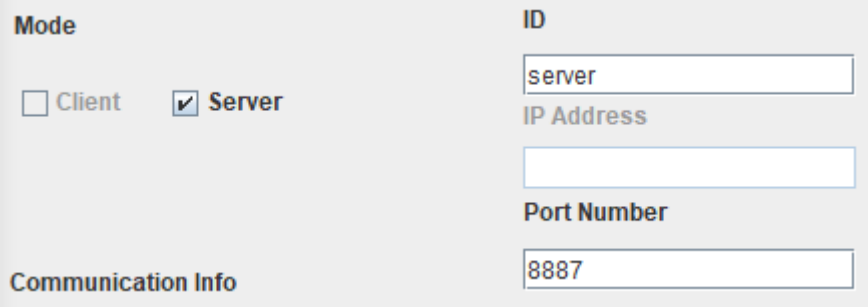


- When user select Server, “IP” is automatically set as user’s IP adress. As this auto setting, the input box for ID and IP and client check box will be blocked. Only the box for input “Port” is enable. Server must input the integer value for the port that will be used for the connection. To make connection between Server and Client, It must be preceded that openning the socket port by a server.

How to use

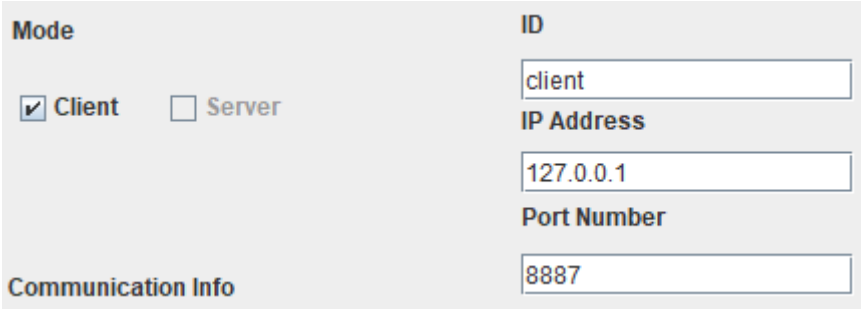
2. Input Connection data

Server Mode



- The Server should input his or her own ID and port number of. This ID will be used like 'nickname' in chat window..

Client Mode



- The Client should input his or her own ID and IP of the server. This ID will be used like 'nickname' in chat window.
- The IP of server is set as "127.0.0.1" (= "localhost"). Therefore, client should input "127.0.0.1" or "localhost" as the IP to make connection with the server.
- And finally, client have to input the appropriate port number that server opens. If port number is not matched, the connection will be failed.

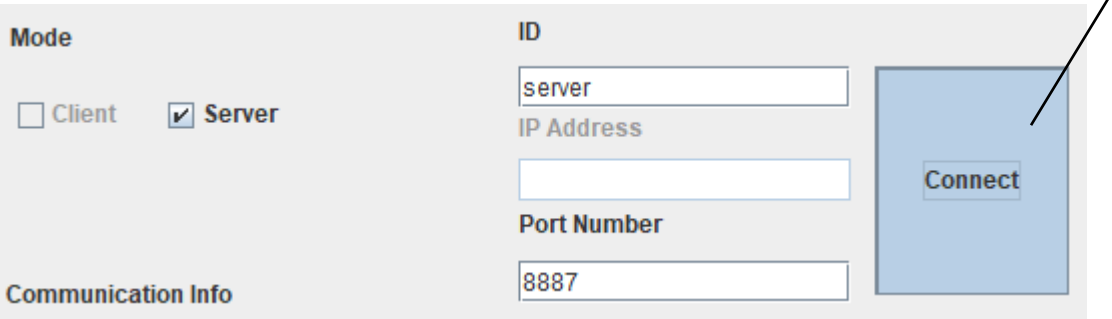
How to use

01

02

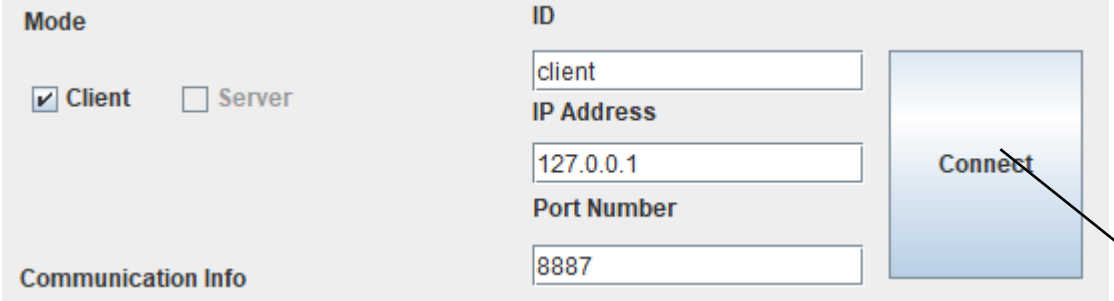
3. Connect

Server Mode



Waiting...

Client Mode



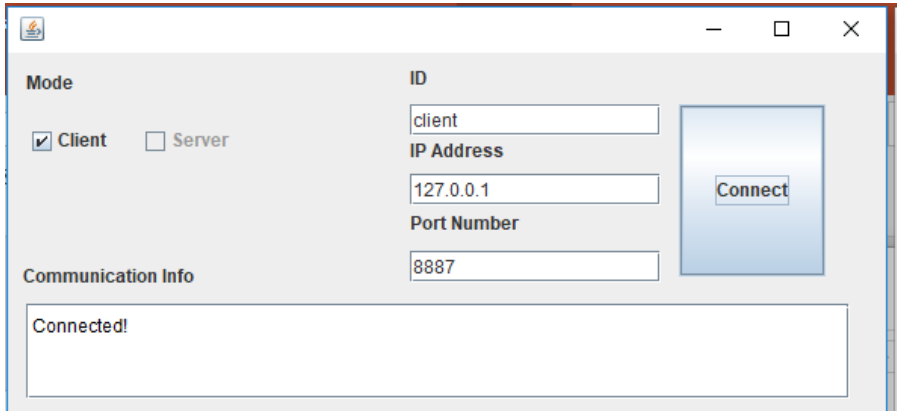
Click!

- With the selection of mode and input variables into input boxes, the program can start to make connection between server and client by using this button.
- As server socket notice the access of a client and accept it, the IO Stream that using socket communication will be activated. The thread for the communication will be started, server and client can send and receive data.
- Important thing is that, Server must open port and wait the client beforehand client try to access to make this connection.

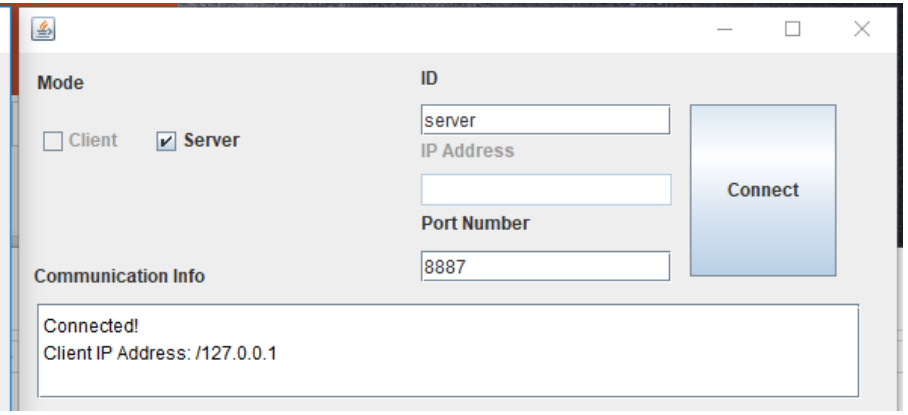
How to use

4. Connection Information

Client Mode

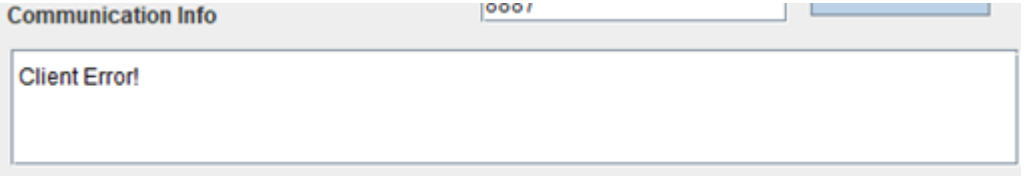


Server Mode



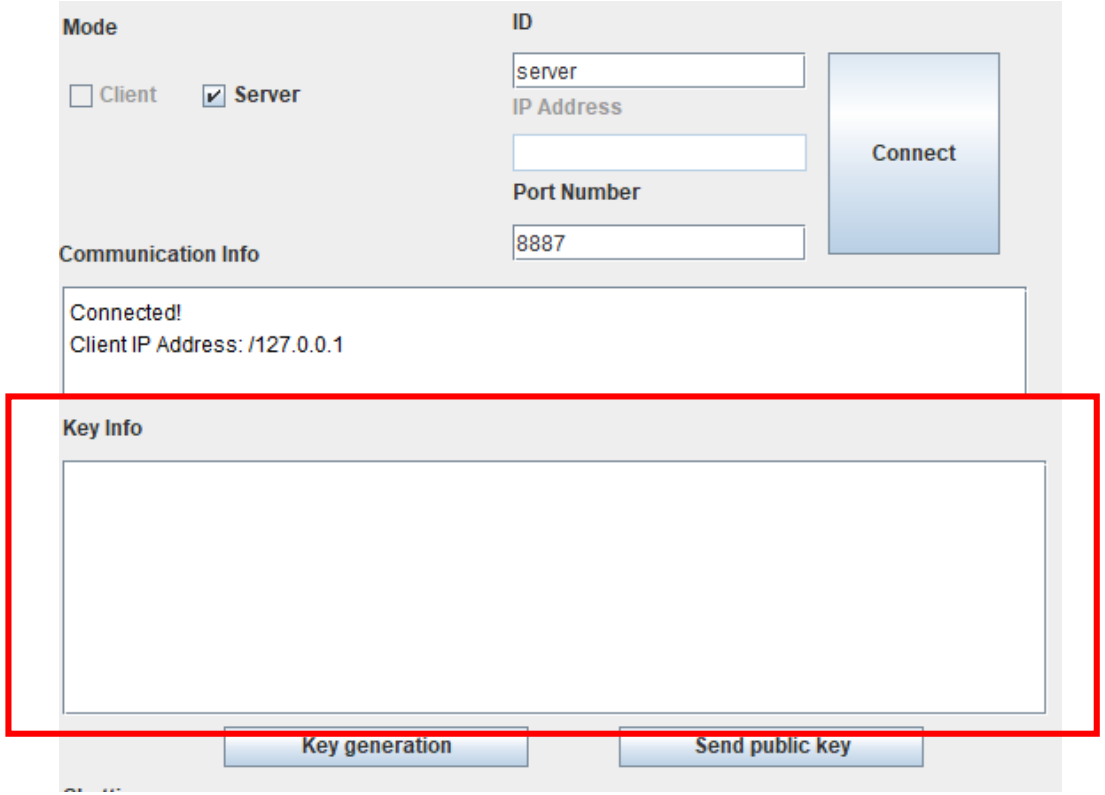
- This Panel shows the current information of the connection. When server opens the port and waits client, client accesses to this port and connection is succeed, or the connection is failed because of certain reason, user can check the status of connection at this panel.

Connection-Fail Case(server has not been opened)



How to use

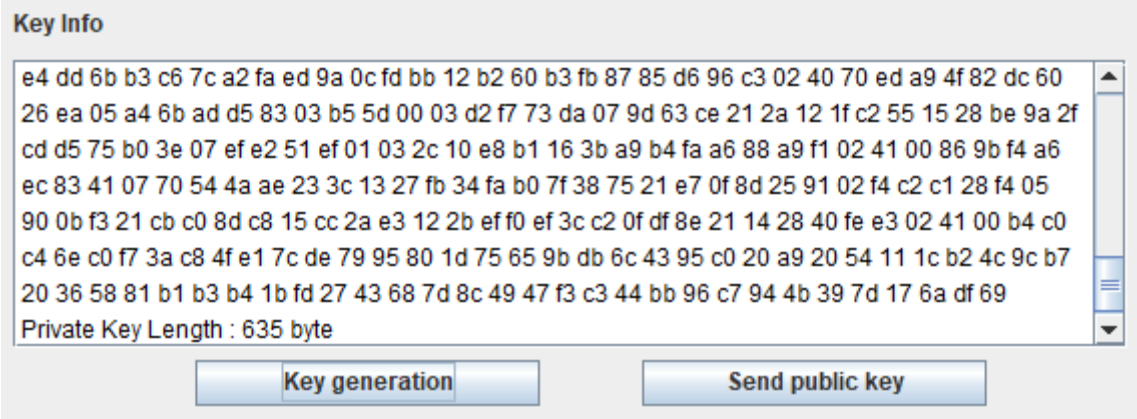
5. Key Information



- This Panel shows the information of user's public key, private key, and opponent's public key.
- To satisfy the integrity, this program uses the RSA encryption and decryption of message and AES secret key.
- The keys appear in hexadecimal on this panel and this panel also shows the byte size of the keys.

How to use

6. Key Generation(public, private)



- By using methods in RSAEcryption, user can generate his or her own public and private key.
- The information of public and private key appear in Key Information panel.
- User can see the key information using scroll.

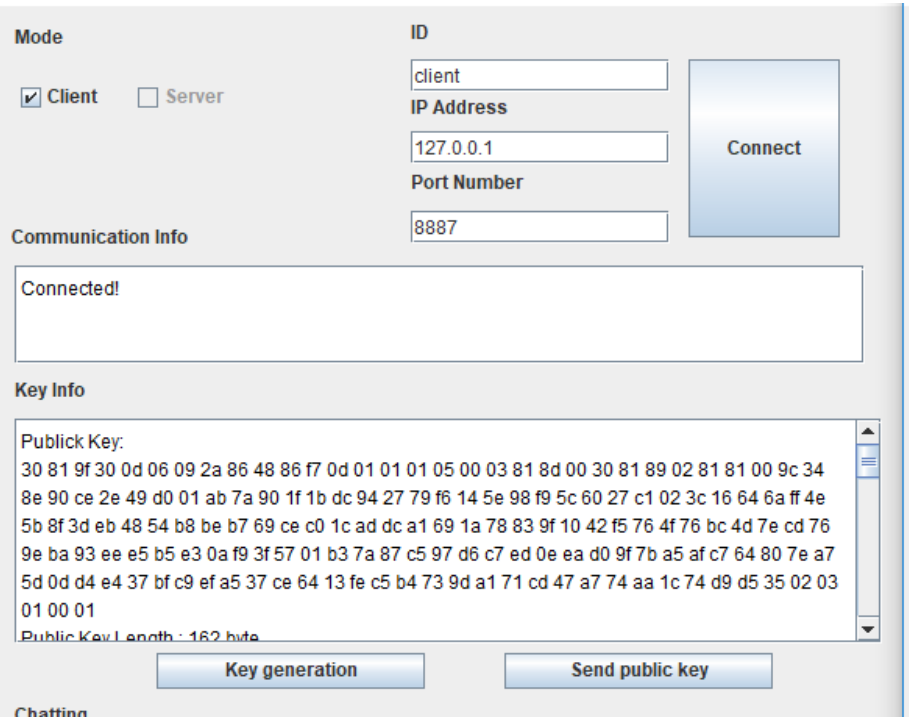
How to use

01

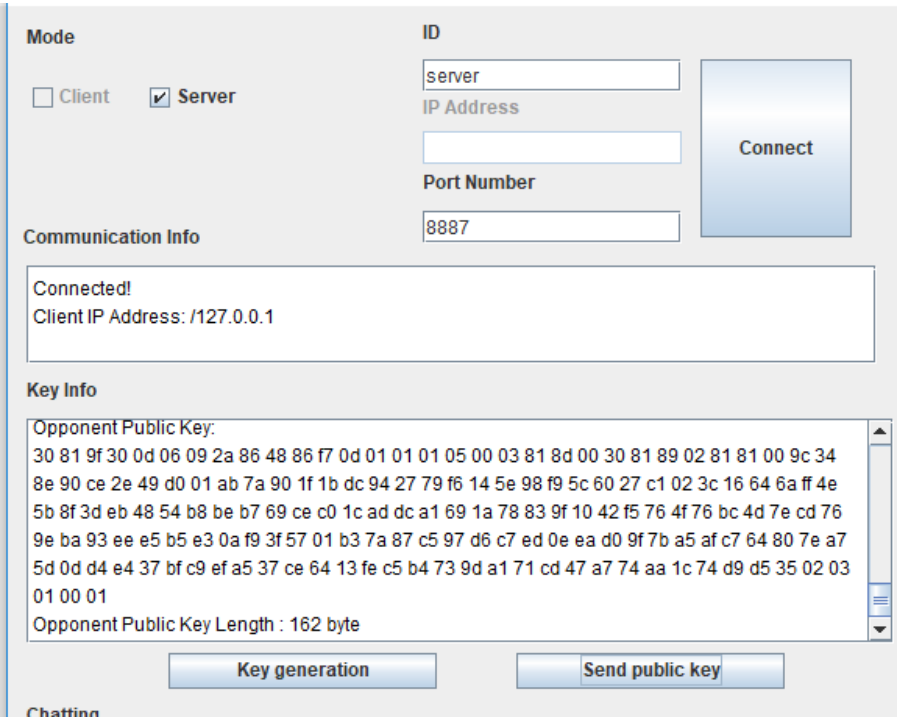
02

7. Send Public Key

Client Mode



Server Mode



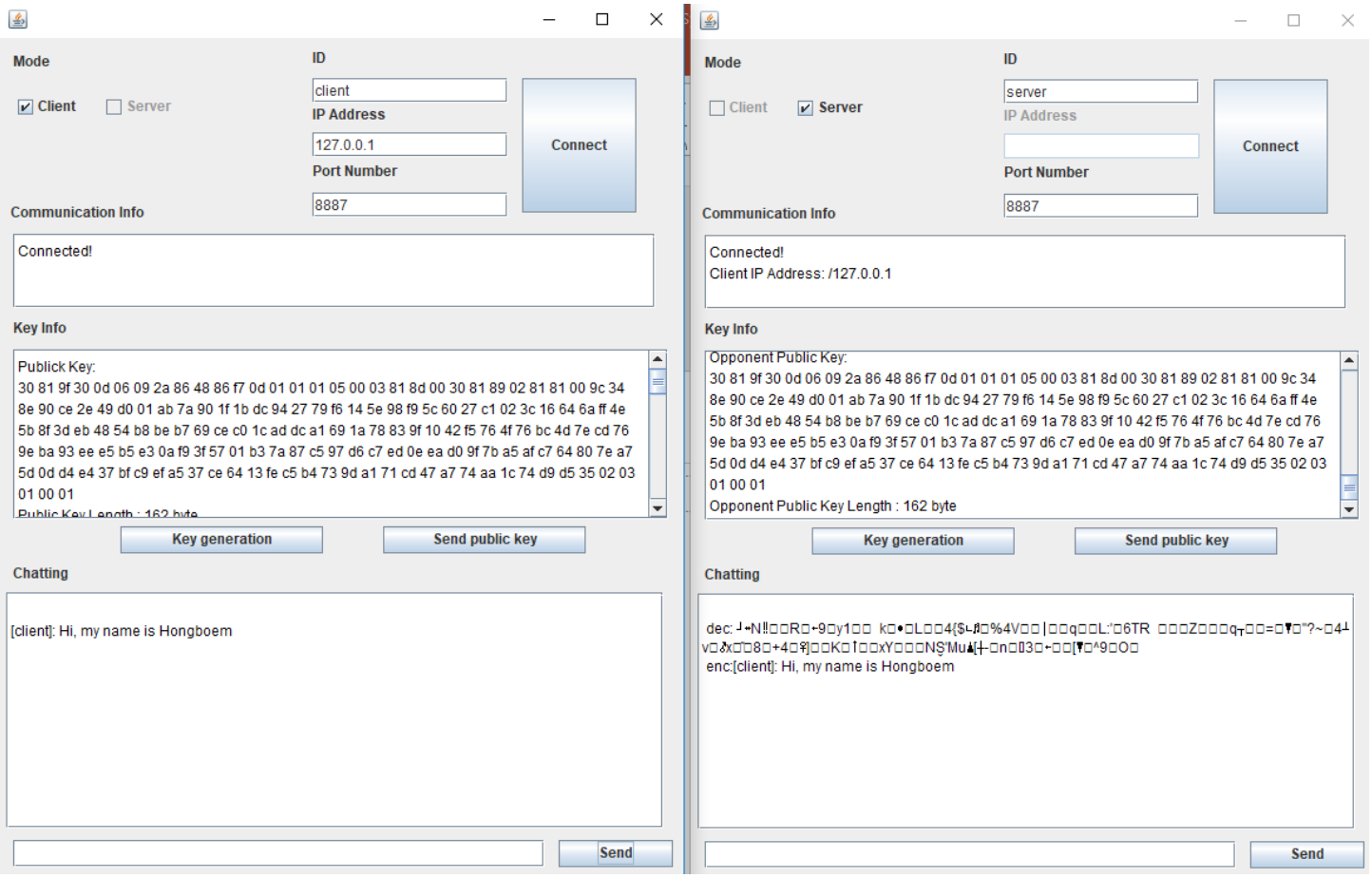
- User can send his or her own public key to opponent after generation of RSA keys.
- Opponent can check the information of received public key in Key Information panel.

How to use

01

02

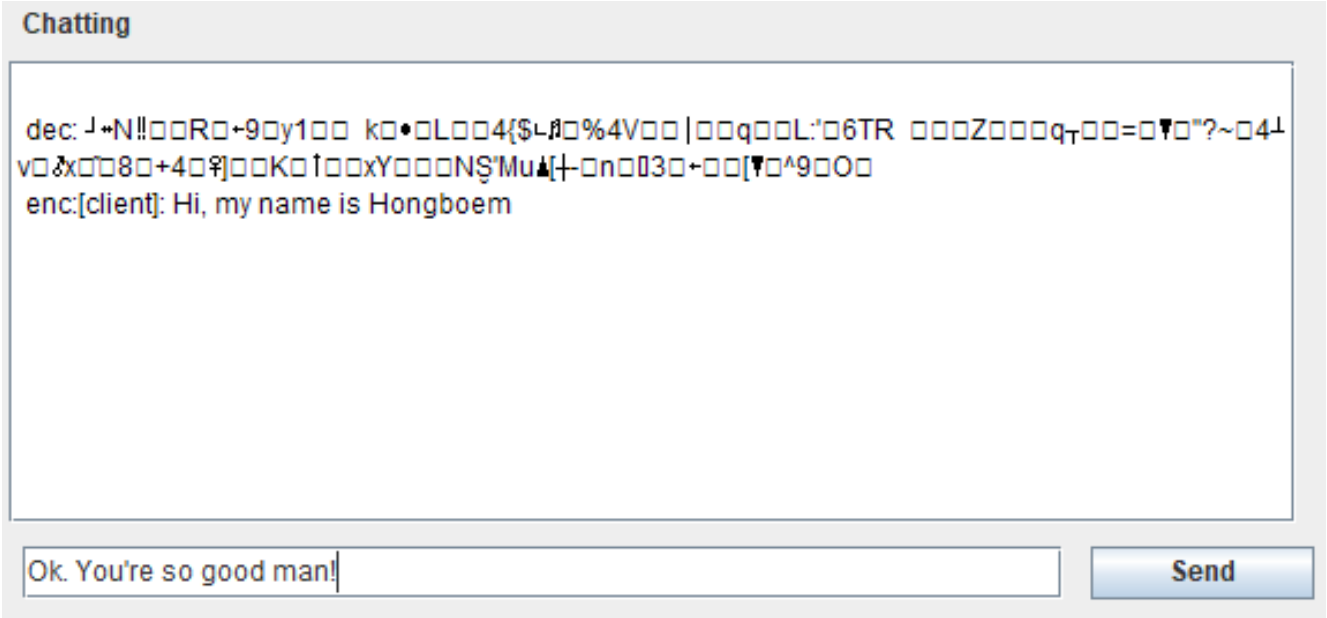
8. Chat Window



- This window shows chatting message in two types, encrypted message and decrypted message.
- Message from opponent was ecrypted with RSA public key in opponents side and decrypted with RSA private key in user side.
- User and opponent's ID appear in this window

How to use

9. Message Text Field



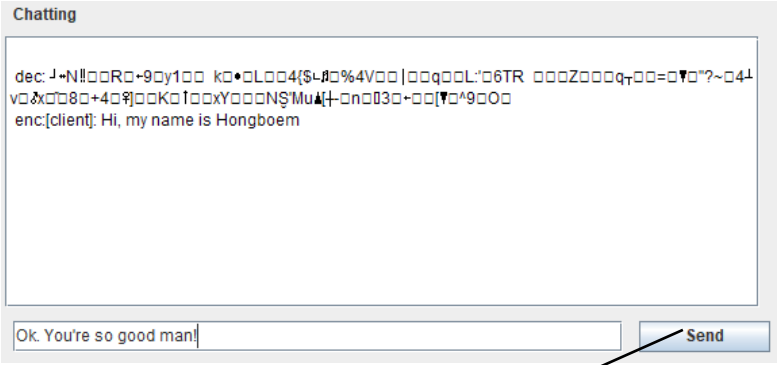
- User can input the message to send in this text field.

How to use

01

02

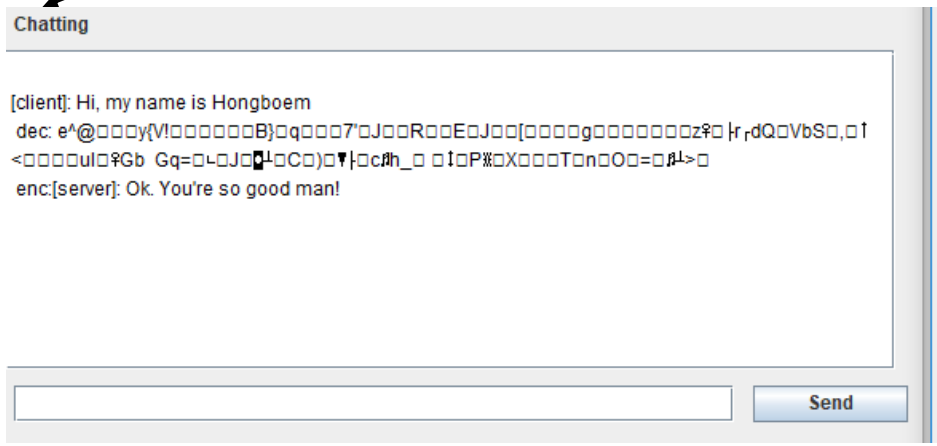
10. Send Message



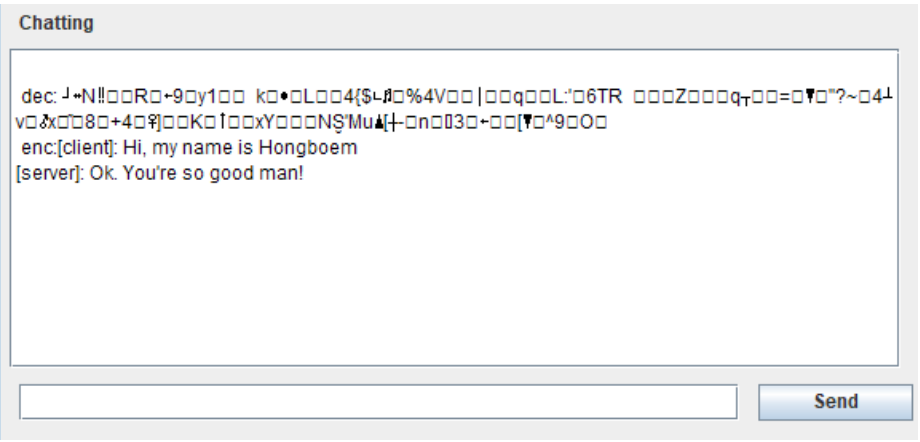
- This button send message in text filed to opponent.
- After clicked this button, the program get message from text filed and encrypt the message using opponent's public key. Then, send this encrypted object to opponent.

Click!

Client Mode



Server Mode

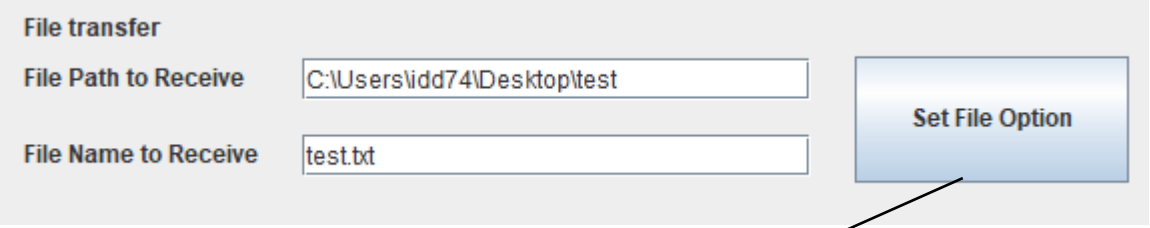


How to use

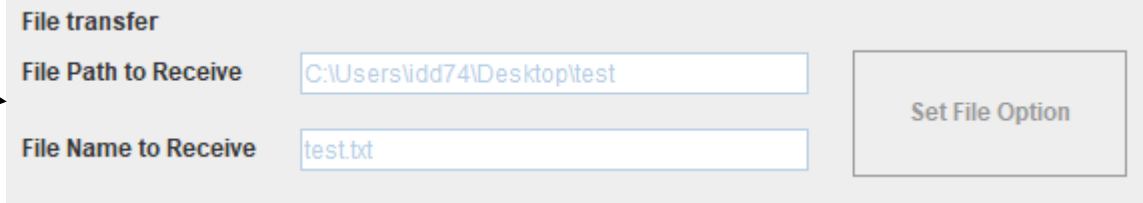
01

02

11 & 12. Input File Data to Receive & Set File Data



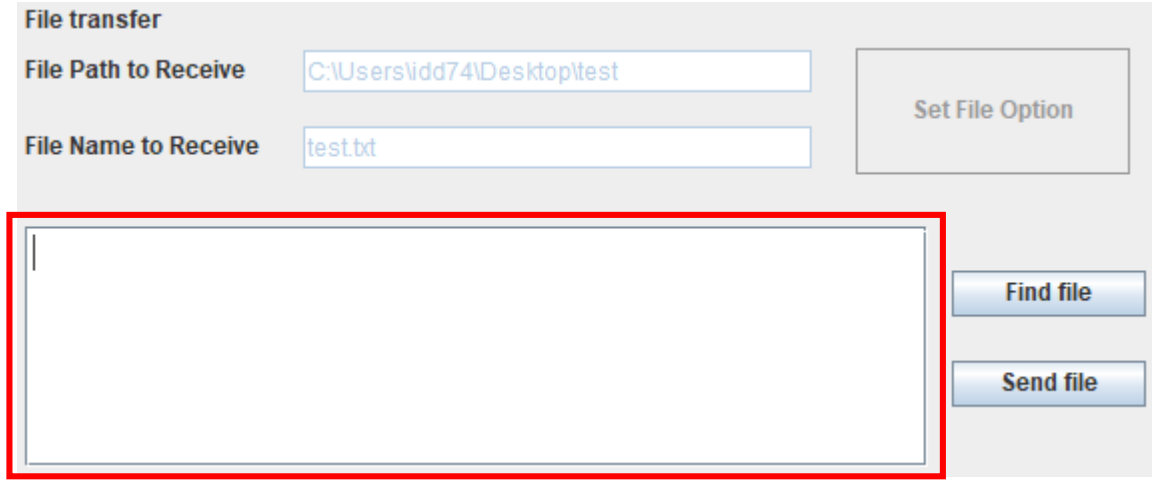
Click!



- User can specify file path to receive and file name to receive. Then, transferred file from opponent could be saved in the specified file path with specified file name.
- User input file path and file name in the box and click the Set File Option button. Then, input boxes and button would be blocked.
- When the user receives the file sent by the other party, the box and button are activated again and the option can be set again.
- If the user does not specify a specific path and name using this function, the file is saved as the default name in the default path.
- Default Name: Received_File
- Default Path: C:₩Users₩idd74₩Desktop
- (You can change default Name and default Path in ChatUi code row num 287)

How to use

13. File Transfer Information

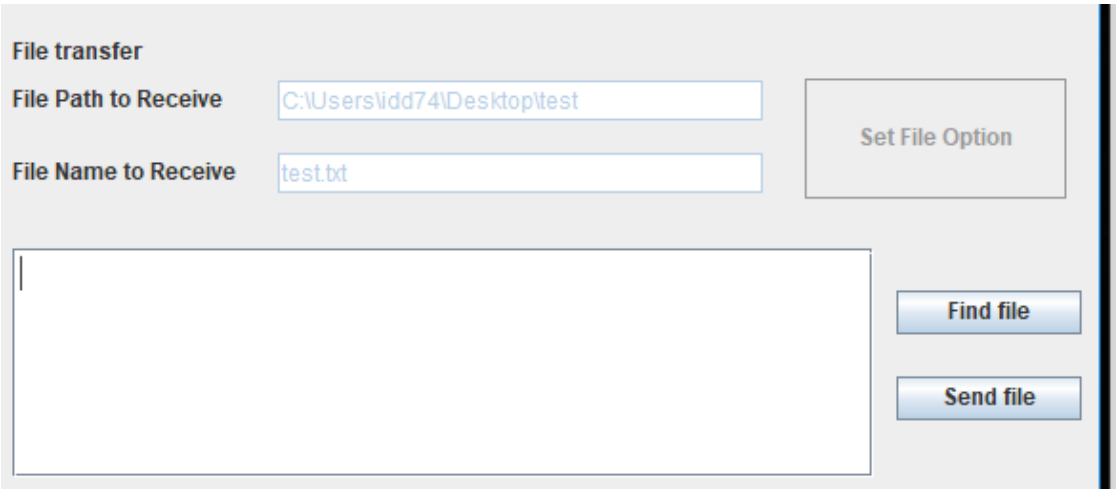


- This panel shows the information of the file to transfer: file size, file path, file name and information of secret key which is used to encrypt file in AES.
- This panel also shows the information of the received file from opponent: file path, file name and information of secret key which is used to decrypt encrypted file in AES.

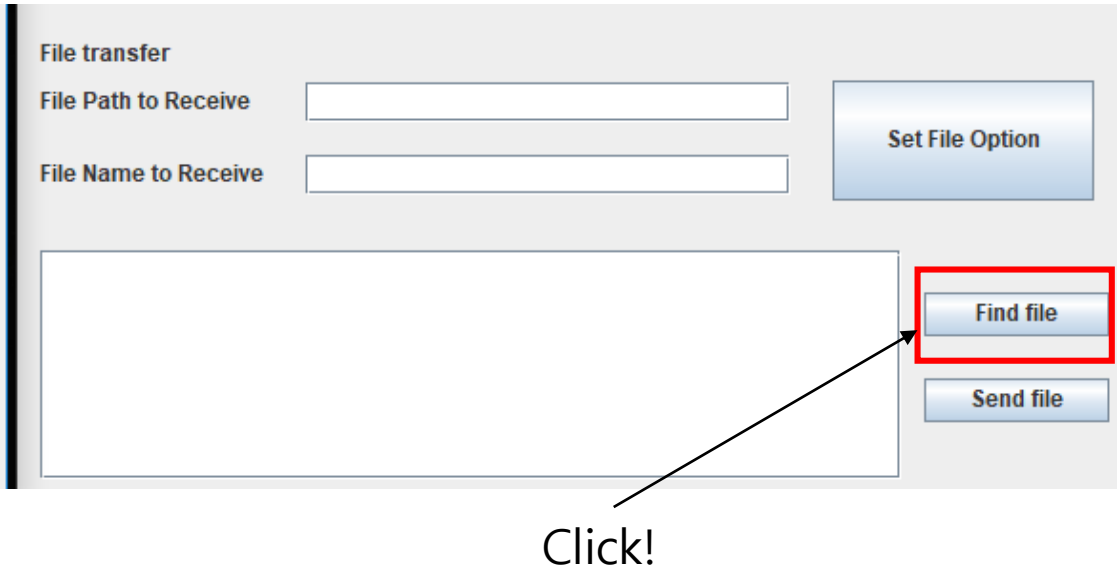
How to use

14. Find File to Transfer

Client Mode



Server Mode



- Firstly, click the Find File button.

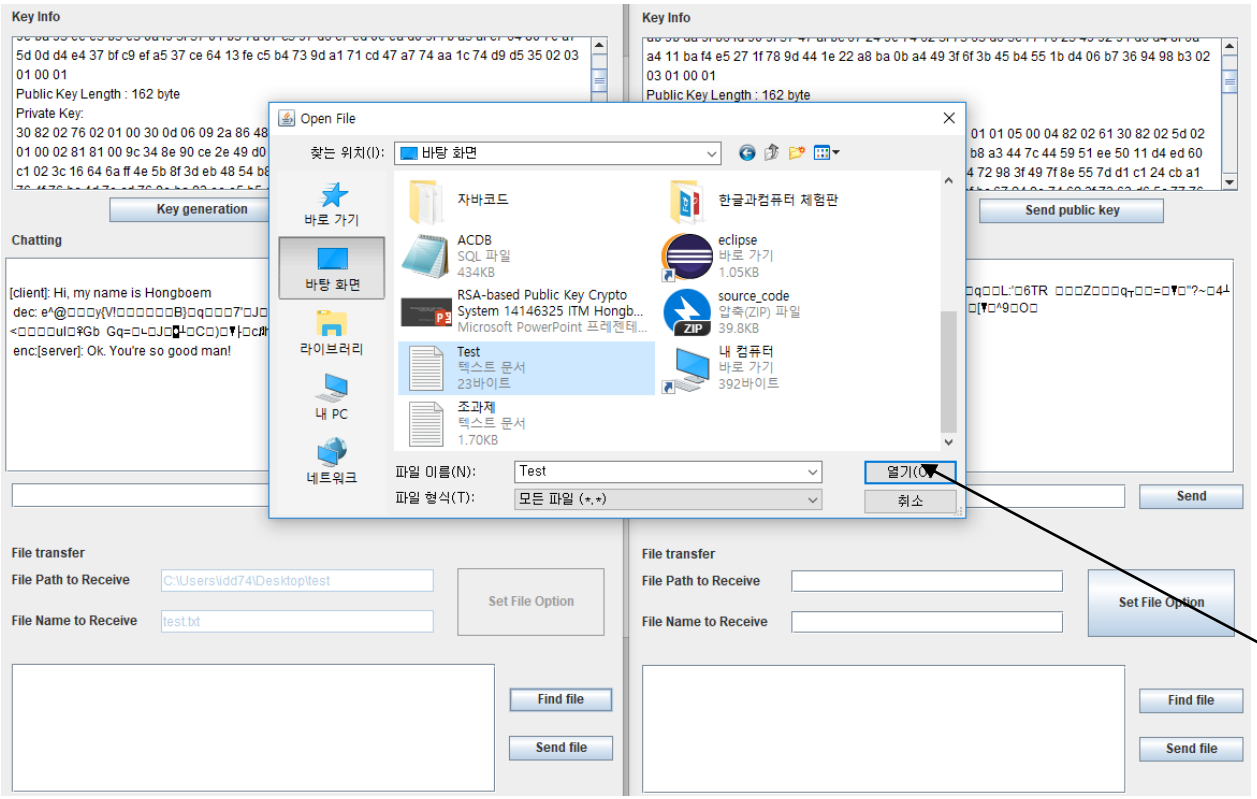
How to use

01

02

14. Find File to Transfer

Client Mode **Server Mode**



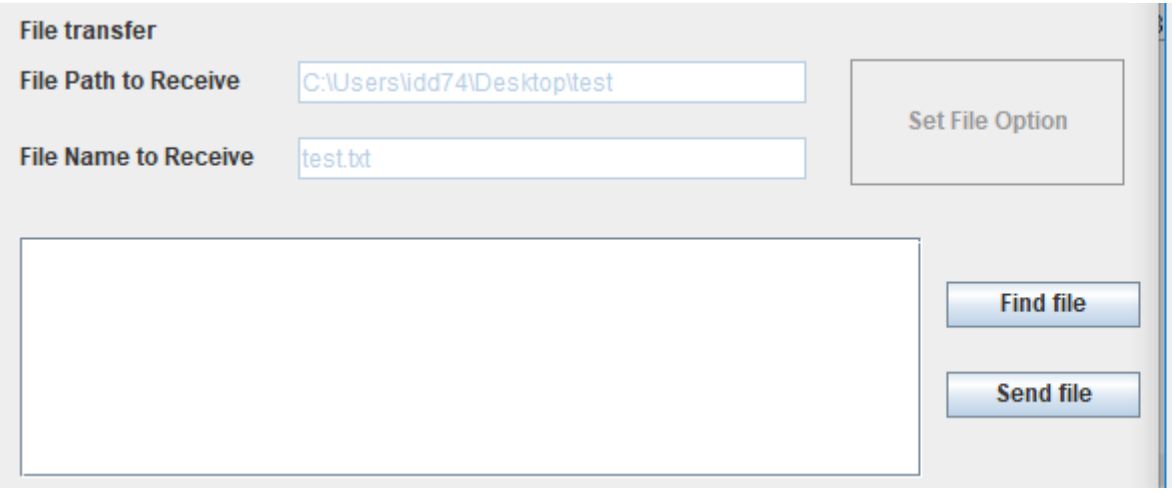
Click!

- Then, select file to transfer.

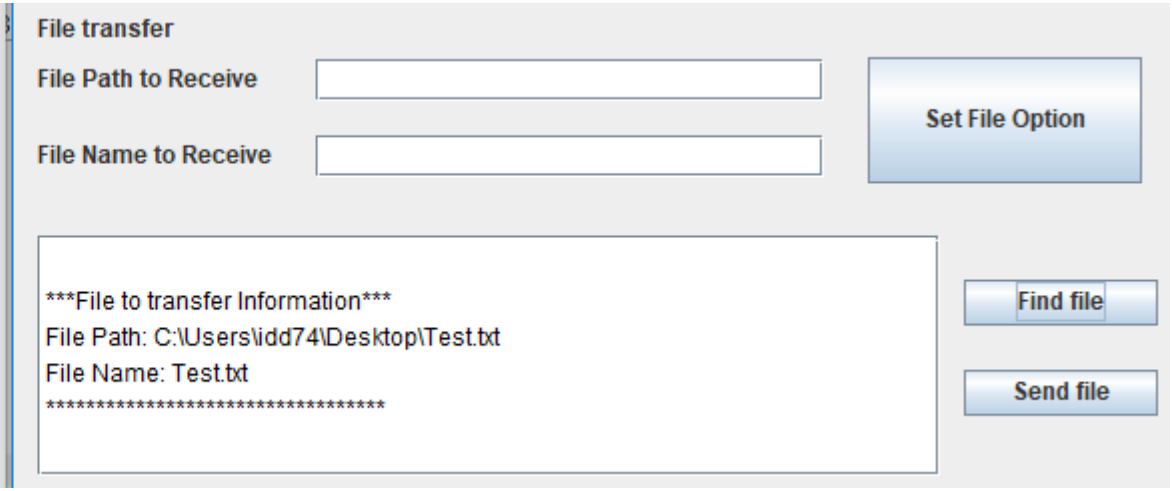
How to use

14. Find File to Transfer

Client Mode



Server Mode



- Then, information of file to transfer (file path, file name) would be appeared in File Transfer panel

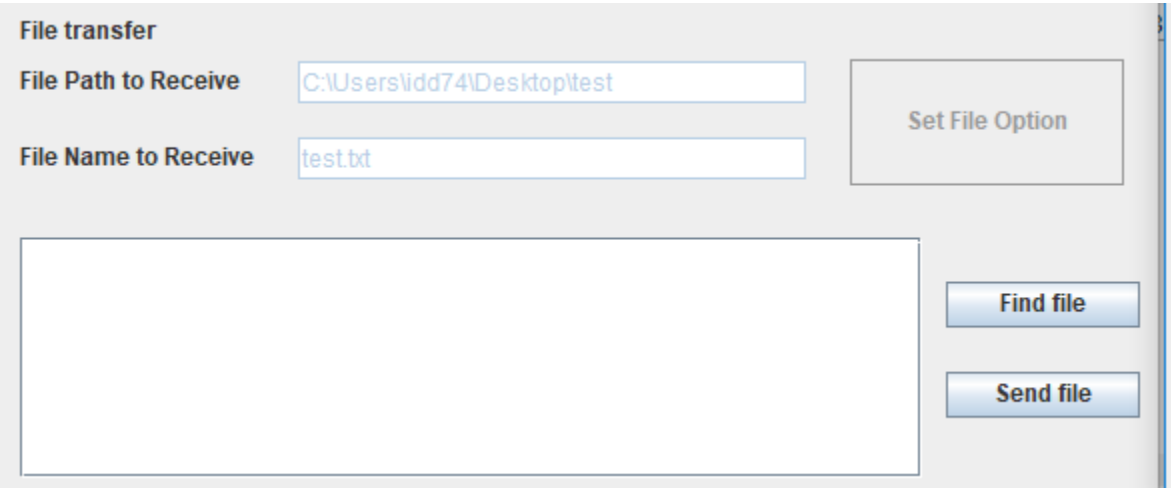
How to use

01

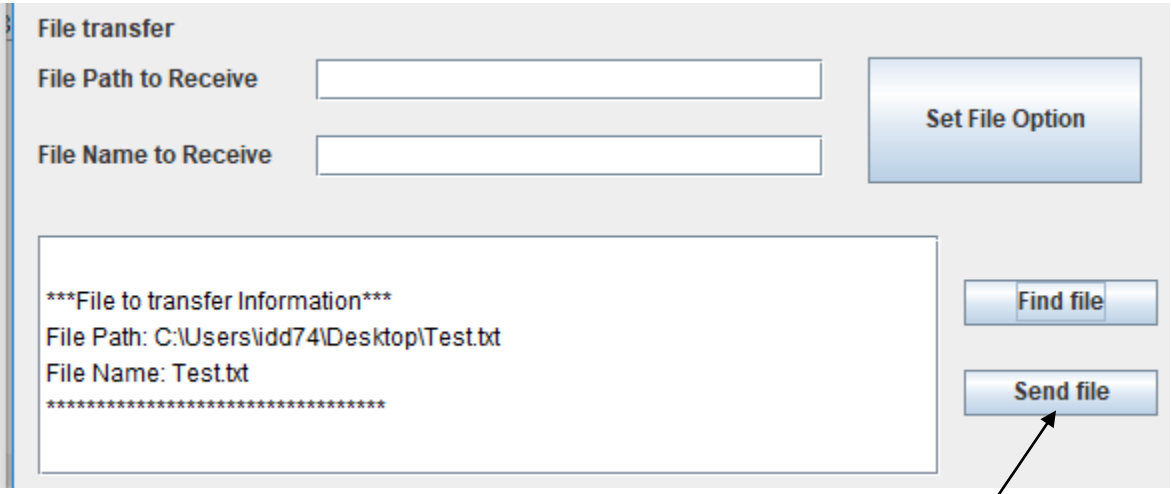
02

15. Send File

Client Mode



Server Mode

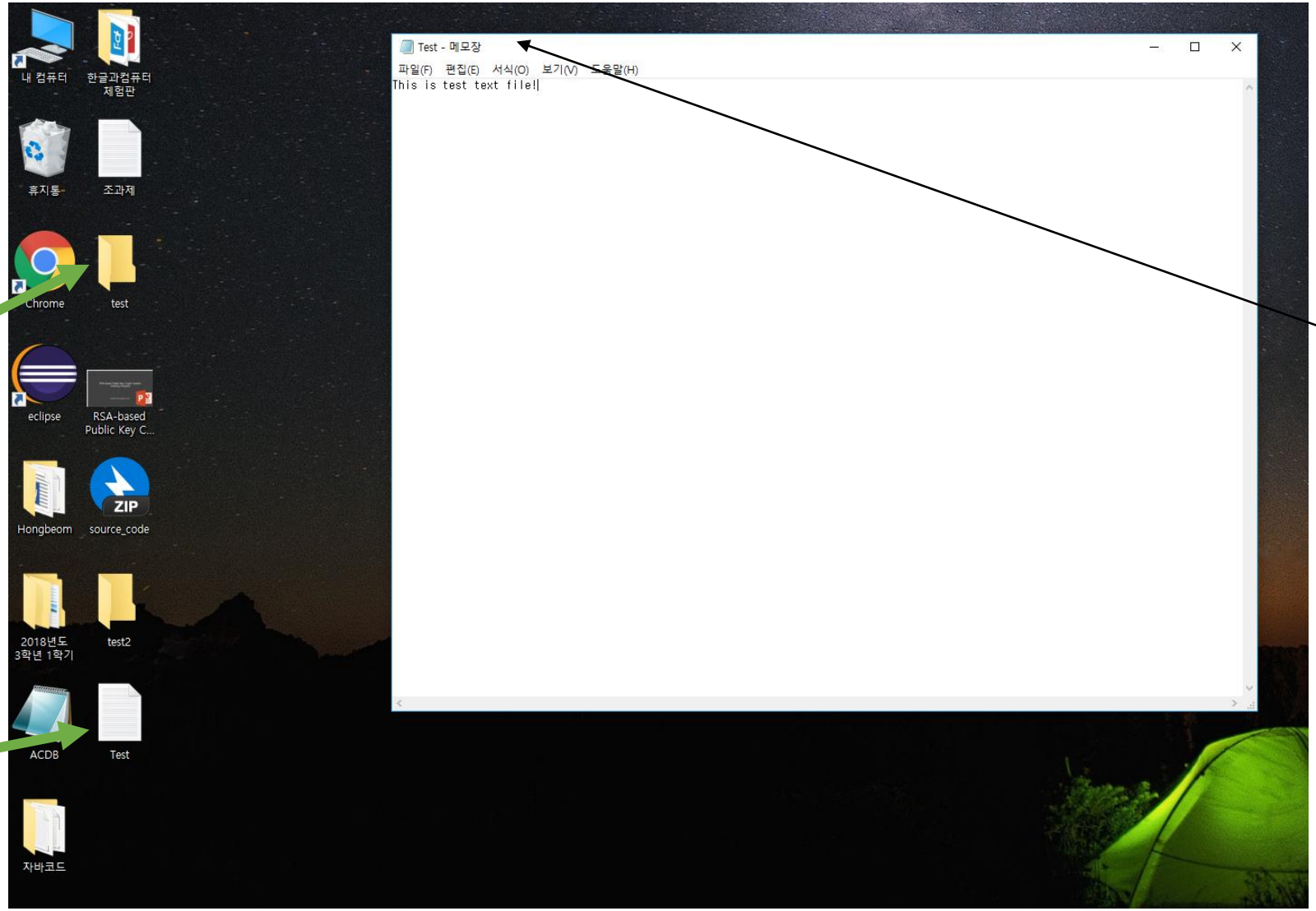


Click!

- After selection of file to transfer using Find file button, user can simply click the Send file button to send file.
- After the button clicked, program generate symmetric secret key(AES) and encrypt this secret key using user's public key. Then, send this encrypted key object to opponent.
- Next, program encrypt the selected file using secret key(AES) and send this encrypted file object to opponent.
- Finally, program create signature of users using file and user's private key. Then send the signature to opponent.

How to use

15. Find File to Transfer



Specified File Path

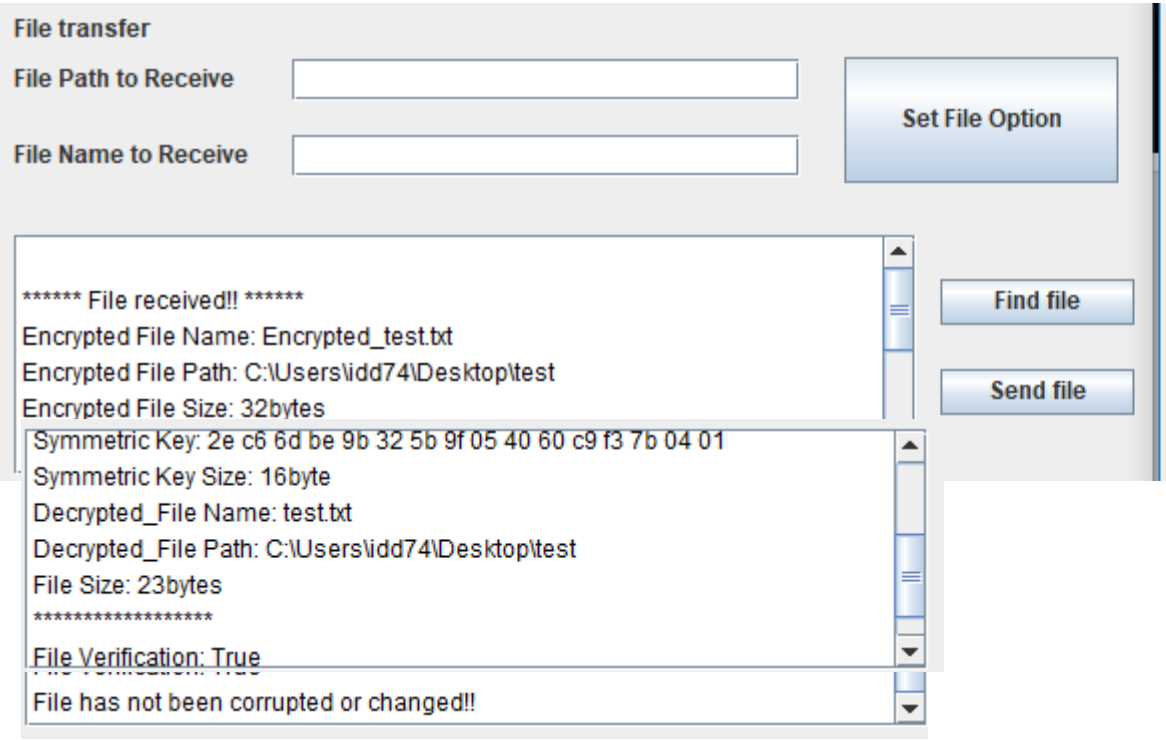
File to transfer

File to transfer

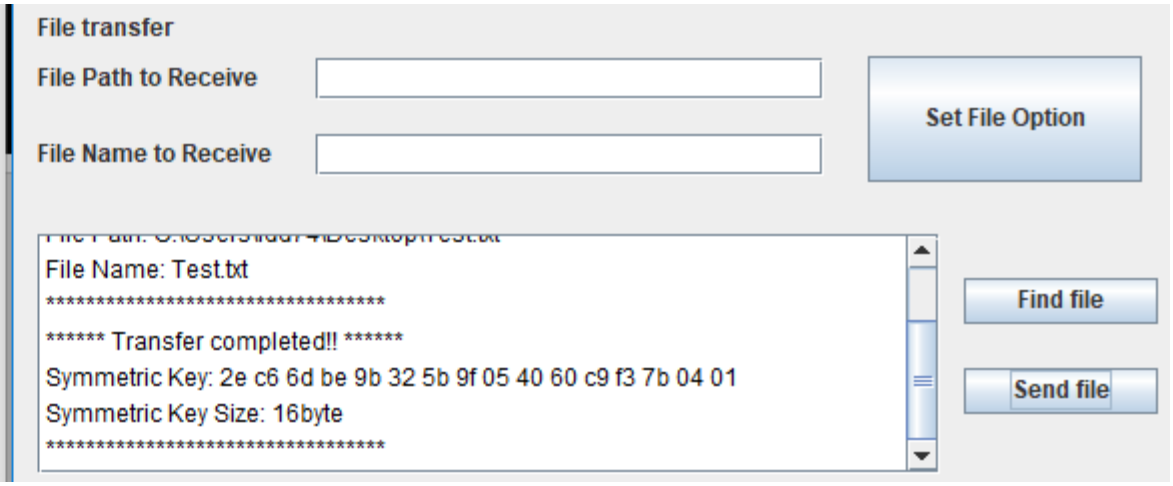
How to use

15. Send File

Client Mode



Server Mode



- Then, in receiver side, information of received file(Encrypted File Name, Encrypted File Path, Encrypted File Size, Decrypted File Name, Decrypted File Path, Decrypted File size) and information of symmetric key and information of file verification using file signature and sender's public key would appear in File Transfer Information panel.
- In sender side, information of symmetric key would appear in File Transfer Information panel.

How to use

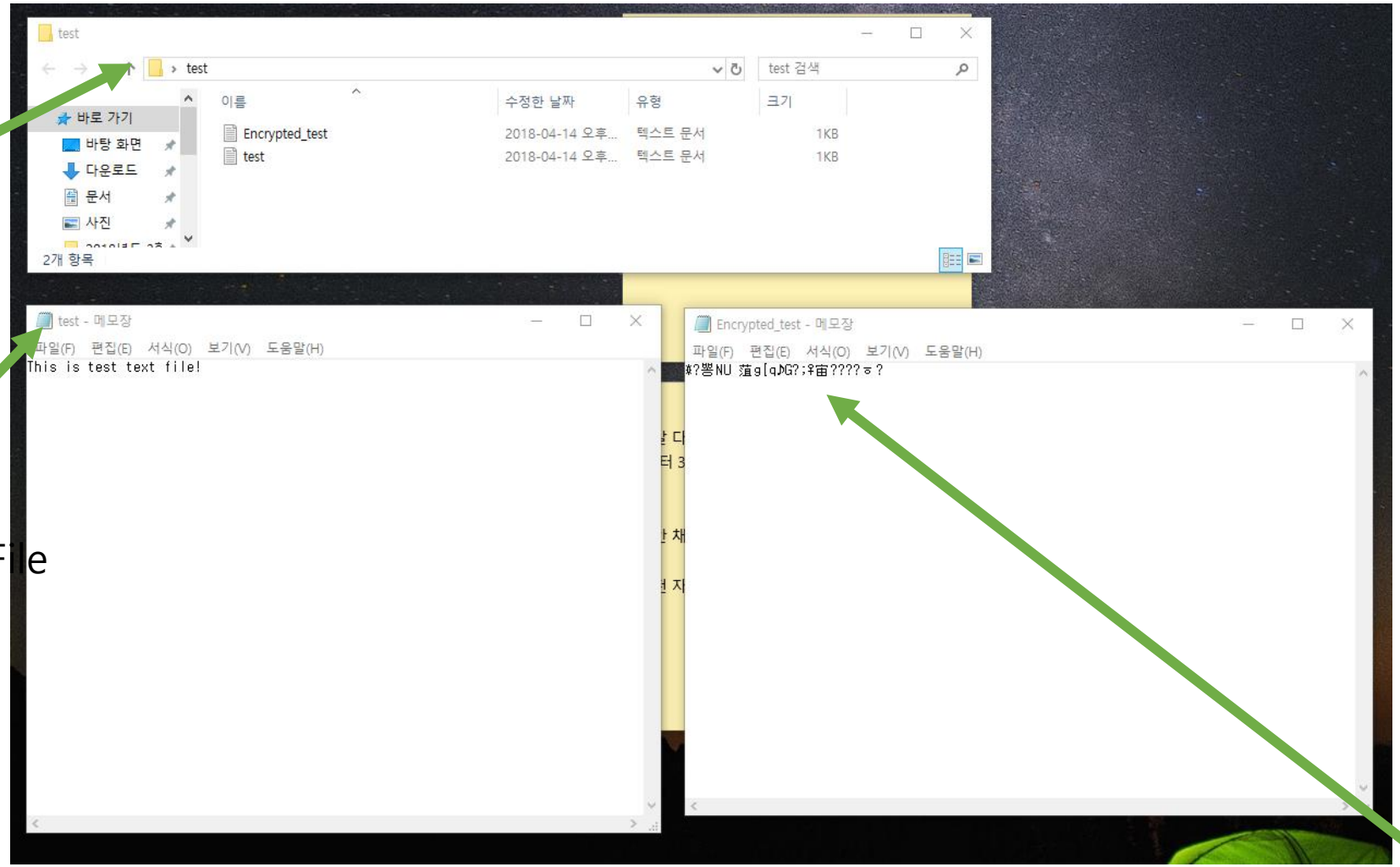
01

02

15. Send File

File Path to receive

Received decrypted File



Received encrypted file

How to use

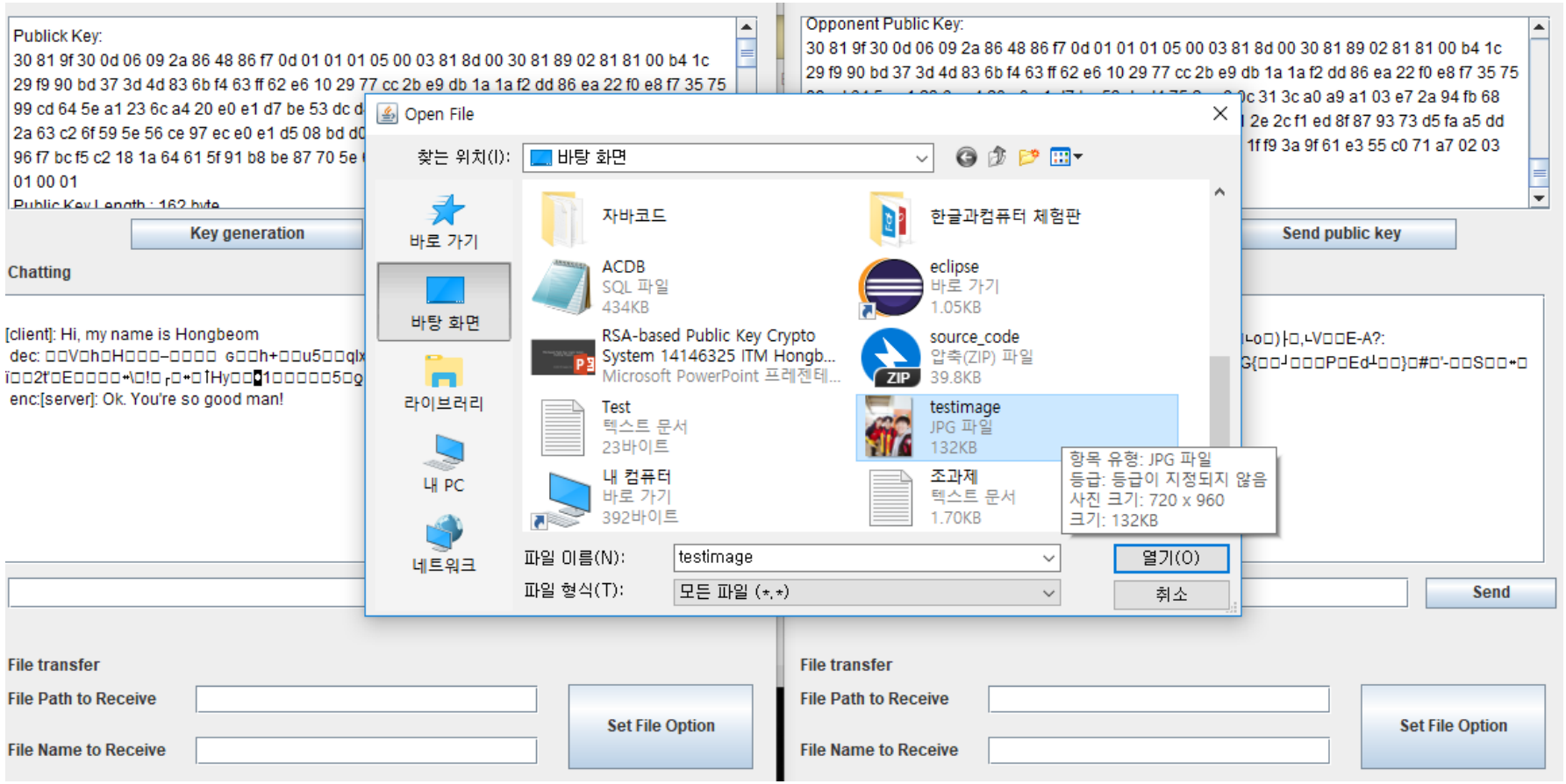
01

02

15. Send File – Default path and file name

Client Mode

Server Mode



How to use

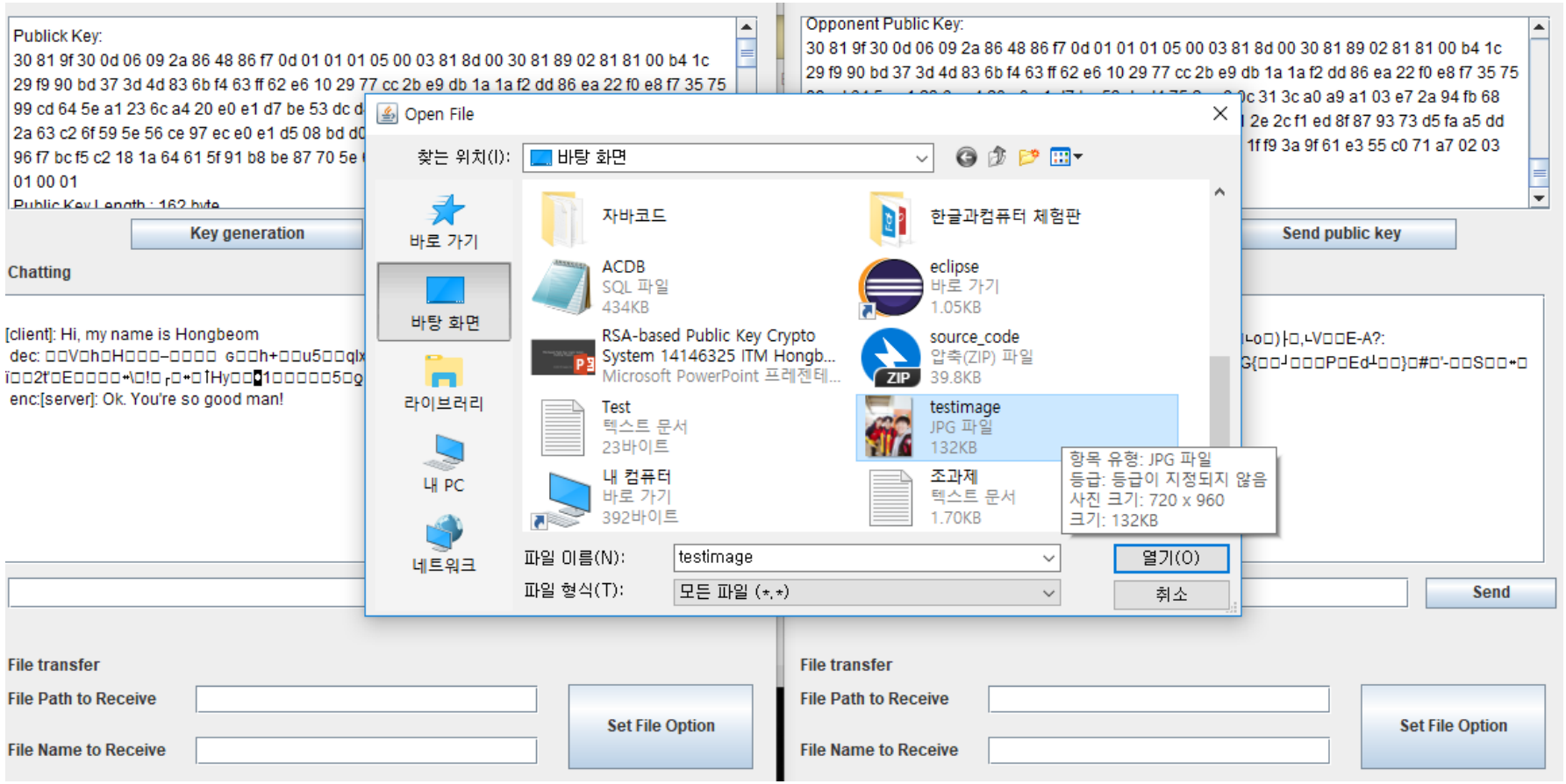
01

02

15. Send File – Default path and file name

Client Mode

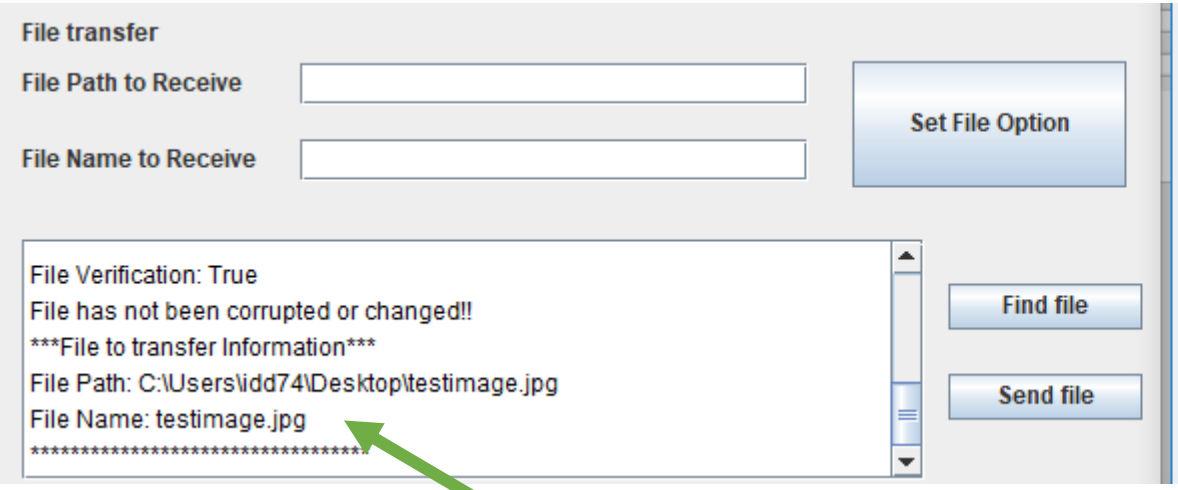
Server Mode



How to use

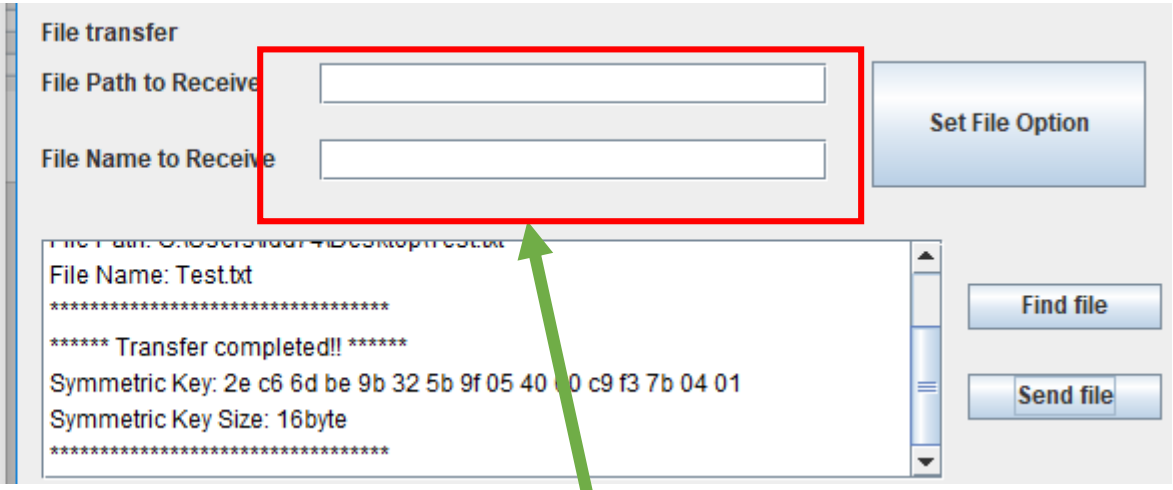
15. Send File – Default path and file name

Client Mode



Test image selected

Server Mode

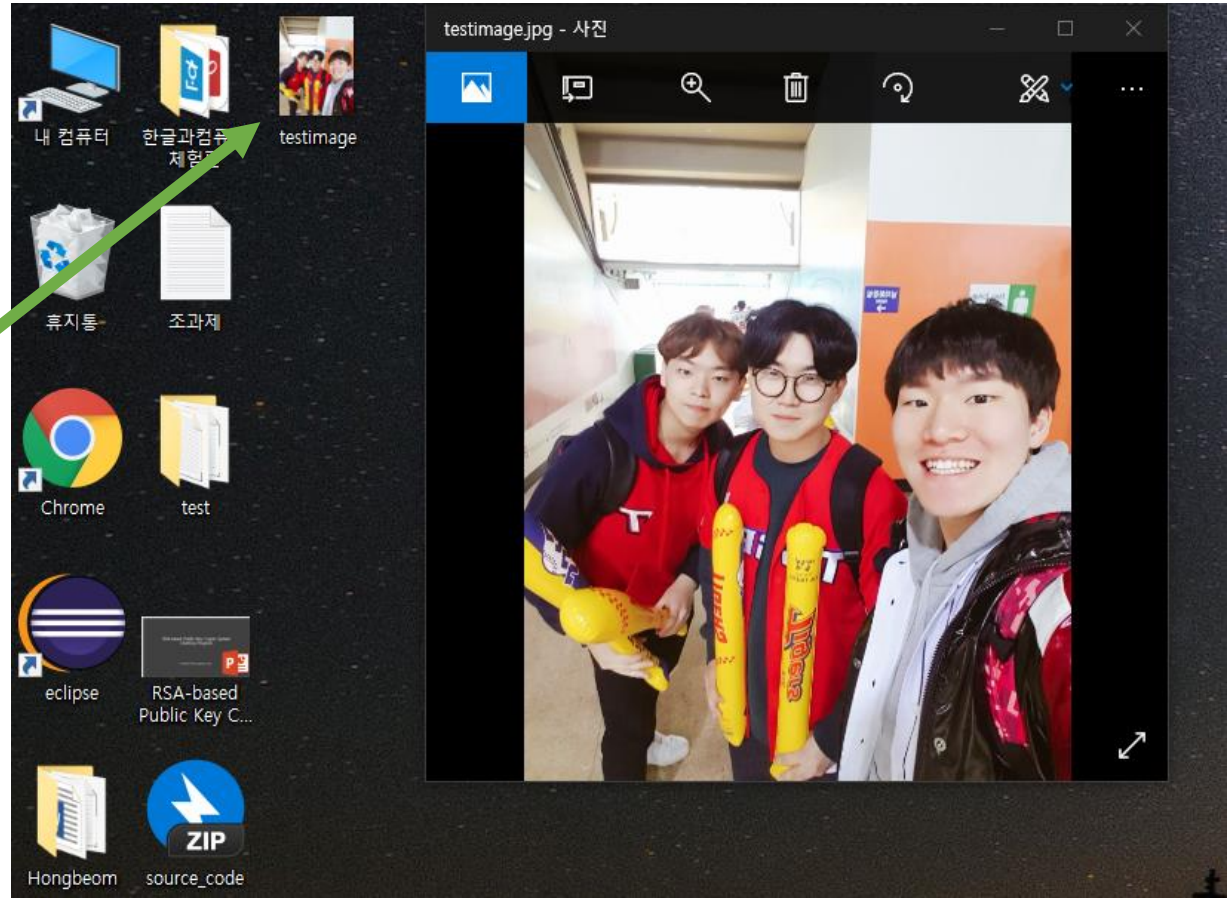


Server not specified file path and name of file to receive

How to use

15. Send File – Default path and file name

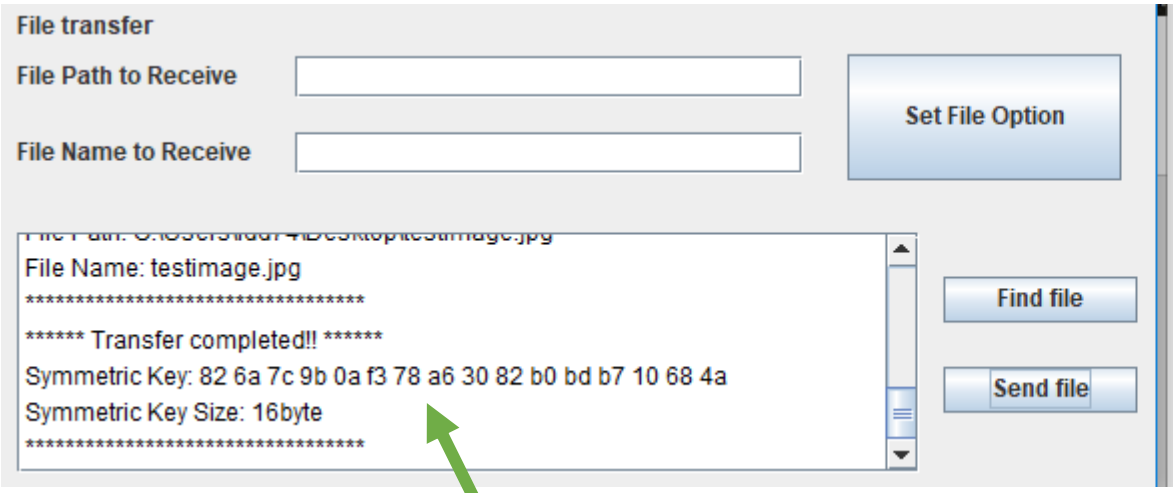
Test image selected



How to use

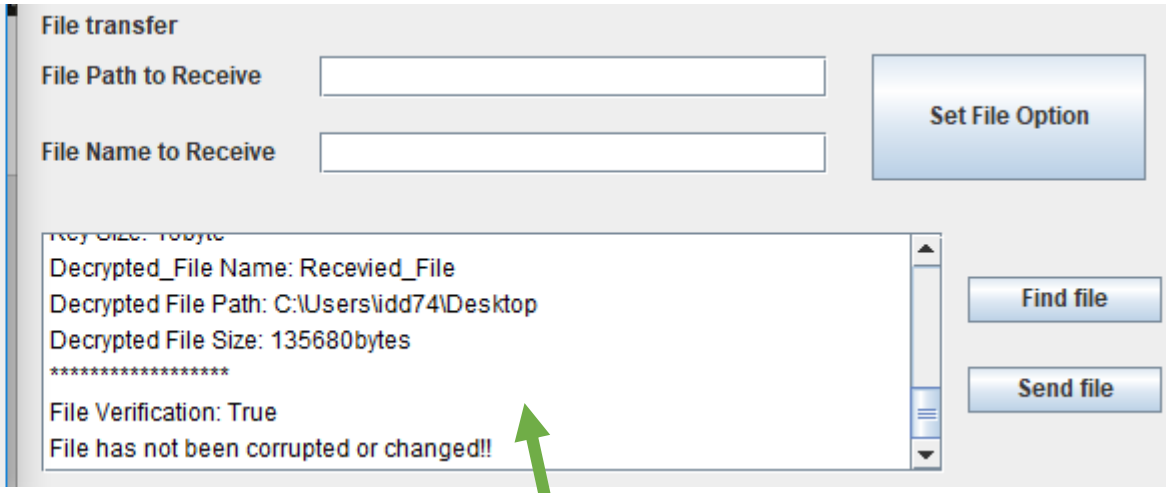
15. Send File – Default path and file name

Client Mode



Transfer completed

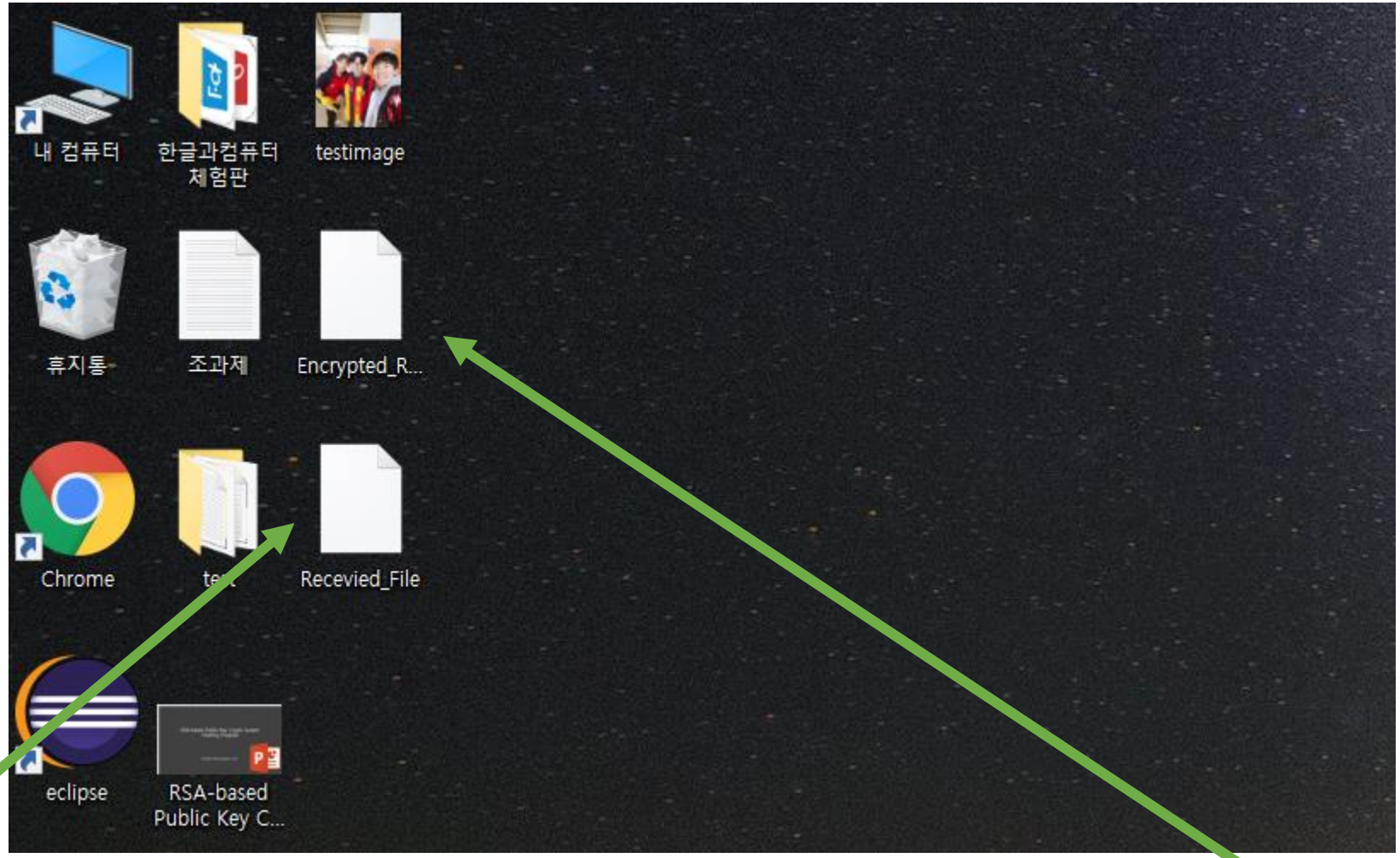
Server Mode



Information of received file

How to use

15. Send File – Default path and file name

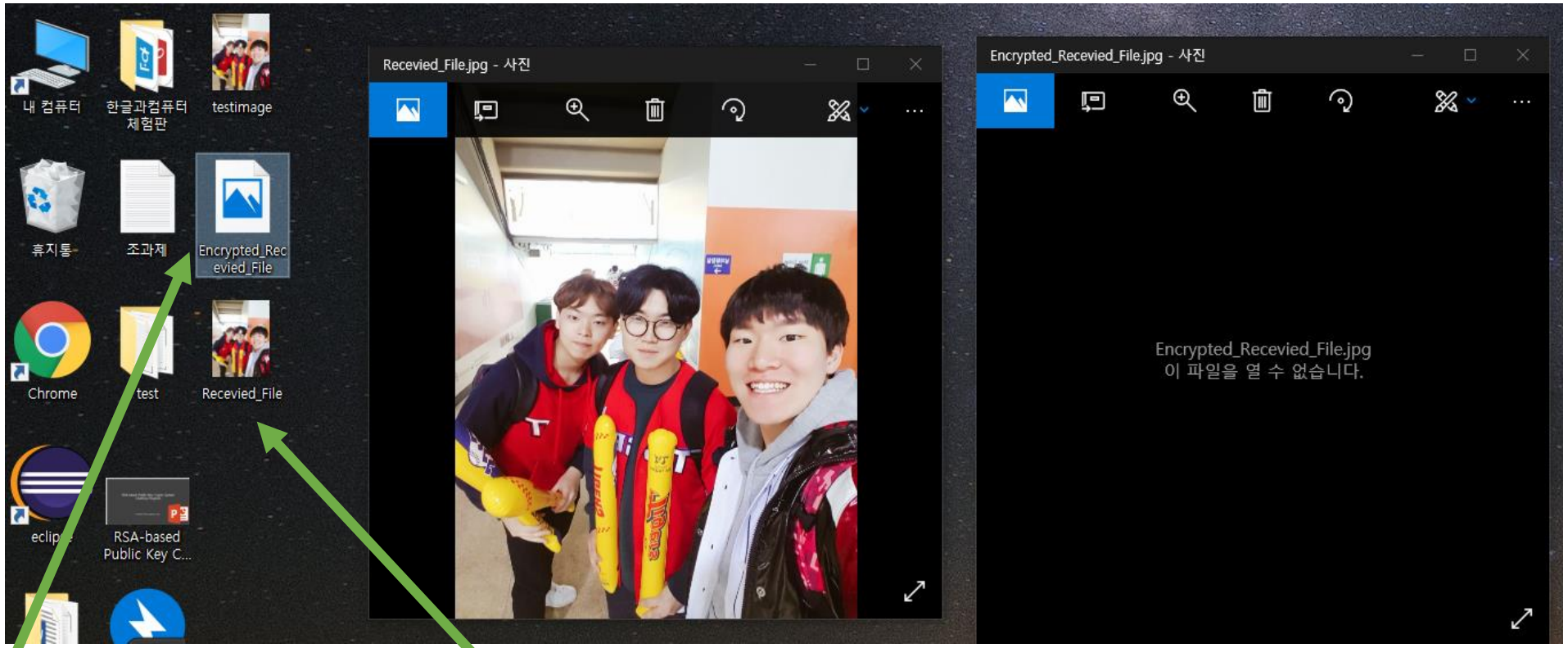


Received Decrpyted File

Received Encrypted File

How to use

15. Send File – Default path and file name



- Then, I changed the extension name and read the file

Received Decrpyted File

Received Encrypted File