**Lab Report**

CIS 3319 Lab 2

Scott O’Hara

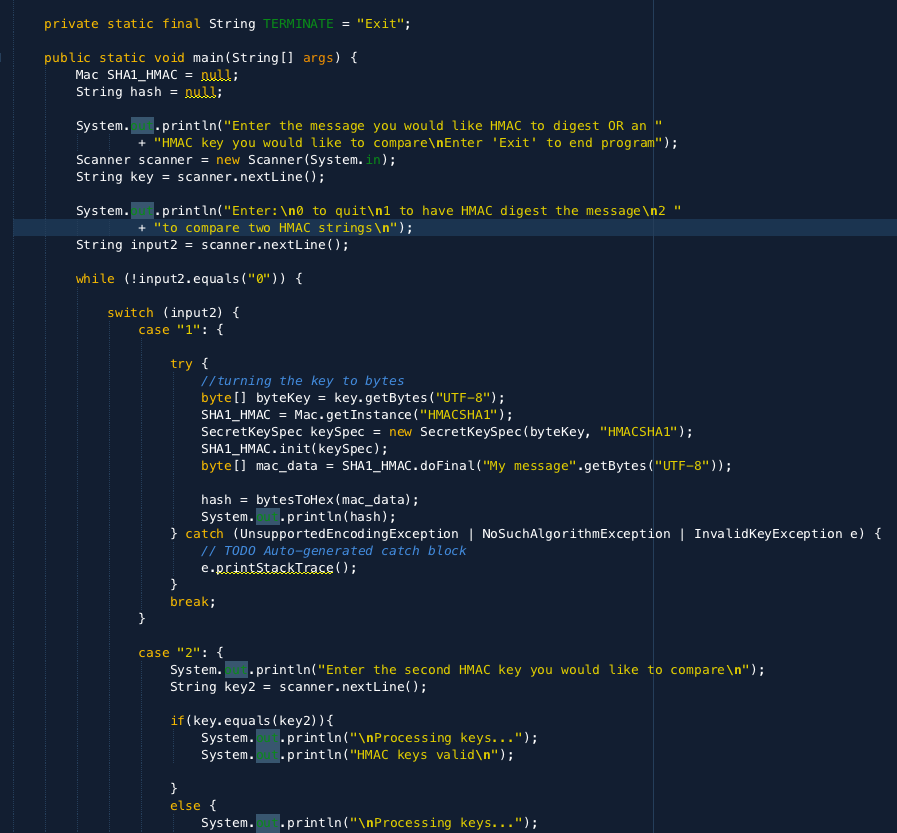
**Part 1)**

Through this report, I learned that HMAC doesn’t actually encrypt messages. Instead, the message (encrypted or not) should be sent alongside the HMAC hash. Parties with the secret key can generate the hash message again themselves, and if it is authentic, the received and produced hashes will match. Additionally, I discovered that HMAC has different variations that offer higher levels of encryption through increasing the HMAC string output length.

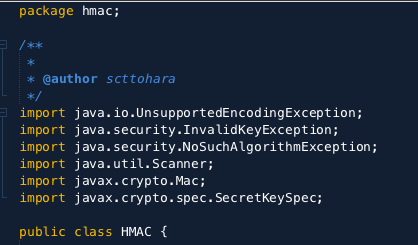
**Part 2)**

The steps to completing the assignment were quite simple this time. First, my partner and I decided to continue building our project in Java; Changing now would create unnecessary work and Java has a lot of libraries to help with these types of assignments. Following that, we were able to find example code of HMAC-SHA-512 quickly online. After editing to use HMAC-SHA-1 for simplicity, we additionally altered the code to receive user input for the key entry and HMAC comparison steps.

Below are 2 screenshots of code added to the example code found online, the bit found online is in the try statement. The catch statement, while originally from the example code found online, was edited for brevity.



**Libraries used:** (below)



**Problems encountered and how they were solved:** (below)

**Problem:** The only issue that continues to persist is creating a socket on the temple wifi, we assume it is a security protocol stopping the socket.

**Fix:** Utilize cell phone hot spot connection to create the socket in an environment with less protection protocols.

I didn’t encounter any other problems with the project, it was a well-defined assignment that was easily done because the base of the project was around built in lab 1.

**Note:**

Please refer to the Lab Report for Lab 1 for screenshots of the groupchat.java and crypto.java files.