ECE 4564 - Network Application Design

Assignment 1: forty-two
Team: 12
Members: Kulneet Singh (iceqman@vt.edu), Eamon Heaney (heaneye@vt.edu), Brandon Cheung (btc2019@vt.edu)

For project forty-two our objective is to set up two Raspberry Pi's, the first of the Pi's will act as the server and the second of the Pis will act as a client. Our Project uses TCP protocol to facilitate communication between the two. The client uses tweepy and the twitter api to receive a question posted with the hashtag "#ECE4564T12". Once the tweet is received its text is extracted. This extracted question the client then encrypts the message using Fernet, and generates a checksum of the encrypted data before packaging the encrypted question, the checksum, and the encryption key. This pickled package is then sent to the server using the before mentioned protocol. The server then unpickels the package and checks the checksum. If the checksum is correct then the server proceeds to unpickle the question. If the checksum is incorrect then the server closes and sends a close signal to the client. After the question is unpickled the server will decrypt the question using Fernet and the provided key. The question is then sent to the IBM Watson API, which results in the question being spoken aloud. After the question is spoken aloud the server uses the Wolfram Alpha API to send the question to the Wolfram Alpha database. The database then returns the answer to that question to the server. The server then pickles then encrypts and pickles the question in the same manner as above before sending that package to the client. The client unpickles and decrypts the question; before sending the answer back to IBM Watson to be spoken aloud. During each of these steps checkpoints are being printed to the console that follow the scheme specified in the grading rubric.

The Twitter portion of this project uses Tweepy to access the Twitter API. It is implemented as a class that inherits from the Tweepy StreamListen() class. This class imports the Twitter API keys for authentication, then opens a stream and filters by the keyword "ECE4564T12." When it finds a tweet with that keyword, it strips out all instances of the hashtag to find the text, then sends that text to the server.

The encryption and serialization portion of the project is accomplished through the use of the Cryptography.Fernet, Pickle and Hashlib libraries. On the client side, when a question is received from the Twitter API, the program first generates a random encryption key through the use of the Fernet Libraries standard generate_key() function, then a fernet cryptography object is created and the generated key is set for that object, next the program encrypts the plaintext question into ciphertext and finally the checksum is computed using sha256 hashing via the standard python hashlib. The next thing that the program does is that it takes the key, the ciphertext and the checksum and places them into a tuple data structure to aggregate the required data, and finally that data structure is serialized into a byte stream using the python Pickle library. Then the program sends the byte stream via sockets to the server. On the server side, once the question byte stream is received from the client, the server first unpickles the byte stream back into a tuple data structure, then the server checks the hash against the ciphertext to ensure that the question cipher was not altered in transition, once that check is completed, then the server takes the key sent by the client and then decrypts the question into plain text, then it is

processed by the other parts of the project described in this report and an answer is determined. The answer is then encrypted using the same key that was sent by the client, so that the client is able to decrypt the answer. The answer cipher text hash is also computed, then both the answer ciphertext and the corresponding hash is tupled and pickled before it is sent back to the client. The client then receives the pickled answer payload. Then the client correspondingly unpickles the answer payload into a tuple and then the answer ciphertext is hashed and compared to the checksum sent by the server to ensure that the answer was not altered in transmission.

IBM Watson was used to perform a text-to-speech conversion and save the audio data into an .mp3 file. This was done in both the client and the server. IBM Watson was accessed through its stored url and dev keys. These keys are located in the ClientKeys.py file and the ServerKeys.py file. IBm Watson takes in a string as it's text and returns an audio stream with the spoken words. This stream was then stored in a .mp3 file. To play this stored file automatically the pygame.mixer was used.

For the Wolfram Alpha portion of the project, an access key was obtained from the Wolfram Alpha engine. This key was stored in ServerKeys.Py and was used through the Wolfram Alpha API to relay the question to the Wolfram Alpha database. The answer to this question was then extracted from the reply sent by Wolfram Alpha, and stored as text.

Our solution to this assignment is complete and meets all requirements set forth in the rubric. The project is able to retrieve and parse a tweet with the specified hashtag using the client, encrypt and package the message, transmit that package to the server, have the server unpackage and decrypt the package, speak the question using IBM Watson, send the question to Wolfram Alpha and receive an answer, encrypt and package that received answer, send that package to the client, have the client unpackage and decrypt the answer, have the client speak the answer using IBM Watson, and be ready to repeat the loop. All checkpoints are successfully printed to the terminals correctly. The project is completely successful as per the specifications set out in the assignment document and in the Rubric.