Assignment #1 – Simple Public Key System

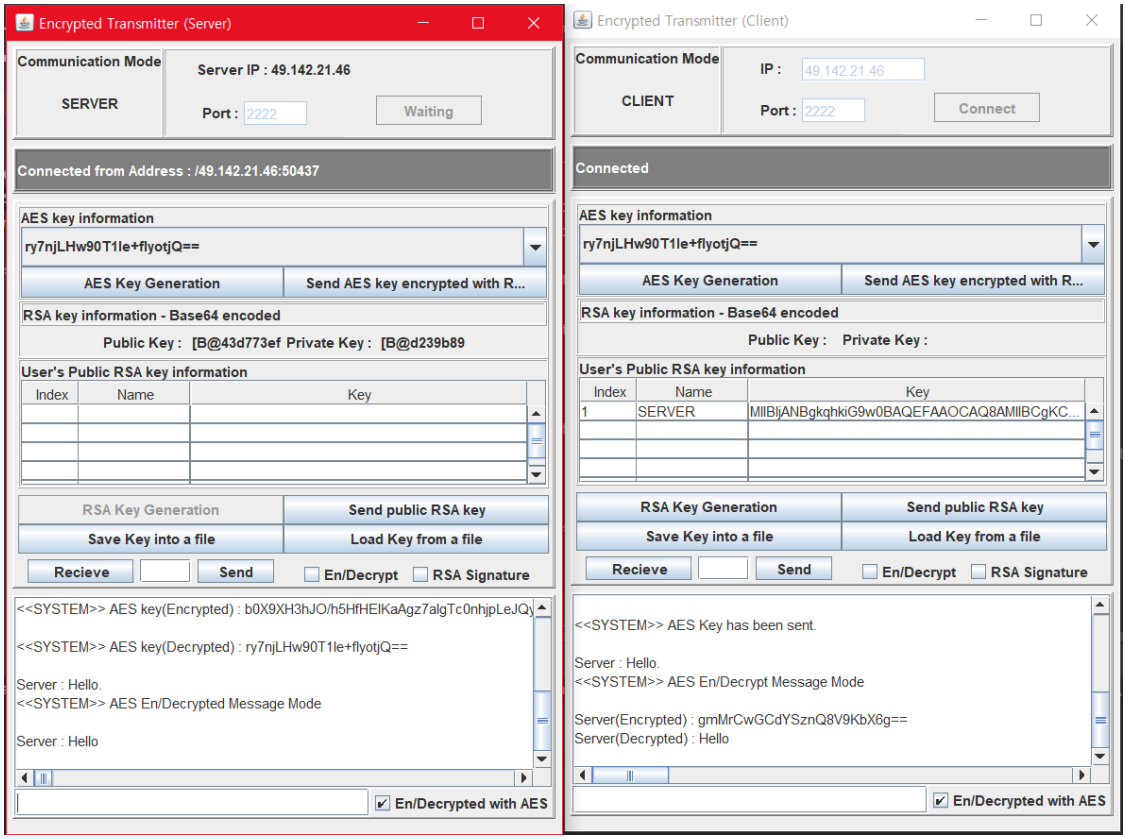
15146308 Junmin Kim

Source Code : I wrote down the description of the code in between in TransmitterServer.java I attached whole Project folder, you can run my Practice with Main.java on **administrator privileges.**

```
/*
* By JunMin Kim 15146308 ITM SeoulTech
* Code is 4 part.  Server GUI, Server method, Client GUI, Client method.
* Most part(97%) is same in TransmitterClient.java and
TransmitterServer.java
* I commented almost information on TransmitterServer.java and only
difference part is wrote in TransmitterClient.java
*
 */
```

3 java file : Main(only use for running) , TransmitterServer, TransmitterClient.

Code consists of big 4 part. Server GUI, Server method, Client GUI, Client method part.



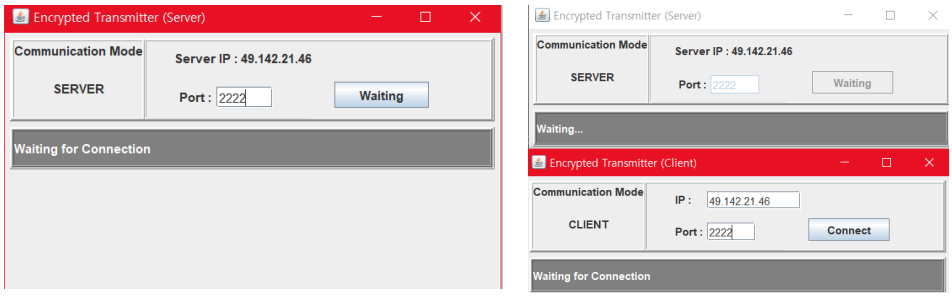
2. GUI : GUI consists of outside 3 Part : Connect State, Key information and Button related in, chatting and system message part.

>> Successful Task

1. Key Generation : RSA pair Key and AES symmetric Key
2. Save into/Load from file: RSA Pair Key can be saved and loaded each.
3. Public Key Exchange : Every method is for both. Server and Client can share Public Key.
4. Encrypted Chat : After Sharing AES Key encrypted with RSA Public Key, S/C can use Encrypt Mode. The recipient can display the received ciphertext and its decryption result.
5. Encrypted file transfer : S/C can encrypt, transfer and decrypt files.
6. Digital Signature : To guarantee the integrity, a signature can be attached with file to be transmitted.

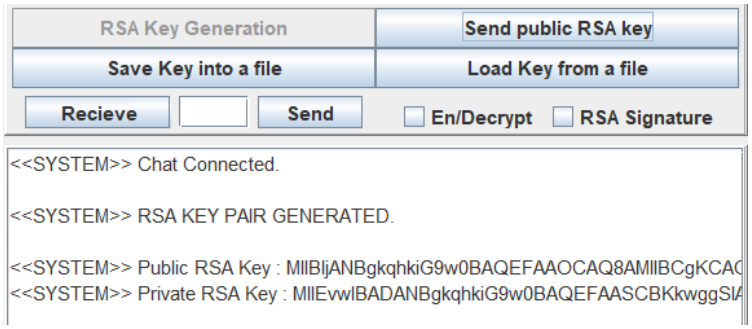
No Extended assignment.

0. Connect Panel and Connect State Panel.

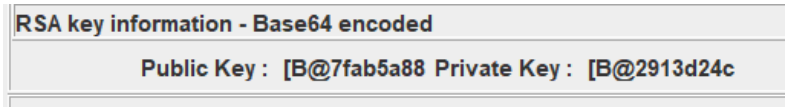


Show Server IP and can choose port number. Client can access with IP and Port.

1. Key Generation

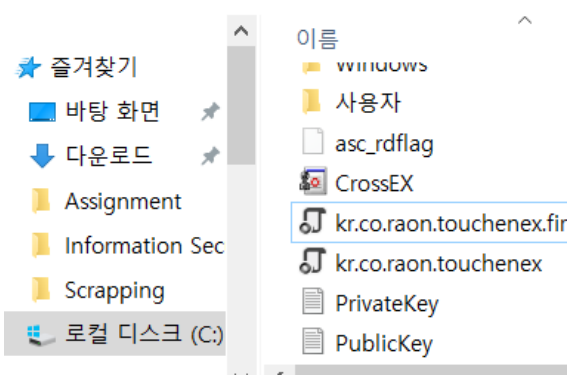


RSA Key Generation Button : generate RSA pair. Shown as Base 64 encoded.



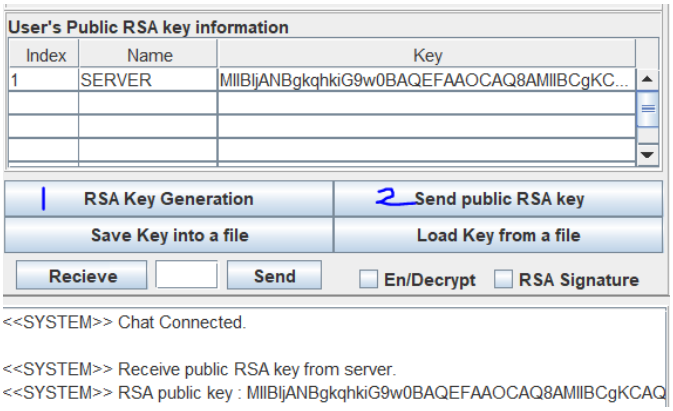
So, same key can have different value. But Key Object is not changed.

2. Save into / Load from File



File must be saved in C drive and loaded in C drive. File name must be PrivateKey.txt and PublicKey.txt to load.

3. Send Public RSA key

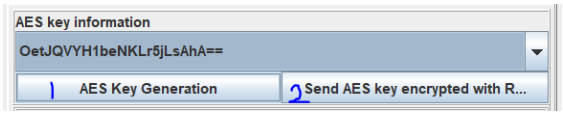



```
out.writeUTF(WARNINGRSA);
out.writeUTF(puKey);
```

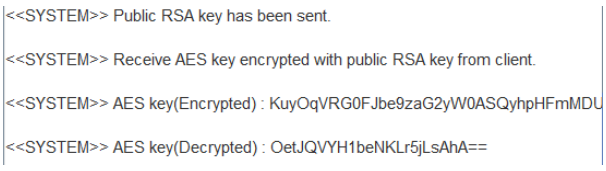
When user send any keys, it is transferred as String type, so in order to distinguish it with message, Warning message is sent first. Then client socket can notice that the key is transferred.

RSA key From Server is saved in Client Table. Client also can generate RSA Key pair.

Generate AES Key

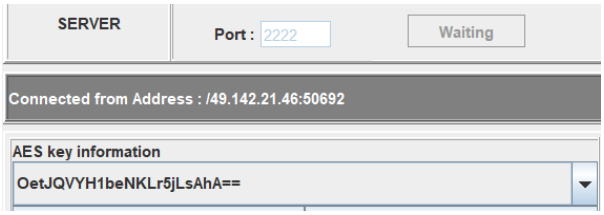


AES keys are saved in JCombobox and User can select and send it to Server/Client. Before Send AES key, It must be encrypted for security problems. It is encrypted using RSA Public key received from Sever/Client.



In left figure, Client send AES key and Server receive it and decrypted won

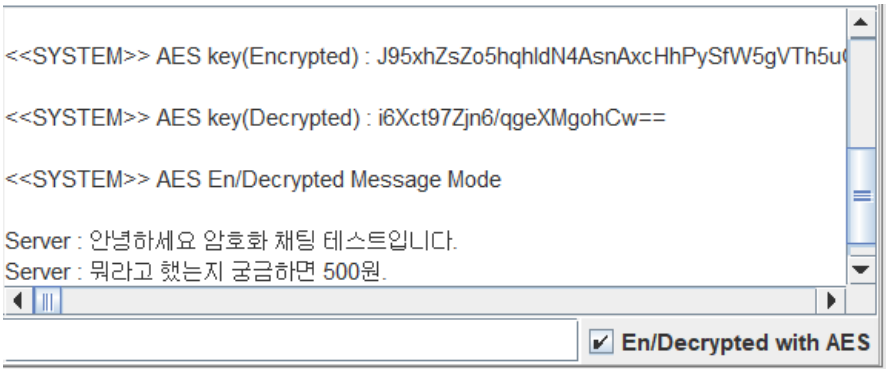
RSA private key. Then it is saved in Server's AES key JComboBox.



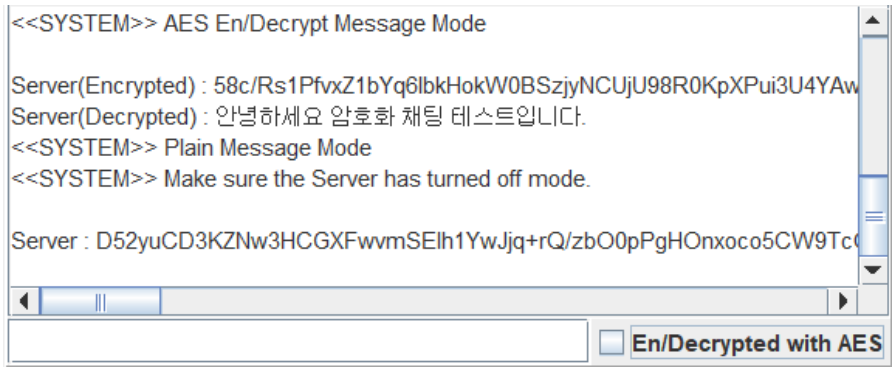
Server and Client can make a lot of AES keys and save in their JCombobox. That makes it safer.

4. Encrypted Chat

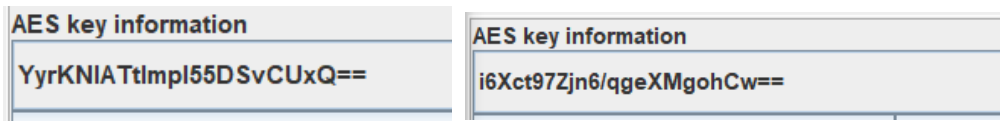
Server:



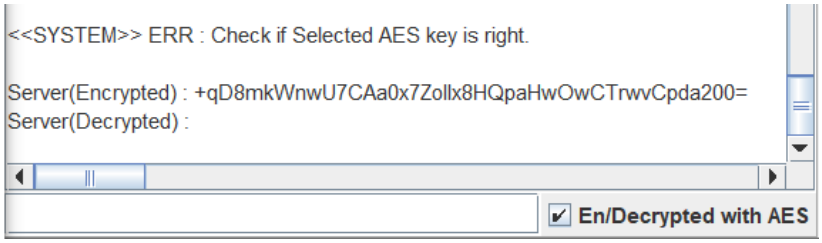
Client :



If Encrypted Message CheckBox is checked, User can send Encrypted message and decrypted automatically. But when only sender checked it, receiver cannot know the message.



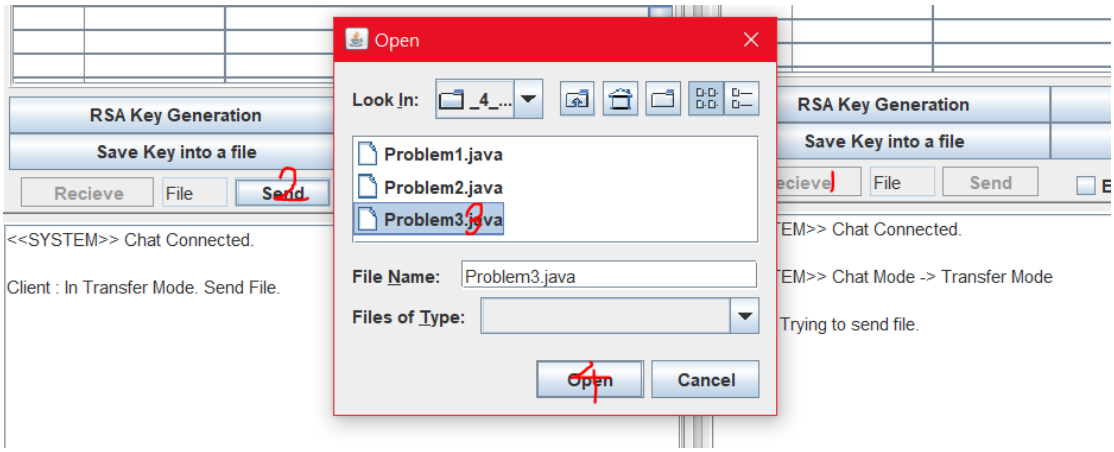
Also if user select different symmetric key,



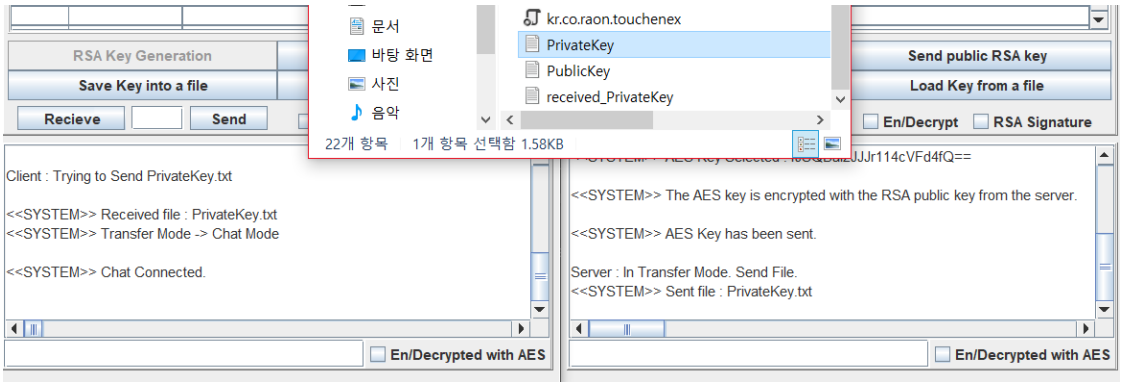
Receiver cannot decrypted message and get ERR System message.

5. Encrypted file transfer

First of all, I use only one socket and one stream, Receiver who receive files must click the Receive Button first. Then, chatting streaming is stopping to transfer files. After that, the sender can click Send Button and choose the file which is sent.



When Server click Receive Button, requestion message is sent and client cannot use Receive Button and Chatting until finish sending file. The sender can send the file he or she chose and click the Open Button. Then transferring is completed with the SYSTEM



I sent 'PrivateKey.txt', you can see received_PrivateKey.txt is transferred. After sending file, system go back to Chat Stream.Server and Client can use Encrypted file mode or RSA signature mode. There is CheckBox for each.

```
void receiveFile() {

    byte[] buffer = new byte[1024];
    if (encryptFileCheck.isSelected()) {
        buffer = new byte[1040];     // en
    }
```

Encrypting a file increases the data in each buffer by 16 bytes. So I choose the way that when receiving data, the size of buffer was increased by 16 bytes and save it to a file as long as the actual data after decryption.

```
data += 1;
if(data == sendingcount) break;
```

```
String stringAESKey = aESKeyBox.getSelectedItem().toString();
SecretKeySpec originalKey = new SecretKeySpec(stringAESKey.getBytes( charse
byte[] t = encryptFile(buffer, originalKey,  mode: "AES/ECB/PKCS5Padding");
outf.write(t);
outf.writeInt(length);  //send each buffer's length for letting receiver k
outf.flush();
```

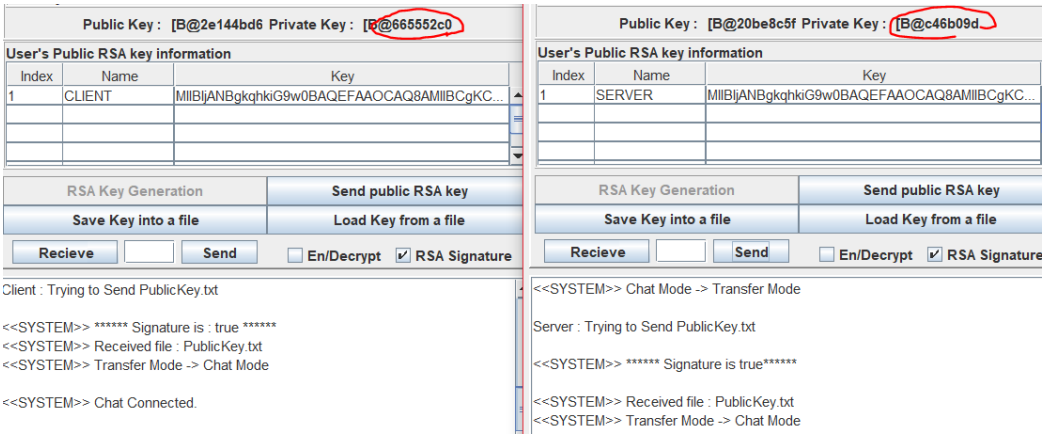
I also send real length of the data. So after decrypt the file received, we can choose only original data which is sent from sender. In addition, I also send a sensing count to check how many buffers user send. The transferred file is stored in the default directory path.

```
outf.writeLong( v: (file.length()/1024)+1);
```

so if the variable "data" equals to sendingcount, receiving is finished. And whole phase of transferring file is done.

6. Digital Signature

To use digital signature, the sender must need RSA private key and share RSA public key to the other. Because the signature is encrypted with private RSA key and sent. Then receiver can check if the signature is valid with public RSA key. This is also file transferring, so administrator privileges is needed.



I could not proceed extend assignment. Again, I commented in TransmitterServer.java and I use standard JAVA API and I wrote every code except the code from this course ppt.