Team Project Report

Goal: Create encryption and decryption programs with a GUI that can take inputs of strings or a file, read the data in that string or file, output the data into an encrypted message, take in that message, and output the original data.

Project Start: 4 – 29

I began by creating the 4 lists of characters (lowercase, uppercase, numbers and special characters) from which to pull entries from. Then, I created a file reader in order to get an input for a message. I didn’t start with a normal string input since I wanted the program to pull from an existing file.

After starting, I looked at the types of encryptions available and decided to pick and choose which features I wanted to use. I settled on offsetting each character in the message by a random offset. In order to have it offset by a random number I needed to store a copy of the original index, in my created lists, of the character and the type of character it was. So, now with the original message being changed to a random assortment of numbers, letters, and special characters, I set up the output to be written to a file. The output would contain the offset, the message, the index of the original characters, and the characters’ original type.

Update 5 – 2:

After running the encryption program for a few messages and confirming that they did work, I began creating the decryption program, which was just taking in the inputs from the encrypted file and reorganizing the original message. Since I didn’t want to have to send my lists through the file, I just put a hard copy of the exact lists of letters, numbers, and special characters.

The decryption program wasn’t difficult to implement since it was just reversing the steps of my encryption program. I scanned the file and broke it down into the necessary components (the offset, the message, the indices, and the character types) and stored all the values into lists. Then, by comparing the original character’s index and type with the offset, I was able to pull the actual character from my given lists.

I ran the file and outputted to another text file and it worked.

Update 5 – 3:

I ran both programs again to test if they were working and I saw there were a few errors in the decryption. I traced the problem back to the fact that my list for special characters was missing a few frequently used characters like commas, apostrophes, periods, and a few others. My reasoning was that if I included these characters in the actual message, then I wouldn’t have a delimiter to be able to split the message by. In the end I settled on using “^” as my delimiter so I could add more special characters into the lists.

In addition, I felt that the encryption was a little too obvious, so I ended up adding a series of methods to break the array of original indices into a random set of chunks which then are places in a random order. After implementing this, I set the encryption to output to the encoded file the new information needed to sort the indices into their proper place. On the decryption side of things, I made a dictionary that zips up the chunks of indices with their respective order number. Then I appended each chunk in its proper order to a string that I plugged back into the original program.

Update 5 – 5:

I noticed that a user could decrypt any .txt file in their storage, so I added a snippet to the decryption file. This exception handler would scan the first line of text to see if the string “3NCRYP” was there. If so, then the file’s message would be decoded, otherwise it would give a message saying that the file is not encrypted.

I also added a way to choose whether to encrypt a message written into a file or to provide the program with a string to be encrypted. This was done for testing purposes to see if there would be any complications.

So far, I have finished with the codes for both my encryption and decryption methods. Everything is documented in the code. I need to finish a couple more things like creating the GUI for both systems and adding a bit more exception handling to both programs.

Update 5 – 8:

I completed the program for the GUI for the encryption method. I needed to rework the encryption an decryption methods in order for the GUI to call their main method. So, I ended up making new encryption and decryption files in order so I could keep the original versions in case they didn’t work out. Also, in the interest of ensuring that the user know how to use the program, I included instructions on the GUI as labels.

The main issues I ran into with the GUI was that I was not familiar with Tkinter and getting the info and passing that into the encryption method was a challenge. In the end, I settled on using Entry widgets because they were easier to use with my exception handling. I feel like they’re slightly limited since the text can only be typed into a short line. The Entry widget can still accept multiple sentences, but unfortunately it can’t take paragraphs. Another more minor problem was the design of the GUI. I wanted to add an image to serve as the logo for the program, but my computer wasn’t able to utilize PIL with Tkinter. I saw there were ways to used both in conjunction, but for some reason, my image would never be able to read into my canvas widget. Unfortunately, I was unable to add the image.

I wanted there to be proper feedback for the user if they were inputting the correct information. Also, the encrypted output file will be placed in the same directory as the program. As a side note, when inputting the names for the files, don’t add “.txt”. The program already does that.

Update 5 – 9:

I finished the GUI for the decryption method. The biggest complications were to execute the exception handling for the decryption method in the GUI rather than the main method. However, the GUI for the decryption function was more of a clone to the encryption GUI, just to be consistent with the theme. This decryption GUI will only have one input box for the name of the encrypted file. So, for this GUI there are three types of exceptions: 1) nothing was inputted, 2) the file exists, but is not encrypted, or 3) what was inputted is not an actual file. The other thing is that this GUI is laboring under the assumption that the encrypted file is in the same directory. If the file is on the desktop, then the program can’t see it.

Update 5 – 10:

I looked at a tutorial and changed the output of the decryption method to the desktop for ease of use. Everything else remained the same.

The zip file will contain 2 GUI programs, 2 encryption programs, and 2 decryption programs. The GUI’s will only use the version 2 for both encryption and decryption programs. The other two encoder and decoder programs are made for the python terminal. These allow the user to take in an input of either a file or a string of characters and encrypt the data. Both versions will encrypt the desired data in the same manner. The only difference is the utilization of a GUI. I’ll actually also include a test file as an example.