Computer Security

CS 5352

Professor L. Longpré

Assignment 2

# Introduction

The purpose of this assignment is to gain experience in implementing a cryptographic protocol that involves key exchange, digital signatures and encryption.

# General description

In this assignment, you are asked to implement a protocol similar to the SSL or the SSH protocols. The protocol involves certificates, key exchange, digital signatures, encryption and decryption. We will use both public key and symmetric encryptions. This assignment can be done either individually, or in teams of 2. If done in a team of 2, each student will need to implement the code on their own computer and submit their own copy of the report. Submit a report with the parts described below. If the part ask for files, include the file names in the appropriate part of the report and include all the files in a .zip folder.

# Part 1: Creating RSA cryptographic keys

The first step was to simply add a Scanner to read the user’s input for the file names. The following images show snippets of the file CreatePemKeys.java:

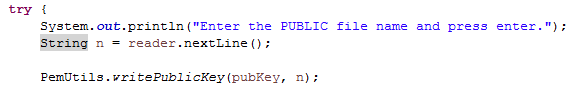


Figure 1

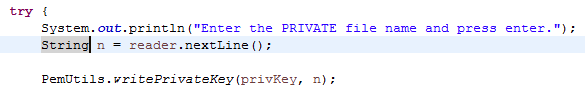


Figure 2

There were four pairs of RSA keys generated; two pairs for the server side and two pairs for the client. Each pair consists of a public key and a private key. So, the server and the client each get two private keys and two public keys. One of the public keys is to encrypt and the other to verify signatures. One of the private keys is used to decrypt and the other to sign. The following image shows the key files generated.

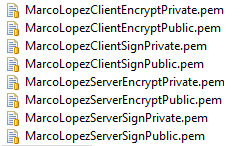


Figure 3

To test the keys, we used the file Encrypt.java with the file MarcoLopezClientEncryptPublic.pem for the RSA key and the string “Marco Lopez 4/1/2017 Testing encryption” to encrypt, which was put in a file named encryptedMessage.txt. The output for Encrypt.java was:

The plaintext is: Marco Lopez 4/1/2017 Testing encryption.

The encrypted string is: TjeZafZd4eGhCUXnCM5ibAq9IehCgQ5ywLNupvx0h+Ym27QIbMYABYPtP6C3S10UrihRDs6ojjaQFymZLWh3u36psyW5sYBuK3xiy2pkIV2zquNY9QKnaqcKA9I14p4RSDmTjXHSAvN7LZKb2ybG1vyys+4HNN8nD1J0RE7EZQs=

To decrypt the message, I used the file Decrypt.java with the file MarcoLopezClientEncryptPrivate.pem for the RSA key and the file encryptedMessage.txt to read from and decrypt. The output of Decrypt.java was:

The encrypted string is: TjeZafZd4eGhCUXnCM5ibAq9IehCgQ5ywLNupvx0h+Ym27QIbMYABYPtP6C3S10UrihRDs6ojjaQFymZLWh3u36psyW5sYBuK3xiy2pkIV2zquNY9QKnaqcKA9I14p4RSDmTjXHSAvN7LZKb2ybG1vyys+4HNN8nD1J0RE7EZQs=

The decrypted string is: Marco Lopez 4/1/2017 Testing encryption.

The next step was to test the signature keys. This was done through the file Sign.java with the string “Marco Lopez 4/1/2017 Testing signature” as the message to sign and MarcoLopezClientSignPrivate.pem as the key. After running the program, it places the signed string into a file called signature.txt. The signed message was:

kGdbWNz+FC0npySA5FDxzBcjP/oGIZanmWKF3HbFZITLzToBK9fA3aN7/S45w2B49MT4QP/rsX8yfL5INK5bszHuiC+5g/qUPQ7O3gIWQRYJeskivE98wcnlaRtkAlZAc3PpeJy9v2HpFfx5xbo4FEWH/7RRd4UTIfM9dwiVSvs=

Finally, I used Verify.java to test the signature. I used MarcoLopezClientSignPublic.pem as the key, "Marco Lopez 4/1/2017 Testing signature" as the signed message, and the file signature.txt to read from and verify. The output of Verify.java was:

Verifying the signature of: "Marco Lopez 4/1/2017 Testing signature"

Signature verification succeeded

* If we encode a byte array of length n with Base64 encoding, what is the length of the string derived from the encoding?

It will be of length ((n + 2)/3)\*4 using integer division.

Reference: <https://en.wikipedia.org/wiki/Base64>

# Part 2: Generating key certificates

Two certificates were generated; one for the client and one for the server. The certificates were generated through this website: <http://cs5339.cs.utep.edu/longpre/authority/CertificateRequest.html> which requires a public encryption key and a public signature key. The client certificate is in a file named MarcoLopezClientCertificate.txt which reads:

-----BEGIN INFORMATION-----

Date: 01-Apr-17

Name: Marco Lopez Client

-----END INFORMATION-----

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCm8JBgAUgfYuv1pu2DXk5b0ZiJ

eNY/ZDXMe2xHzGzdzuDrYzjcC6x65LWZy3OG1Tv/Sgx56YLcvDlR/CoQllZ4srE6

FweQFuyZDMa4ra9xarXWn5k7MpXCCCo+WQ1xh7u/nf/bvMadm9YR+nDhuONAsHOA

tSMzU96aOAOrEzVYZwIDAQAB

-----END PUBLIC KEY-----

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCRyUgztK/VOTKfYjpMxHX1/CSE

MJShMDnN/yBUklRodLkl8htxKigrqQ5jgrSDv0Yn38uB3NmO1BdM3qosGQfmZ1ln

pJcYhIY2vFvzWqQsmuY2gjXPJgcn81wWaiu7UIJNimFGHCbWeywroy/Zv8/rmQhK

mVC2DkcxsyPbRheybQIDAQAB

-----END PUBLIC KEY-----

-----BEGIN SIGNATURE-----

ZxlBOs14Askr+8Lw1FE5CEiD7s7OxH6D6+2p272J+i7z/DtvRZGdBVCn6MguR+poUyNPannkQAhgCkVsbe3iX72qwYAV5KFgR4WD99xdQ0r+Eg6ejFId3AyhnC4sCAUY/1Yw3xteORLDfVH2d2Obf6+a1U8+z9FjvuauEkt64U8=

-----END SIGNATURE-----

The server certificate is in the file MarcoLopezServerCertificate.txt which reads:

-----BEGIN INFORMATION-----

Date: 01-Apr-17

Name: Marco Lopez Server

-----END INFORMATION-----

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCpVMn/Iw/Q/6GnZGjQ/idPjHU+

7+aEhuKDQ1tC4G093NaxO4dOwpVjfnhSCgxB/zJDmVA1oZ6UzMd/arAn4w0ADplk

+rTcD1Z97EKmNvlXWbs5EpymrO9CvSu14f8n44sTNx8qVJRAYh+gE4EuTs3xf4ZD

TqRNjgmQ6pUcDNtzXwIDAQAB

-----END PUBLIC KEY-----

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCoJUfD1wdmqIDZyzqbyN/zXS2i

YdFzWxmGVgKohFqH9V0/dv1XbWj6iTcLwfN8v3lfH6ryQKzN6nGCv2fA39HbTToO

jNRGYVB8X4ZWaDvQpl5imim8/7J0qBIBJlCWMk3SGquzrNHQ3i6frS/lQXPpY83G

PhEeZhEQyhYh3PleBQIDAQAB

-----END PUBLIC KEY-----

-----BEGIN SIGNATURE-----

E8kMQah0x1q2XKbRwzN2UIYCw6p4+KxB2nzn5bMf5VBuAvYFBQFEhl57nyJWMAKNbcdyP5qPhNENJApN68EqFM+bNEoe1d9QcL2zG/X3Z/UItqmxcXhkKlwd9M6ZW6FJwDVABLt5TKLpm+8W3vL0Od0XfildGfvkXrGFkqm1rtw=

-----END SIGNATURE-----

# Part 3: Verify your certificates

To do this, I used the verifyCert.java to test both certificates. I made a slight modification to allow for user input asking for the certificate file name. When testing the client certificate MarcoLopezClientCertificate.txt with the program this is the output:

Enter the file name of the certificate to test and press enter.

MarcoLopezClientCertificate.txt

-----BEGIN INFORMATION-----

Date: 01-Apr-17

Name: Marco Lopez Client

-----END INFORMATION-----

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCm8JBgAUgfYuv1pu2DXk5b0ZiJ

eNY/ZDXMe2xHzGzdzuDrYzjcC6x65LWZy3OG1Tv/Sgx56YLcvDlR/CoQllZ4srE6

FweQFuyZDMa4ra9xarXWn5k7MpXCCCo+WQ1xh7u/nf/bvMadm9YR+nDhuONAsHOA

tSMzU96aOAOrEzVYZwIDAQAB

-----END PUBLIC KEY-----

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCRyUgztK/VOTKfYjpMxHX1/CSE

MJShMDnN/yBUklRodLkl8htxKigrqQ5jgrSDv0Yn38uB3NmO1BdM3qosGQfmZ1ln

pJcYhIY2vFvzWqQsmuY2gjXPJgcn81wWaiu7UIJNimFGHCbWeywroy/Zv8/rmQhK

mVC2DkcxsyPbRheybQIDAQAB

-----END PUBLIC KEY-----

Signature verification succeeded

The output for MarcoLopezServerCertificate.txt for the same program is:

Enter the file name of the certificate to test and press enter.

MarcoLopezServerCertificate.txt

-----BEGIN INFORMATION-----

Date: 01-Apr-17

Name: Marco Lopez Server

-----END INFORMATION-----

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCpVMn/Iw/Q/6GnZGjQ/idPjHU+

7+aEhuKDQ1tC4G093NaxO4dOwpVjfnhSCgxB/zJDmVA1oZ6UzMd/arAn4w0ADplk

+rTcD1Z97EKmNvlXWbs5EpymrO9CvSu14f8n44sTNx8qVJRAYh+gE4EuTs3xf4ZD

TqRNjgmQ6pUcDNtzXwIDAQAB

-----END PUBLIC KEY-----

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCoJUfD1wdmqIDZyzqbyN/zXS2i

YdFzWxmGVgKohFqH9V0/dv1XbWj6iTcLwfN8v3lfH6ryQKzN6nGCv2fA39HbTToO

jNRGYVB8X4ZWaDvQpl5imim8/7J0qBIBJlCWMk3SGquzrNHQ3i6frS/lQXPpY83G

PhEeZhEQyhYh3PleBQIDAQAB

-----END PUBLIC KEY-----

Signature verification succeeded

# Part 4: Communication

The file MultiEchoServer.java was used to echo any received message from a client. It can connect to multiple clients at once. The file EchoClient.java connects to an IP address specified by the user in port 8008.

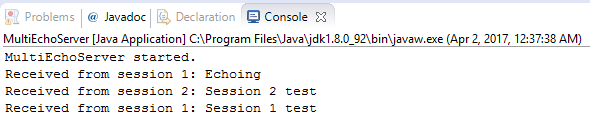
I could successfully run the server in my computer and have clients connect in a linux-based virtual machine. I connected two clients from my virtual machine

Figure 4 – Local computer as server

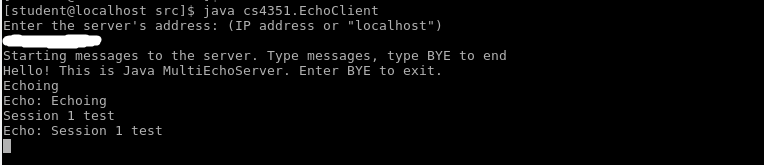


Figure 5 – Vitrual machine as client. Blurred IP for privacy purposes

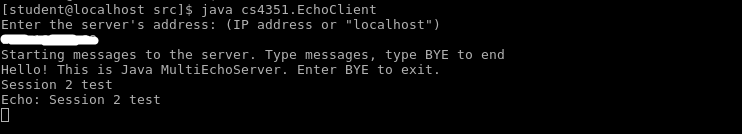


Figure 6 – Vitrual machine as client. Blurred IP for privacy purposes

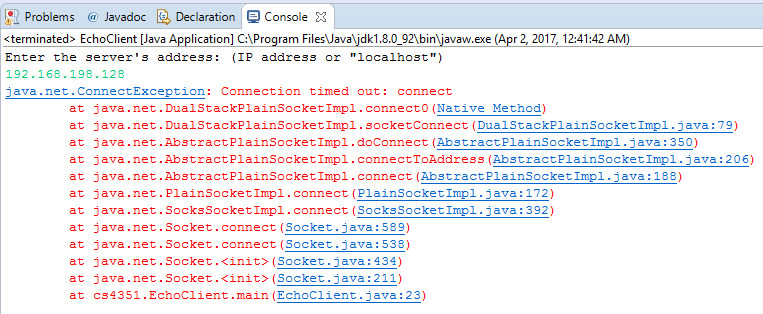
However, whe I tried to do it the other way around (having the virtual machine be the server) my local machine would not be able to connect due to a connection timeout. 

Figure 7 – local computer as client could not connect to virtual machine server.

Lastly, while I did not test this at UTEP, I did test with two different computers connected through UTEP vpn. The eclipse console shows one computer while the linux terminal shows the other computer. Both the server and the client functionality were tested through this vpn.

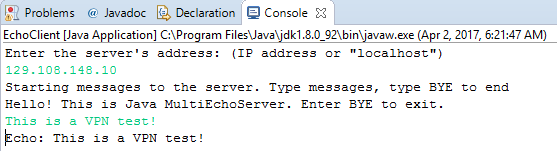


Figure 8 – Computer 1 as client



Figure 9 – Computer 2 as a server

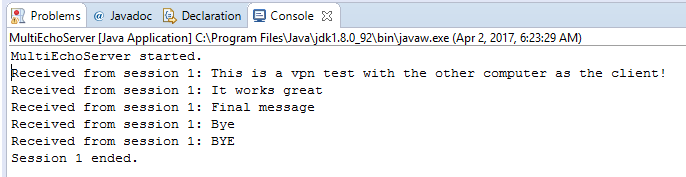


Figure 10 – Computer 1 as server

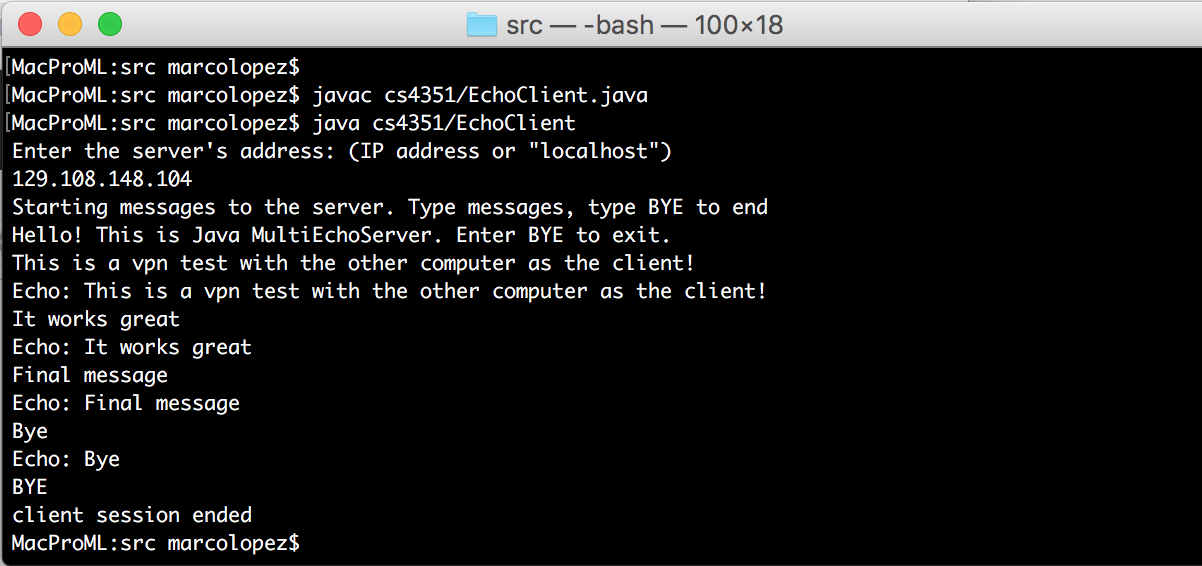


Figure 11 – Computer 2 as client

# Part 5: Encrypting communications through the socket

For this part, we had to make the server encrypt whatever it echoed to the client and have the client decrypt it when it received it. To do this, I created a new cipher for the server which would be used for encryption as opposed to decryption like the cipher that was already provided. I created the cipher with the same parameters as the previous one, and then sent its initialization vector (IV) to the client. The client then created a new decrypting cipher with the same parameters as before and initialize with the IV received from the server. This is illustrated in the following two images. I wasn’t sure if another secret key was necessary (meaning a new set of random bytes), but I tried with the same key and it worked. I’m not sure if this is not secure, some feedback would be appreciated on this.

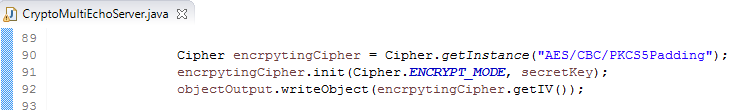


Figure 12 - Server encrypting cipher

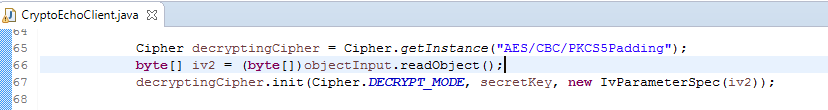


Figure 13 - Client decrypting cipher

Then, the server needs to encrypt whatever it is echoing to the client with the cipher I just described. It encrypts the echo string bytes using the cipher, and then outputs it to the client. Illustrated with the following image.

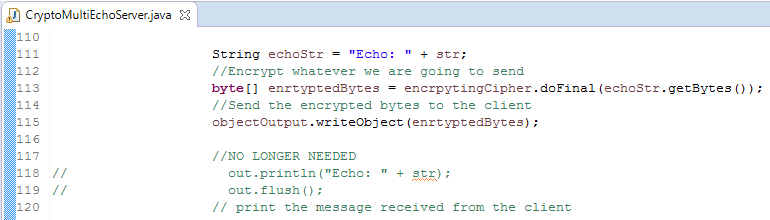


Figure 14 - Server encrypting echo message

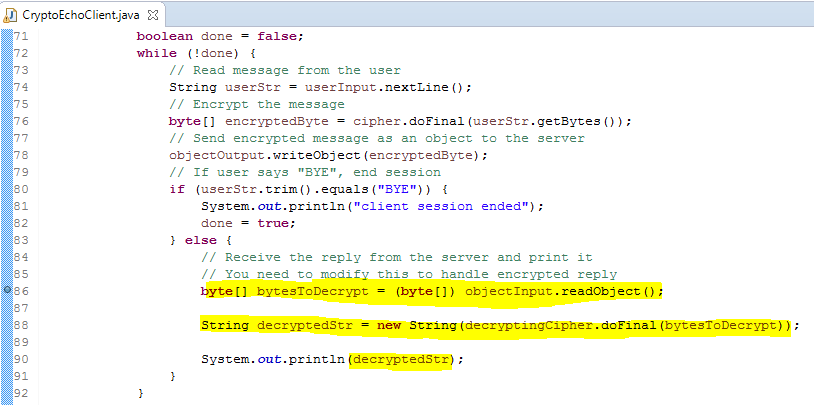
The client then has to decrypt that message with the cipher created for it. This is done in the while loop after the user inputs a line to send to the server. 

Figure 15 - Client decrypting echo message

# Part 6: The secure communication protocol

Still working on this part

# Part 7: Conclusion

1. How easy was this assignment? Explain any challenge you encountered and how you overcame the challenge.
   1. I had trouble in part 4. I wasn’t able to have my virtual machine be the server as it would time out from my local computer.
   2. I also had trouble when testing part 4 on two different computers. I didn’t realize that I had different random bytes for both computers, resulting in erroneous messaging/padding.
   3. Part 6 is also challenging, I am having trouble figuring out how to decrypt the server’s random bytes for key generation.
2. What difficulties did you encounter during this assignment, and how did you overcome the difficulties?

//Left blank as requested by instructor

1. If you worked in a group of 2, explain your group dynamic, including whether you worked physically together, whether you worked individually and communicated by e-mail, whether you separated the work, and generally, the contribution of each member of the group.
   1. So far, I have worked on this assignment on my own.
2. Explain what you learned in this assignment.
   1. I learned how encryption works, base64 encoding insights, and how certificates work.
3. Do you have any suggestion on how to improve this assignment if we use it the next time we teach this course?
   1. This assignment was challenging, but very interesting. The only thing I could suggest is perhaps using different variable names that provide a bit more insight. Although perhaps it was intentional for us to figure out what they are for.