CYBR372 Assignment 2

Part 1

The first part of Part 1 is that the Client and the Server both generate public and private keys. This is shown in the program as it outputs the public keys for both the Client and Server.

Client:

```
$ java EchoClient
Client Public key is MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAgQP2NYaCFGdnVcDjFBkjlQAA+QBIDG++ExaZ8a3kdETrP2uUrkgy3VnCzQop4M4BqHJYy
r91uJSqRDhenADnRms7368X56DpTTDSGKyekVTiB4d/1QFogbCaE5gViWTBgpHY1v/c6LcsZTgpaIaBTtlfjyllGr2CpZHjK6Ej/3h3s23vfyK3mEnFtxVl99TBtc+Ot7WxEoH
9vV834oxPryMSYcTmQLP5ZTBO1uYhB4ceyO9emtDx+4OaXJL4vfx/P8Y5KKgtV9NRiTqQZL6+UZjyCDn3QAgbl26NGdp3t+y8wNFlqeKk5G6pkGFDI5mcKjZKcfv+1umWXCEP2
wuv6wIDAQAB
```

The Client generates a key Pair and then prints out the Public Key

Server:

```
$ java EchoServer
Public key is MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArhq+x/OlpFjpMXjIPf6Hj9wQJSu2zBmZKftz3ZtgVMwNCc5lroNT4sw4OOOr+xE/69sZRePuIpvW
iASXXTr5gUHNZVf9WLzfWOI5oUS5W2+RD5dYu4SrqSKwEb77hPgZTLiUDzS+7zdpp+7AK6waAUTbqAyrD3a28OQakygwuSRUa3zkZQn6X9xlIXkI4H/3mAzVYw4o4mBvJHLrkn
R/Ye8UURZGk4pmRhInT61zcop+XAquZEA5ixw8pfWKnwt1Pgaf1out7ovcN8+X8ofbVxGPsu+brBbVEnGKdylV1Tw5UlTo2PPRrtZJ4Hmk4aviI3YqySctb786K38yBdhS7QID
AQAB
```

The Server generates a key pair and then prints out the Public Key.

The next part is the reading in of the public keys back in. The server needs to read in the Client's public key and the client needs to read in the server's public key. The server needs to read the client key in first because otherwise the client will start trying to send the message to the server and it won't be up and running yet.

The server:

```
Please enter the clients's public key below:
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAgQP2NYaCFGdnVcDjFBkjlQAA+QBIDG++ExaZ8a3kdETrP2uUrkgy3VnCzQop4M4BqHJYyr91uJSqRDhenADnRms736
8X56DpTTDSGKyekVTiB4d/1QFogbCaE5gViWTBgpHY1v/c6LcsZTgpaIaBTtlfjyllGr2CpZHjK6Ej/3h3s23vfyK3mEnFtxVl99TBtc+Ot7WxEoH9vV834oxPryMSYcTmQLP5
ZTBO1uYhB4ceyO9emtDx+4OaXJL4vfx/P8Y5KKgtV9NRiTqQZL6+UZjyCDn3QAgbl26NGdp3t+y8wNFlqeKk5G6pkGFDI5mcKjZKcfv+1umWXCEP2wuv6wIDAQAB
Waiting to complete Exchange
```

The server asks for the client's public key and once it has been entered, it waits for the client to start sending messages.

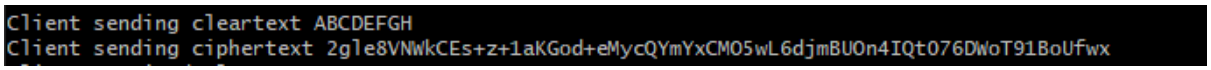
The client:

```
Please enter the servers's public key below:
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArhq+x/OlpFjpMXjIPf6Hj9wQJSu2zBmZKftz3ZtgVMwNCc5lroNT4sw4OOOr+xE/69sZRePuIpvWiASXXTr5gUHNZV
f9WLzfWOI5oUS5W2+RD5dYu4SrqSKwEb77hPgZTLiUDzS+7zdpp+7AK6waAUTbqAyrD3a28OQakygwuSRUa3zkZQn6X9xlIXkI4H/3mAzVYw4o4mBvJHLrknR/Ye8UURZGk4pm
RhInT61zcop+XAquZEA5ixw8pfWKnwt1Pgaf1out7ovcN8+X8ofbVxGPsu+brBbVEnGKdylV1Tw5UlTo2PPRrtZJ4Hmk4aviI3YqySctb786K38yBdhS7QIDAQAB
Keys exchanged
```

The client asks for the server's public key and should be entered after the server has been given the client's public key. Once the server's public key has been entered, the keys have been exchanged and the client can start to send messages to the server.

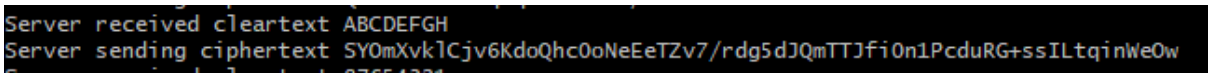
The client then needs to encrypt and sign messages to send them to the server where the messages will then to be decrypted and authenticated.

Client:



The client encrypts the message provided using RSA/ECB/PKCS1Padding to encrypt the message. The cipher text is then printed out to show that it has been encrypted. It has been encrypted with the server's public key. The client then signs the message with SHA256withRSA using the client's private key and both the encrypted message and the signature are sent to the server.

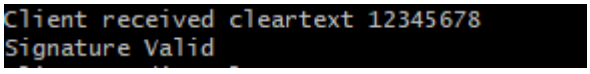
Server:



The server decrypts the message that has been sent and prints what the message that it received was. It uses the server's private key to decrypt. It also uses the client's public key to authenticate that the message came from the client and prints that the signature is valid if it has been correctly authenticated.

Once the server has correctly decrypted and authenticated the message, it sends it back to the client. It encrypts the message with the client's public key because signing the message with the server's private key. The server prints out the encrypted message before sending the encrypted message and the signature are sent back to the client.

Client:



The client decrypts the message that has been sent from the server using the client's private key. It prints out the message that it has received. It also uses the server's public key to authenticate that the message came from the server and prints that the signature is valid if it has been correctly authenticated.

Error Messages:

Both the client and the server print out human-readable error messages so that the error messages are readable to a human. There are a number of different error messages depending on what has

gone wrong. It tells the user if they are using an incorrect key, start the client before the server and more.

Running of Part 1:

The client and the server can be started in either order as they both generate their key pairs before waiting from the public key from the other. No arguments are needed to run the program. The server needs to be given the client's public key before the client receives the server and once both program receive the other's public key, messages can start to be sent.

Design Decisions for Part 1:

One design decision was how to read in the respective keys and how to get the client key to the server and vice versa. I made the decision to read the keys in using System.in. This means that the user has to copy the keys so that they can be read in.

From there, the code follows the practice for encrypting and sending messages. The code encrypts a message before signing it and then sending it. And then it decrypts and then authenticates the signature to verify that the message was correct.

The code prints human-readable messages to the user so that a user can understand what has gone wrong.