Dillon Van Buskirk & Sandy Becerra

Applied Cryptography

Dr. Qinghua Li

31/03/2016

For this lab assignment, students were asked to set up a secure communication between a server and client(Bob and Alice). This secure communication was to follow the description given to students by Dr. Li and students were able to work in pairs but was not required.

To begin the assignment, we started by gaining a full understanding of the packages available to us within java. After having understood what javax.crypto and java.security could do for us, we proceeded to writing code in a single file. First we tested our learned knowledge by writing simple code that would encrypt a message and decrypt it all within the same file. After getting the message to decrypt and encrypt we tested our knowledge of the Signature class by signing the message and making sure we were able to verify it. Again, all of this was done within a single file to test our understanding. The same process was followed for Diffie-Hellman and HMAC testing.

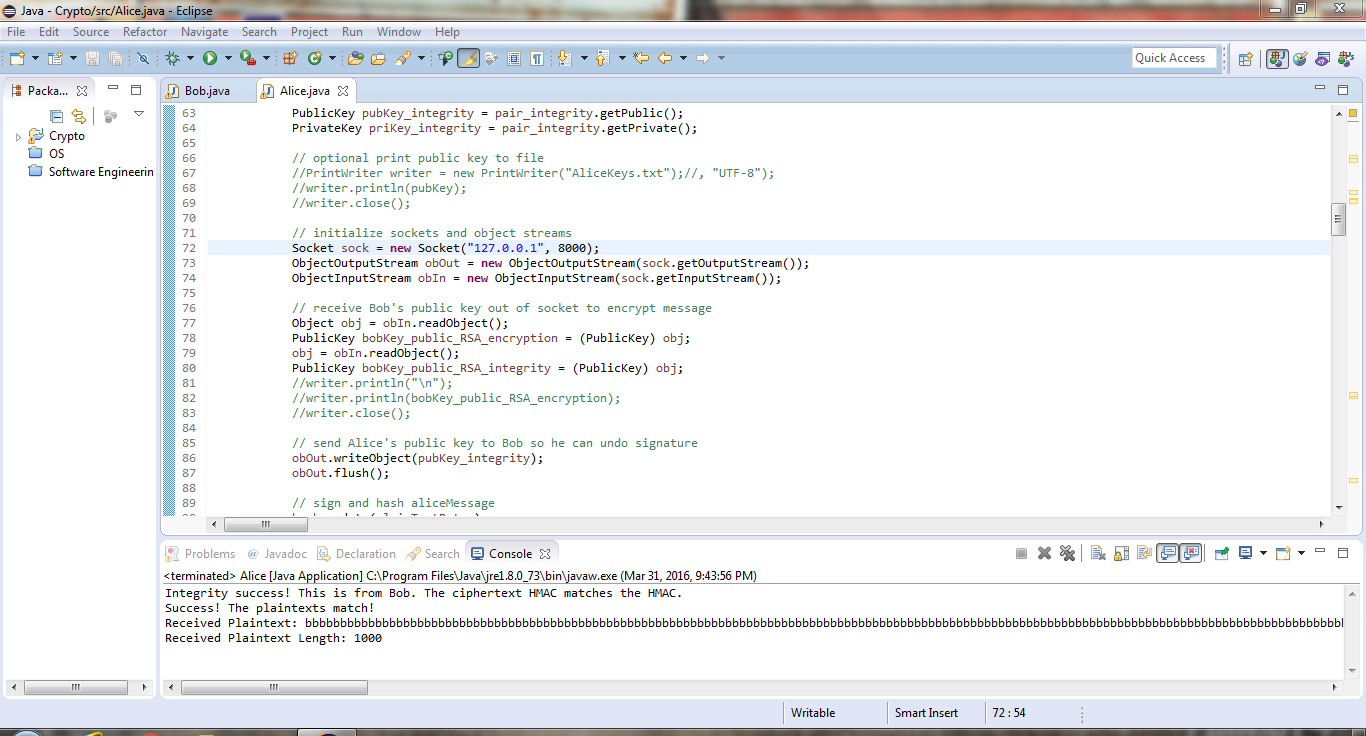
After we finished testing, we separated code into server(Bob) and client(Alice) and made sure that we were able to encrypt and decrypt as if in a single file. There were a few problems we ran into that had to be fixed. For example, we were getting “final block padding” errors which was due to using the wrong algorithm. Also, in order to get Diffie-Hellman to work properly the “Unlimited Strength Java package” had to be installed. This resulted because java only allows up to 128bits of key generation and for Diffie-Hellman and we were required at least 512bits.

Overall the structure of our code is very sequential to the assignment description. As programmers, sequential coding allowed for us to better debug when errors occurred. It also made it easier for the other person to continue coding from where the other left off in coding of the assignment.

As a final note our secure communication setup worked! Alice was able to send her public key through the socket and Bob was able to receive and use it (likewise for Alice) and a shared secret key was successfully created and sent. Aside, from the “final block padding” error, another issue we encountered was verifying the signature. Once the signed hash of the message was sent, verification issues were seen. After much review of our code we realized we had forgotten to update the signature variable with the hashbyte received while verifying.

Included in this report are images of what the server and client received. We printed out if the integrity test passed along with the message. Because this assignment was to simulate a secure communication between two people, the keys were not printed out, however, comments in the code can be looked at for further information regarding how keys were generated and passed.

*Below is an image of what the client(Alice) received.*



*Below is what our server(Bob) received*

