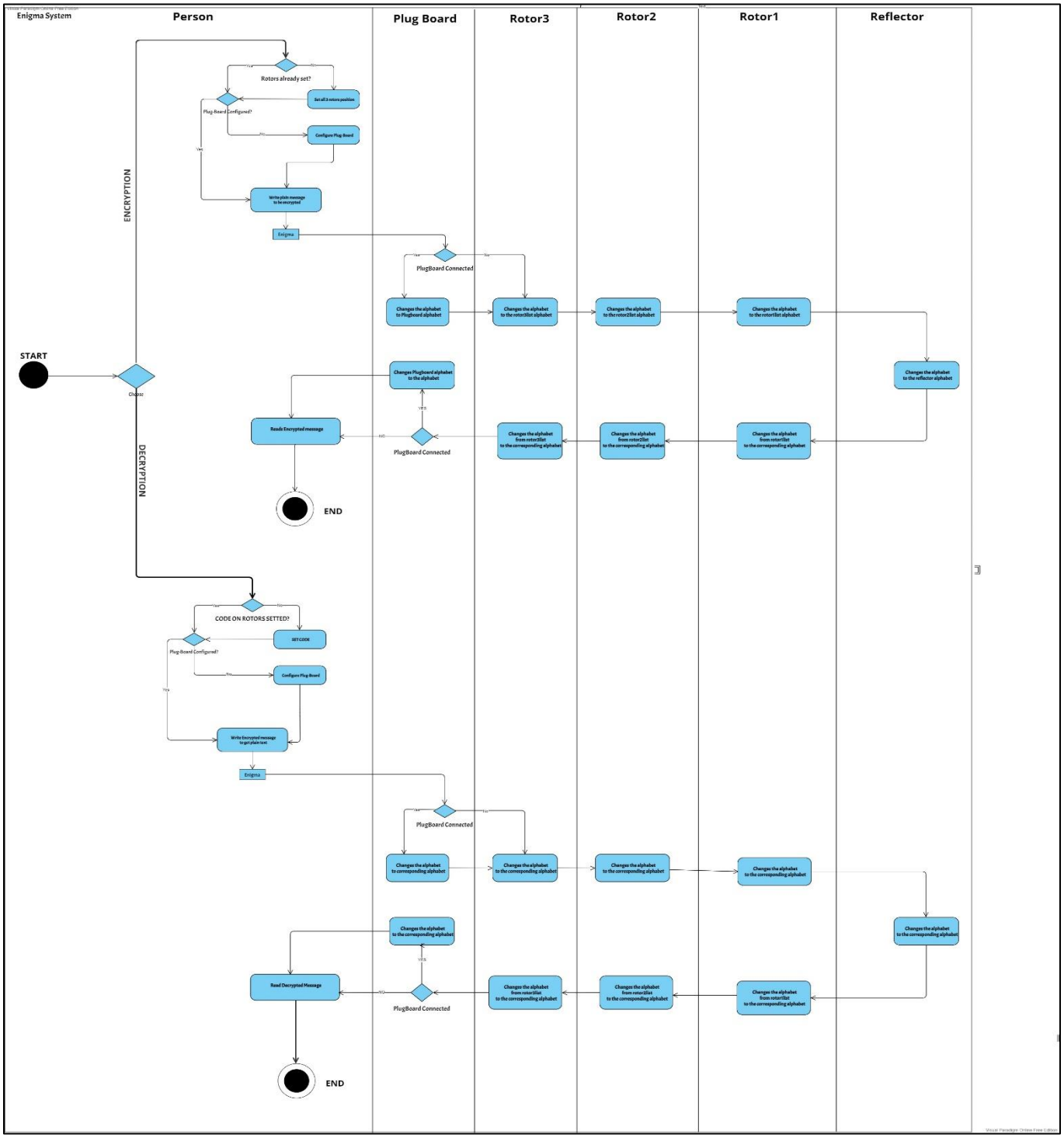
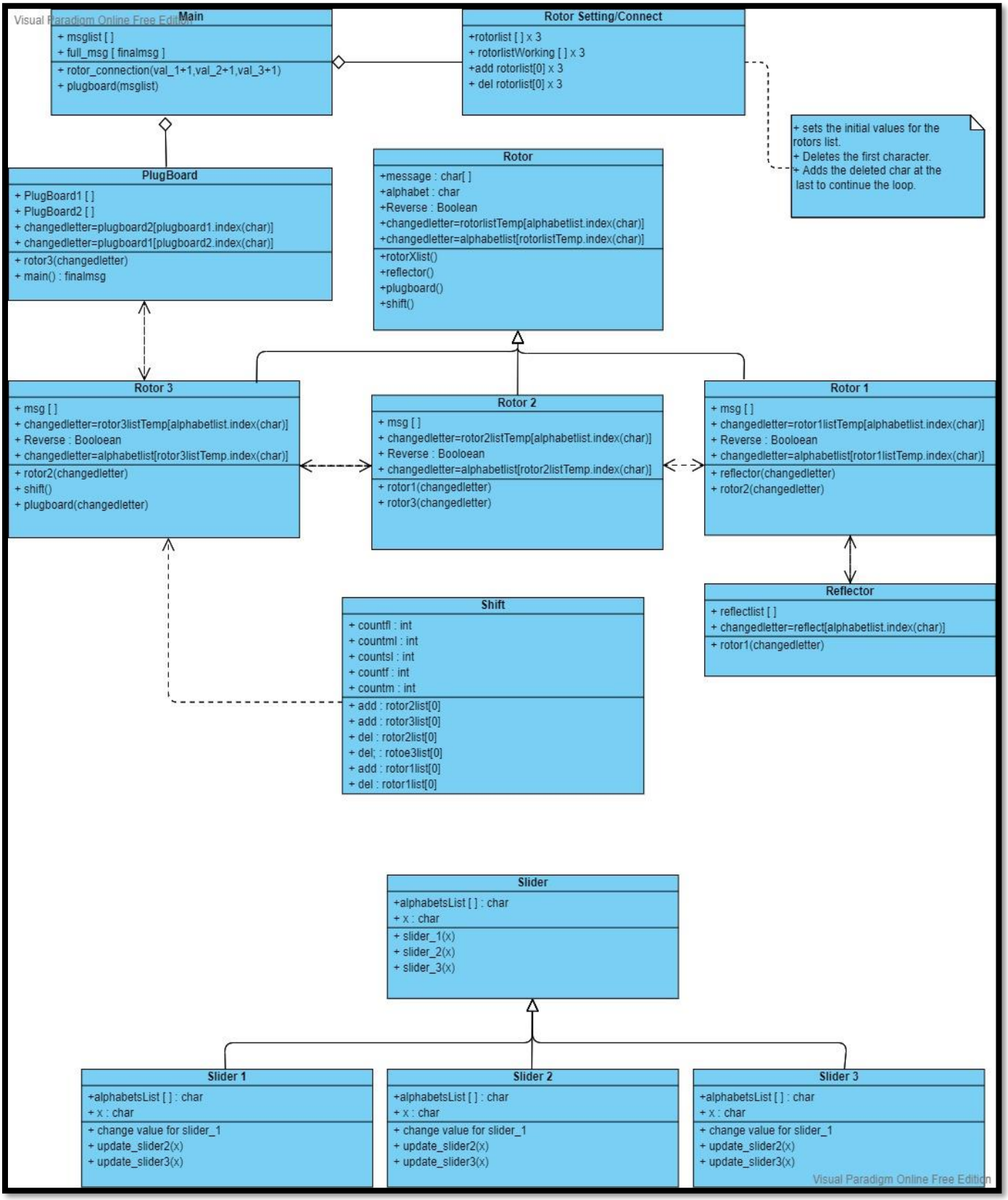
Report – The Enigma Machine

Phase I - System Design

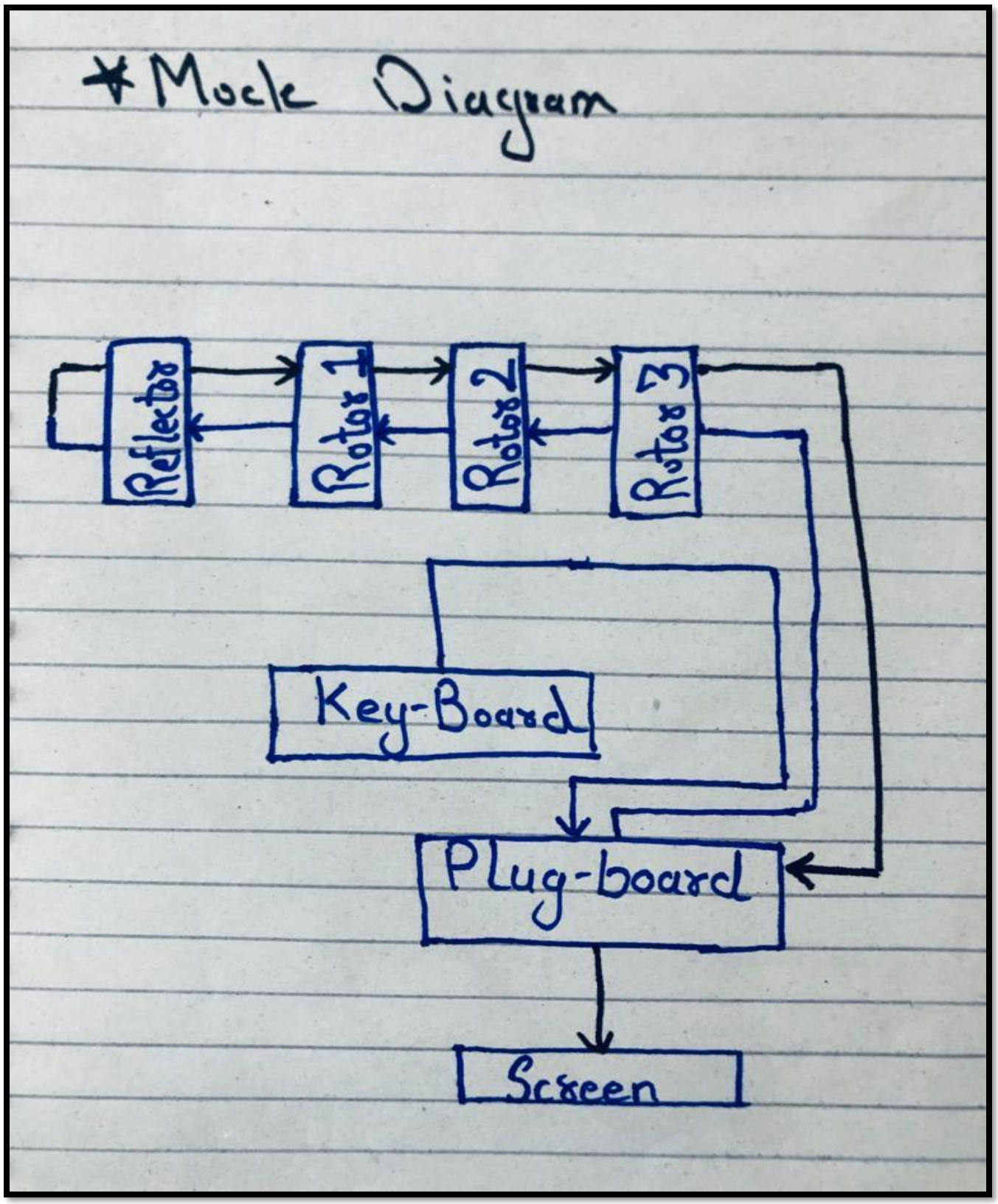
- **Activity Diagram**



- **Class Diagram**



- **Mock Diagram**



Phase II – Logs and Development

#PROBLEM 1

Which language to choose?

The first ever problem that I faced was which language am I going to use and I was stuck around two major high-level languages C# and python. As I am going to develop GUI-based software so the best language to use is C# but it's complex coding and a structure made me to step back from it.
Python is easy and fast to use. My main goal is to understand the cryptographical mechanism used in enigma machine. I can achieve this goal by using simple and understandable code for encryption and decryption of enigma. Easy code can make us understand how different components of enigma work for example plugboard, rotors, reflector keystroke and screen, how rotor rotates and how does it make encryption strong. I said no to classes in python programming language because I wanted our software to be as simple as possible but at the same time it does not affect our main goal.

I choose python over C#, our gui feature was tackled by using python library tkinter, our functions worked properly and our code was small and simple to understand.

```
File  Edit  Format  Run  Options  Window  Help
from tkinter import *
from tkinter.messagebox import *

gui = Tk()

gui.title("Enigma")
gui.configure(background="white")
```

#PROBLEM 2

Decryption was not working even after passing rotorlist [] array?

Setting up the rotors only gave us result from plain to cipher text but we were not able to decrypt it back to plain text.

This issue was resolved by debugging, our rotorsettings was changing continuously and this continuous change effect our rotorlist [] as some chars were added and some were removed. This became a serious problem. After thinking one two hours, I came up with an idea of using two separate arrays. One array was used for the encryption and the second array was used to reset the whole array when decryption is called. This first array is called rotorlist [] and the second one is called rotorWorkinglist []. Now when decrypting, it first reset the values from the rotorlist [] and then it successfully decrypts the message.

#PROBLEM 3

Converting my command line interface working to graphical user interface?

The third major problem that I faced was How am I going to convert my code into graphics, As I have already chosen tkinter as my python library but still I have to get a structure of how my software should look like.

Points that I noted as:

- Make logo
- Frame from 3 sliders that are connected to rotorsetting
- Using labels for everything
- Textbox for taking inputs from the user
- Enter button to run the input from the textbox to the algorithm
- Output box for displaying the result of encryption and decryption
- Clear text button that clears the output box

Outlining these points almost solved my problem, now I only had to work on the points one by one to get the finest output. Another important thing that I have to look was taking good care of errors and exceptions, for example when the fastest rotor is at Z it should shift to A automatically and at the same time middle rotor is also incremented by 1 alphabet.

Another problem that has to take care of, was the sequence of changing letter from one component to the next.

The sequence was

Plugboard -> Rotor3 -> Rotor2 -> Rotor1 -> Reflector -> Rotor1 -> Rotor2 -> Rotor3 -> Plugboard

In the first half the letter was changed from the alphabetlist to the plugboardlist / rotorlist / reflectorlist, but in the second half the letter was changed from the plug/rotor/reflector list to the alphabetlist.

NOTES

Encryption:
sending msg should not be read by anyone else
During war encryption was important, now encryption is used by computers
Enigma German militarties1930 and throughout WW2
Keyboard -> 26
Lamboard -> 26
press one letter on keyboard lampboard outputs different letter
Sender needs enigma machine to encrypt it + rotor key
Receiver need enigma to decrypt it + rotor key

Understanding the circuit
#Mid
Rotors: it has 3 rotors, every rotor has 26 numbers
As it has 3 rotos single letter is changed 3 times
From A to D
From D to T

From T to H
that's because of the wires which are scrambled

#Left
Reflector
#Right
Plugboard

Word is changed 7 to 9 times:
-->3 rotors x 2
-->1 reflector
-->Maybe plugboard x2

LOGS

->Write Functions
->Link them
->Note Default values
->Take input
->Display Output
->Rotate Rotor Function
->Keep eye on sequence on character that changes through the whole algorithm
->Check Decryption for the encrypted text
->Transfer and link every component to graphical user interface
-> Add logo and title
->Add rotors slider and link them at backend
->Add Input Text-box and link it at the backend
->Add Buttons
- Enter
- Quit
- Clear
- Plugboard
- Export

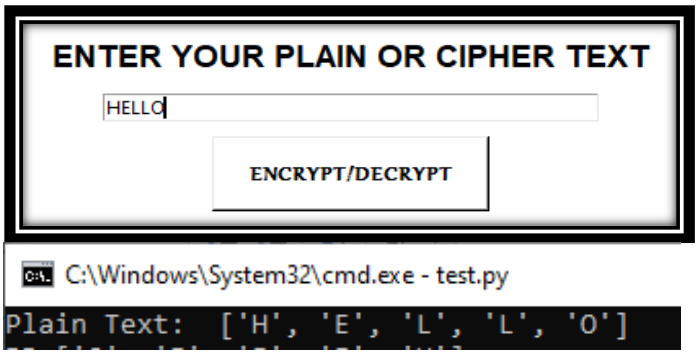
->Add Display Box and Display output over there
->Add Custom Plugboard and link it with the code that will change default plugboard with the custom one
->Add Export Feature to make a text file and dump output text in it + Add key of the rotor for decryption

Phase III – Testing

TEST TABLE

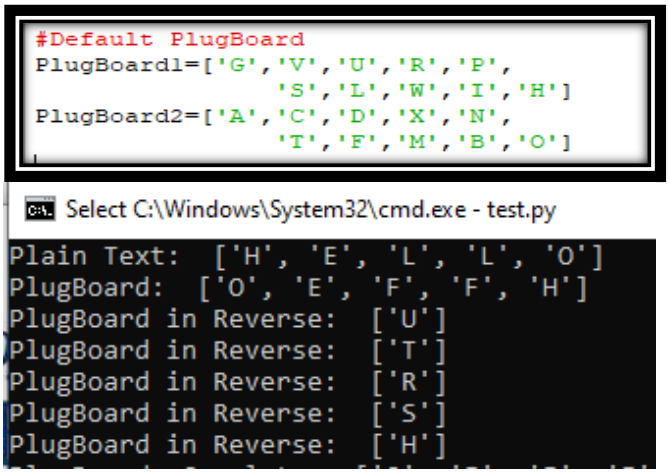
	Plain-Text	Plugboard	Rotor 1	Rotor 2	Rotor 3	Reflector	Encryption	Decryption	Cipher-Text
Mark									
	Working	Working	Working	Working	Working	Working	Working	Working	Working

✓ Plain-Text

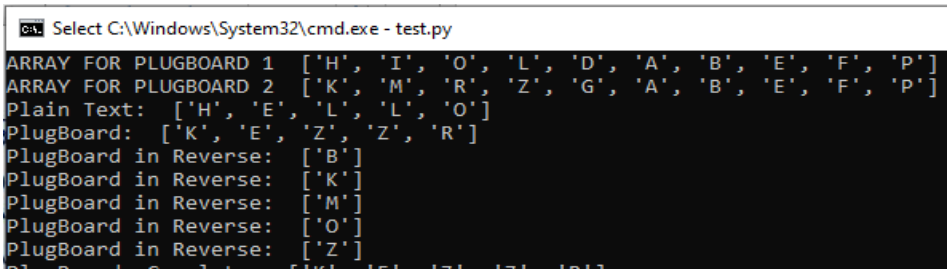


✓ Plugboard

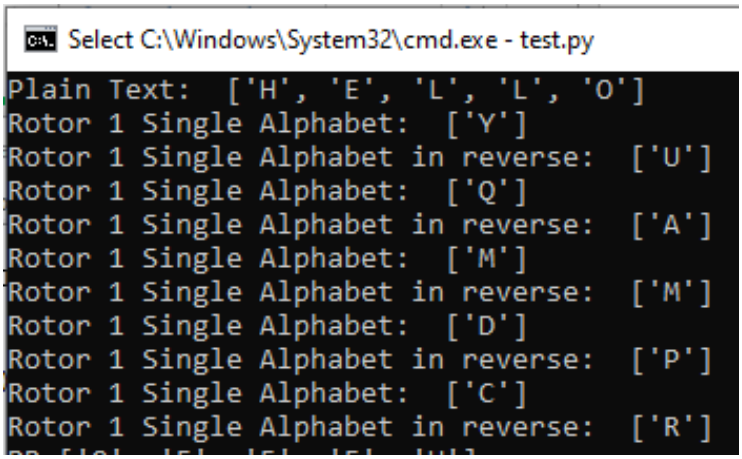
- **Default-Plugboard**



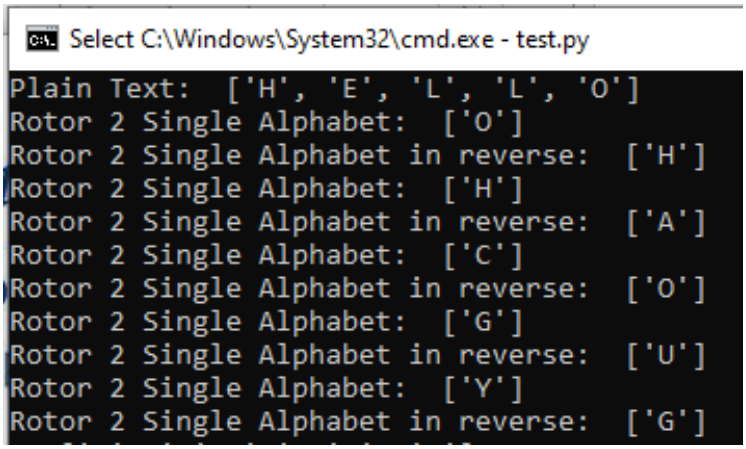
- **Custom-Plugboard**



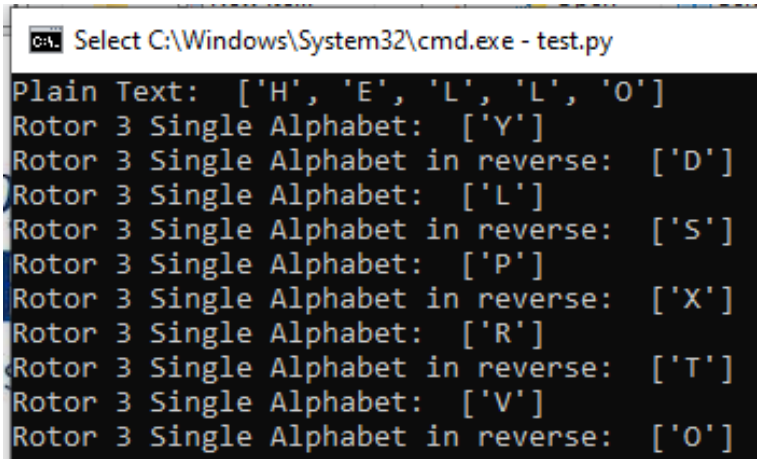
✓ Rotor 1



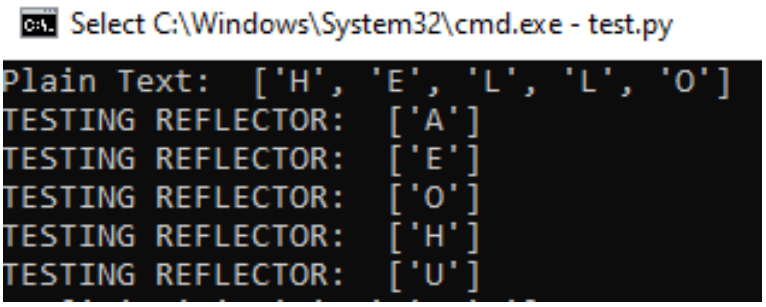
✓ Rotor 2



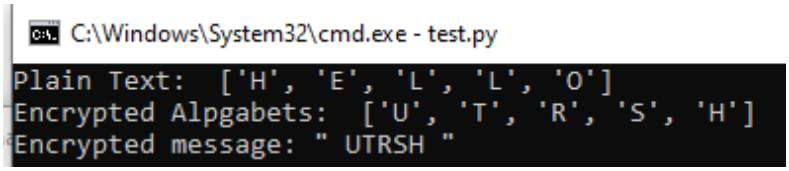
✓ Rotor 3



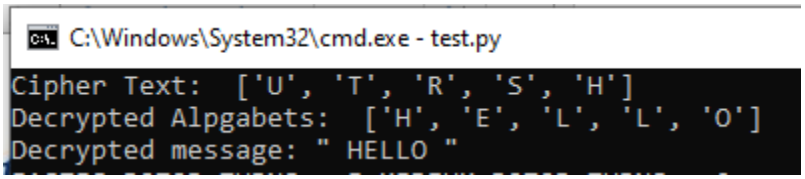
✓ Reflector

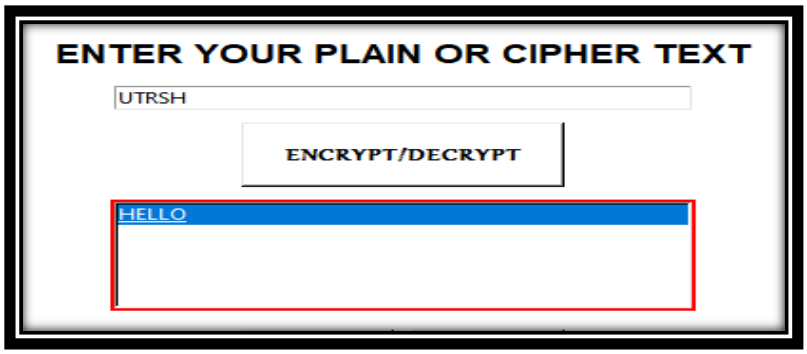


✓ Encryption



✓ Decryption

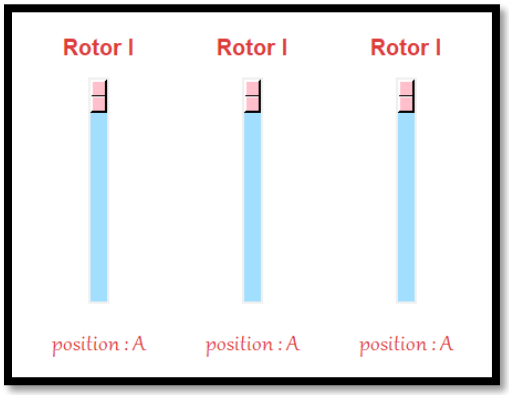


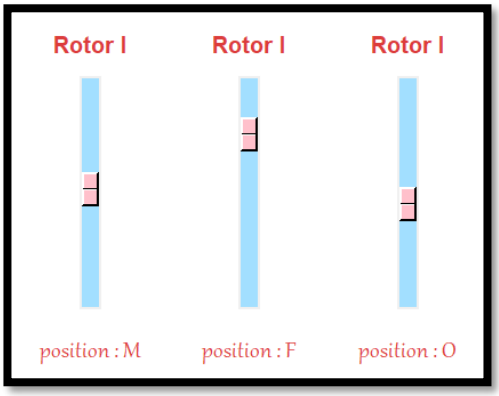


USABILITY TEST TABLE

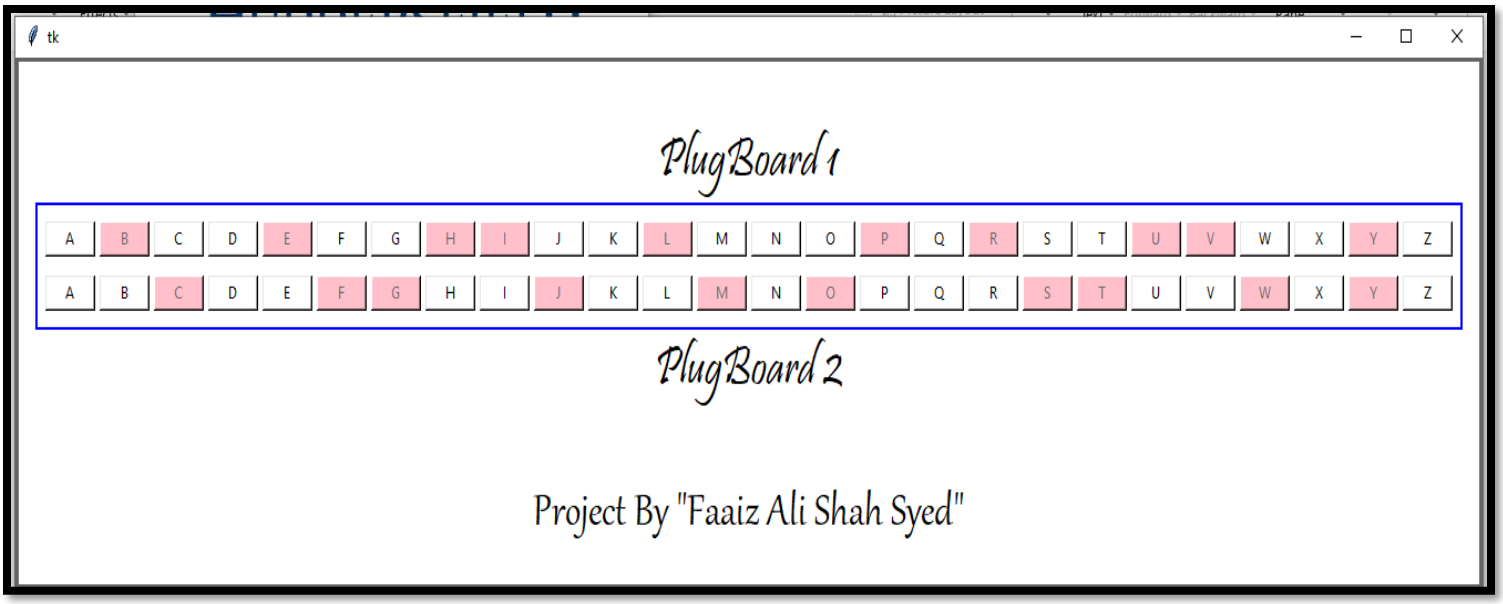
	Rotors	Plugboard	Text-Box	Output-Box	Export
Mark					
	Working	Working	Working	Working	Working

✓ Rotors

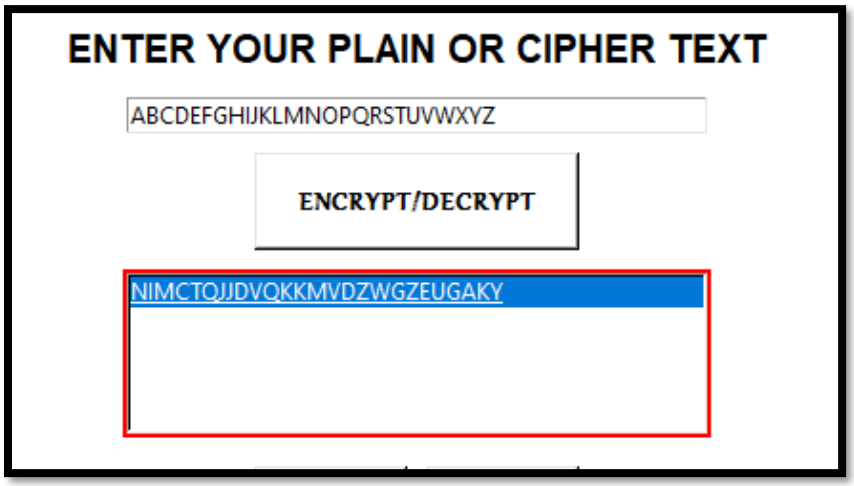




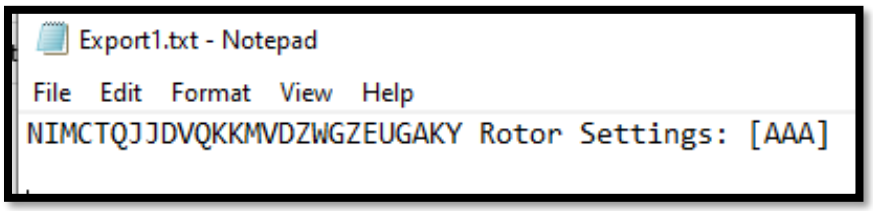
✓ Plugboard

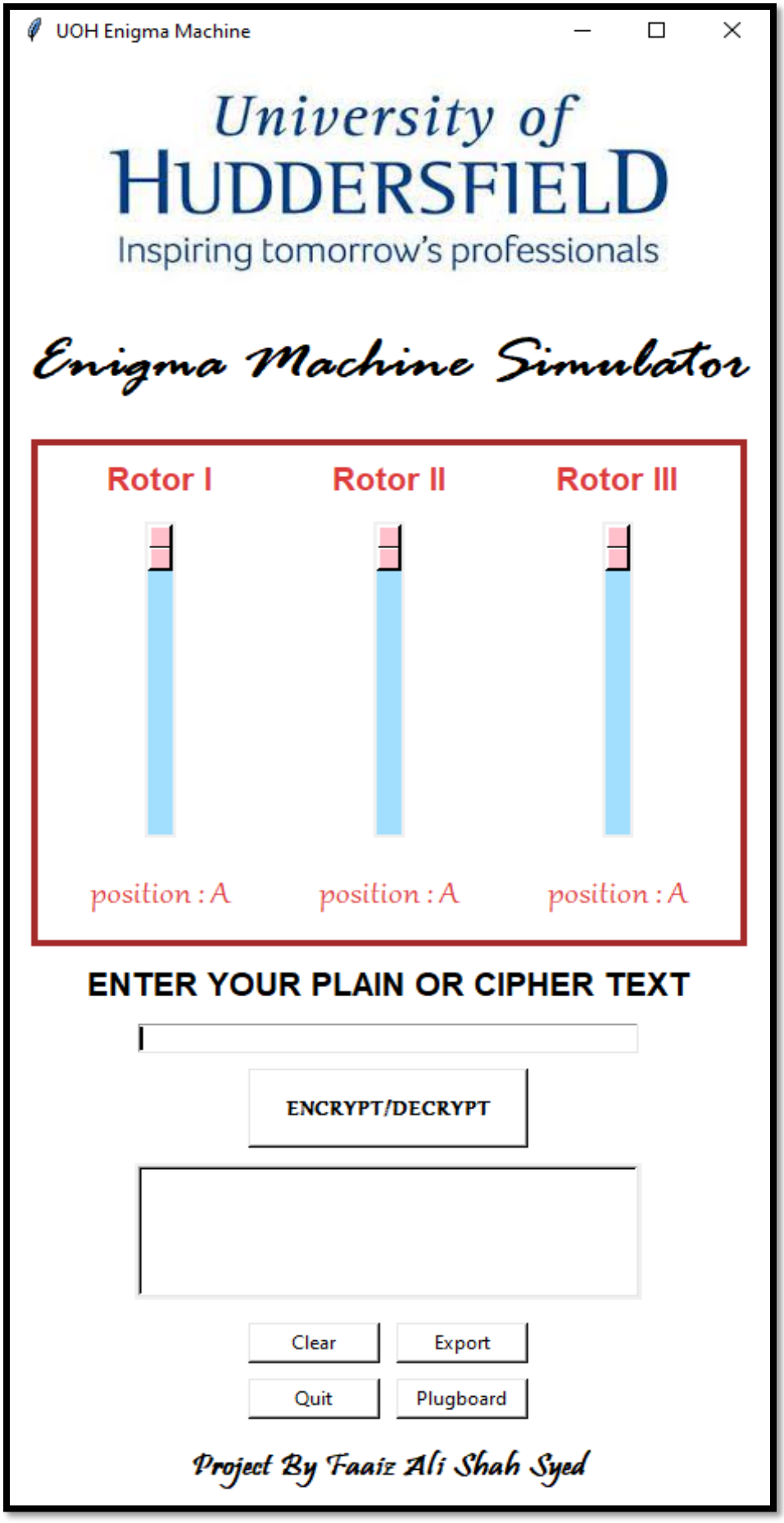


✓ Text-box & Output-box



✓ Export




UOH Enigma Machine
University of
HUDDERSFIELD
Inspiring tomorrow's professionals
Enigma Machine Simulator
Rotor I
Rotor II
Rotor III
position : A
position : A
position : A
ENTER YOUR PLAIN OR CIPHER TEXT
ENCRYPT/DECRYPT
Clear
Export
Quit
Plugboard
Project By Faaiz Ali Shah Syed

Phase IV – 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 helped me.

It helped 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 . 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. 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.