Reflective Analysis

Enigma was kind of an interesting project that focus on many aspects of cryptography. From its encryption algorithm to the way it was designed, Enigma was the strongest encryption hardware used during WW2. When I was given this project, I felt it difficult to be to build from the scratch. My first approach towards it was to learn about How enigma works and this link [**Click ME**](https://www.youtube.com/watch?v=ybkkiGtJmkM)helped me**.**

Ithelped in understanding:

* the basic structure of Enigma,
* components that are used
* how they are connected
* the algorithm of encryption and decryption
* Mechanical movement of rotors
* The concept of plugboard

I also took help from [Enigma Code](https://github.com/MWeis91/Enigma-Machine/tree/master/EnigmaMachine) . This way I was able to structure my software in my mind, Then I choose python as programming language and used tkinter library for gui. First of all, I made functions for all the components such as rotor\_1(), rotor\_2(), rotor3(), reflector (), plugboard () and use the default values to encrypt and decrypt the message. The default values of the board were taken from this [website](https://www.ciphermachinesandcryptology.com/en/enigmatech.htm). My next step was to synchronize the algorithm to successfully perform encryption and decryption and the sequence was:

**Plain-Text -> goes as array -> Goes to plugboard -> Goes to Rotor3 -> Breaks as alphabets -> Alphabet One by one goes to Rotor2 -> Rotor1 -> Reflector -> Rotor1 -> Rotor2 -> Rotor3 -> Plugboard -> Single alphabet joins in array to become Cipher-Text**

There is one thing to keep in mind in this whole algorithm is that when character moving from plugboard to Reflector it is changed from the alphabet list [] to the corresponding component list [] and from Reflector back to the plugboard it is changed from the corresponding component list [] to the alphabet list [].

Ok so now it’s time to focus on movement of rotors as one character is replaced the rotor list moves forward which means character from the rotor list should be removed but at the same time added at the end of the array, to do this I made another function named rotor\_connections (). For this purpose, I need to use two array list for one rotor, one which was to be used as working and the second was to be used for resetting as original and I named them as rotorlistWorking [] and rotorlist []. Another thing for the movement of rotor was that when the third rotor completes its first circle second rotor increments by one and when second rotor completes the whole circle the third rotor increases by one.

Here we are done with it and our text was able to successfully converted to cipher text and cipher text back to original.

Our next target was to convert everything into interface that is easier for the user, so for the rotor’s, sliders came into my mind and for the plugboard a big giant board with limitation of selecting 10 alphabets came in my mind. I have also added a logo at top and the name of the machine. At the backend I connected rotor, plugboard, input textbox and output-textbox with the corresponding components.

Another thing I did was to add export feature so whenever I encrypt long text, I am able to grab it using export button and it also gives the rotor setting key that were used for encryption.

What is it that I am going to do next time?

There is one exception error that I would like to tackle with next time. The error comes in the plugboard when two alphabets are selected at the same time for example {A.B} {C, B} here B is used for both A and C and this will generate error. Other than that, it was a perfect project to deal with.

Moreover, I will improve its graphical interface, instead of disabling plugboard buttons that gets selected on click I will use lines to connect them with. This will improve user interaction. To further improve user interaction, I will divide the system in two parts, where as soon as user execute the software, it will be given choice to choose whether to encrypt the plain-text or to decrypt cipher text.