Carson Wittwer: bbplayer@vt.edu
Jason Nelson: njason18@vt.edu
MH Charles-Etuk: mh6698@vt.edu
ECE 4564, Assignment 1

Objectives:

The objective of this assignment is to become familiar with socket programing between servers and clients, programming with APIs, and working with encryption. For this assignment, we communicated using low-level, socket connections between a client, bridge, and server, all running on separate Raspberry Pi 3+'s. The client interfaced with Twitter, using the Tweepy API, the bridge with IBM Watson, and the server with WolframAlpha.

The goal was to use the streaming feature of Tweepy to capture tweets in real time using a specific, designated hashtag. The tweet contains a question that would be sent from the client to a bridge that would use the IBM Watson API to speak the question. From there the question would pass to the server which would use WolframAlpha to answer the question. The answer would then be passed back to the bridge, and spoken using IBM Watson, then passed again back to the Client. The entire time, the question and answer would be encrypted using Fernet as well as including a checksum created with md5.

Design:

Client:

The client file is intended to be the portion to capture Tweets and build the initial payload. It first starts by ensuring that all the information necessary to access Twitter is available. The command line arguments are then parsed, and using that information, the socket is connection is built. From there a connection is created with the bridge. Before capturing the Twitter information, we generate the key for the Fernet encryption as well as authorize the Tweepy API. From there, the program moves into listening for Tweets containing the specified hashtag. Once that hashtag is received, the payload is built using the previously created key and md5 hash to create a checksum, then it is passed on to the bridge. When information is received, it is decrypted, printed, and then the process starts again. – Carson Wittwer

Bridge:

The bridge file serves as an intermediary step between the client and the server, using IBM Watson to convert both the question and answer text it receives into a .wav audio file, which it then plays. After all of the necessary setup (command line arguments, Watson authentication, etc.), the process begins by opening a socket for the Client device to connect with. After receiving and decrypting a package from the Client, the question string is sent to IBM Watson and returned as a 'Question.wav' file, which is immediately played and then deleted. The question package is then forwarded to the server via a different socket, before awaiting an answer package in return. The same exact process using IBM Watson is repeated to generate and play an 'Answer.wav' file, before finally passing the answer package back to the client, and awaiting another input. – Jason Nelson

Server:

The server is intended to be the part of the project that receives the encrypted payload from the bridge, decrypts it, and sends it to WolframAlpha's knowledge engine. It does this by first defining its port, the size of the backlog and that of the socket. Then, after parsing the arguments it creates the link

to WolframAlpha and builds the package it will send to Wolfram by way of a function that utilizes an MD5 hash subroutine and encryption-decryption subroutines. It then creates a socket and binds it to the host address so that it will be able to pick up the message the bridge eventually sends to it. Then it uses a loop to receive the question sent by the bridge, ask it to WolframAlpha, receive the answer, encrypt it, send it back to the bridge, and wait for the next question. - MH Charles-Etuk

Conclusions:

At the end of this assignment, we all greatly increased our knowledge on the main objectives of this assignment, as well as a better understanding of other smaller tasks that weren't explicit goals. Our group came in with very limited knowledge of Python overall, with only a few lines of code written between us all, and no knowledge of the APIs to be used.

We learned a great deal of socket programming from class as well as other outside resources, giving us a better understanding of the simple network protocol used to connect the Pi's. Through each member building their files and interfacing with their APIs, we became more familiar with reading documentation on the functionality of each one and how to implement them within code we develop. Encryption was also new to us, so implementing the Fernet algorithms and ensuring that the correct variable representations are passed was a clear learning point.

Through all these different learnings, we all built a better knowledge base of Python overall. From setting up working environments and working on multiple systems, to running the scripts and implementing the APIs.