INDUSTRIAL TRAINING REPORT

ON

"Enigma Encryption System"

AT

Bharat Ratna Bhim Rao Ambedkar Institute of Telecom Training

*Submitted in partial fulfillment of the requirements for the 7th semester of **Bachelor of Engineering in Computer Science and Engineering***
of Chhattisgarh Swami Vivekanand Technical University

Submitted by:

S SHASHANK **300202220542**

Under the Guidance of:

Mrs. Preeti Shukla	**Mr. Vikas Soni**
Assistant Professor	**Training Coordinator**
Dept. of CSE	**BRBRAITT**

Department of Computer Science and Engineering
Chouksey Engineering College
Bilaspur, Chhattisgarh

Affiliated to

Chhattisgarh Swami Vivekanand Technical University

DECLARATION

Declared that entitled **“Enigma Encryption System”** has been successfully carried out by **S Shashank** bearing Roll Number **300202220542** bona fide student of **Chouksey Engineering College** in partial fulfillment of the requirements for the **7th semester Bachelor of Engineering** in **Computer Science and Engineering** of **Chhattisgarh Swami Vivekanand Technical University**.

Date: 04th DEC 2021

S SHASHANK

ACKNOWLEDGMENTS

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. I would like to take this opportunity to thank them all.

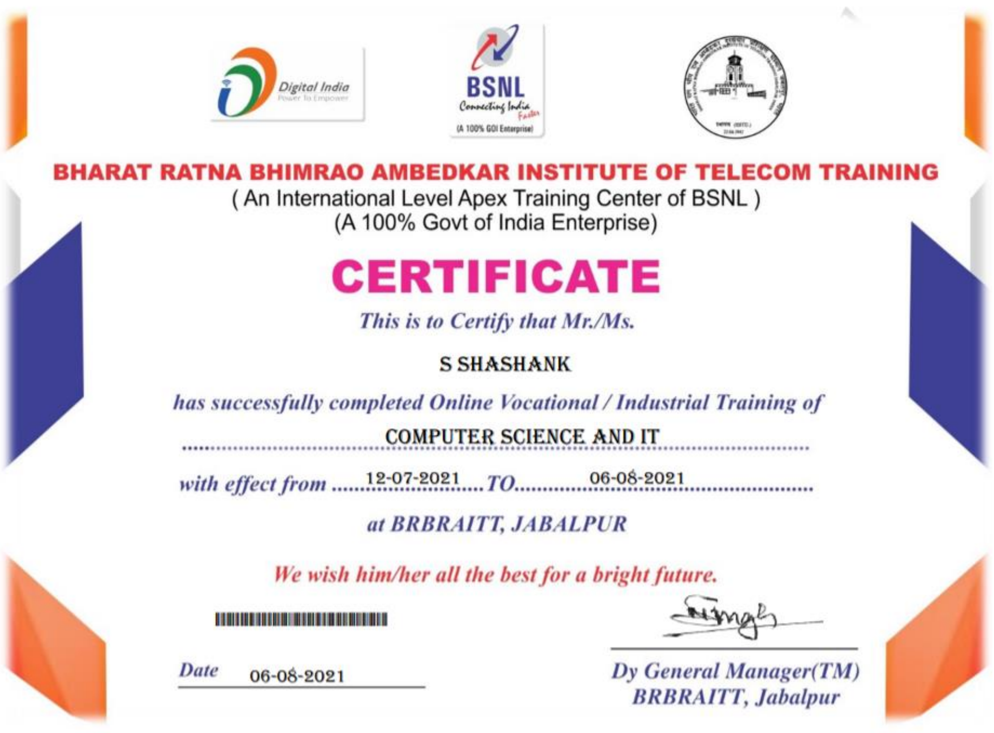
I would like to thank **Mrs. Preeti Shukla,** Assistant Professor, Department of CSE, Chouksey Engineering College, for their support towards the successful completion of this project.

I am grateful towards **Mr. Vikas Soni,** Training Coordinator and **Mr. J K Bachani**, Course Instructor, BRBRAITT, for their valuable suggestions and and assistance throughout this project.

Date: 4th DEC 2021

S SHASHANK (300202220542)

CERTIFICATE



Digital India
Power To Empower

BSNL
Connecting India Faster
(A 100% GOI Enterprise)

BHARAT RATNA BHIMRAO AMBEDKAR INSTITUTE OF TELECOM TRAINING
(An International Level Apex Training Center of BSNL)
(A 100% Govt of India Enterprise)

CERTIFICATE

This is to Certify that Mr./Ms.

S SHASHANK

has successfully completed Online Vocational / Industrial Training of

........COMPUTER SCIENCE AND IT........

with effect from ....12-07-2021.... *TO*....06-08-2021....

at BRBRAITT, JABALPUR

We wish him/her all the best for a bright future.

[illegible]

Date 06-08-2021

Dy General Manager(TM)
BRBRAITT, Jabalpur

ABSTRACT

The Enigma Encyrption System aims provides a simulation of the Engima Machine that was used during the Second World War by the German Forces to send encrypted messages for communication. The project provides an interface to input the plaintext and rotors for setting the key to obtain the ciphertext. The Enigma Encryption System uses a 3 Rotor System for encryption and decryption of the messages. It also provides an option to export the generated codes to a text file for external use. The Enigma Encryption System is implemented with the objective of simplifying the use of the encryption system by eliminating the use of complex hardware used in the past.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 ORGANIZATION PROFILE

BRBRAITT is an International Level apex Training Center of BSNL. Training is a life transformation process with upgradation in skills. Their mission is to deliver training of International Standards with a Team of highly skilled and experienced training professionals. BRBRAITT is having latest technology labs and huge infrastructure to bridge the gap between industry and academia. In addition to imparting industrial training, projects. internship we are delivering customized corporate training in the field of Telecom, Software and Management.

BRBRAITT is an Internationally Recognized and Acclaimed Training Institute of India, One of the Biggest in entire ASIA pacific region, More than 75 year old Premier Training Institute of India, More than 75 year old Premier Training Institute of India, Trained millions of National and International (CTO,APT ,UN,ITU etc) batches across the world, BRBRAITT is Bridging the gap between Industry and Academy having Industry exposure of more than 75 years with many International Trainers, The Training Centre has a campus of more than 35 acres with administrative building of 11,792 square meters in area and four hostels with a capacity to accommodate about 400 trainees(AC/Non AC), BRBRAITT is equipped with 2G/3G/4G/IOT/NGN/MPLS-VPN/ CMDA/ DWDW/FTTH/BRAS/ IDC(Cloud Computing & Data Center)/ Sat Com/ CDOT /Broadband/ ILL etc.

1.2 PROJECT DESCRIPTION AND PROBLEM STATEMENT

Problem Statement

Design a Graphical User Interface in a high-level programming language to simulate the working of Enigma Encryption System.

The aim of this project is to develop an interface to simulate the working of German Enigma Encryption System that was developed during the Second World War. Originally, a hardware machine, Enigma Encryption System aims of convert this system into a software feasible and compatible for today's Digital Age by reducing the complex hardware and manpower required to operate the system.

Description of the Enigma Machine

The **Enigma machine** is a cipher device developed and used in the early- to mid-20th century to protect commercial, diplomatic, and military communication. It was employed extensively by Nazi Germany during World War II in all branches of the German military. The Germans believed, erroneously, that use of the Enigma machine enabled them to communicate securely and thus enjoy a huge advantage in World War II. The Enigma machine was considered so secure that it was used to encipher even the most top-secret messages.

The Enigma has an **electromechanical rotor mechanism** that scrambles the 26 letters of the alphabet. In typical use, one person enters text on the Enigma's keyboard and another person writes down which of 26 lights above the keyboard illuminated at each key press. If plain text is entered, the illuminated letters are the encoded ciphertext. Entering ciphertext transforms it back into readable plaintext. The rotor mechanism changes the electrical connections between the keys and the lights with each keypress.

The security of the system depends on a set of machine settings that were generally changed daily during the war, based on secret key lists distributed in advance, and on other settings that were changed for each message. The receiving station has to know and use the exact settings employed by the transmitting station to successfully decrypt a message.

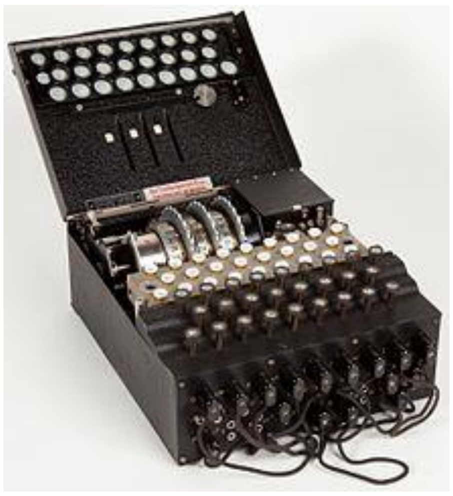


Fig.1.1 Enigma Machine

Like other rotor machines, the Enigma machine is a combination of mechanical and electrical subsystems. The mechanical subsystem consists of a keyboard; a set of rotating disks called *rotors* arranged adjacently along a spindle; one of various stepping components to turn at least one rotor with each key press, and a series of lamps, one for each letter.

The rotors form the heart of an Enigma machine. Each rotor is a disc approximately 10 cm (3.9 in) in diameter made from Ebonite or Bakelite with 26 brass, spring-loaded, electrical contact pins arranged in a circle on one face, with the other face housing 26 corresponding electrical contacts in the form of circular plates. The pins and contacts represent the alphabet — typically the 26 letters A–Z, as will be assumed for the rest of this description. When the rotors are mounted side by side on the spindle, the pins of one rotor rest against the plate contacts of the neighboring rotor, forming an electrical connection. Inside the body of the rotor, 26 wires connect each pin on one side to a contact on the other in a complex pattern.

CHAPTER 2

REQUIREMENT ANALYSIS

2.1 HARDWARE REQUIREMENTS

The Hardware requirements are very minimal and the program can be run on most of the machines.

Processor	:	Pentium4 processor
Processor Speed	:	2.4 GHz
RAM	:	1 GB
Storage Space	:	40 GB
Monitor Resolution	:	1024*768 or 1336*768 or 1280*1024

2.2 SOFTWARE REQUIREMENTS

Operating System	:	Windows 7,8,8.1,10
IDE	:	Thonny/PyCharm IDE

2.3 FUNCTIONAL REQUIREMENTS

2.3.1 Major Entities

1.The system must consist of a way to provide the plaintext as input and a text area to display the output.

2. The system must provide an option to export the outputs to an external text file.

3. The system must consist of at least three rotors for the encryption process.

4. The system must provide the functionalities of the original Enigma System such as rotors, plugboard, reflectors etc.

2.3.2 End User Requirements

The end users of this project would be a sender and a receiver who wish to communicate over a secure channel without revealing the original message contents. The sender should be able to input the plaintext and encrypt the desired message. The receiver upon receiving the secret rotor setting and the ciphertext should be able to produce the original message. A computer supporting Python would be enough to run this project.

2.3.3 Python

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a cycle-detecting garbage collection system (in addition to reference counting). Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.
Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's

semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages, indentation does not have any semantic meaning. The recommended indent size is four spaces.

2.3.4 Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's *de facto* standard GUI. Tkinter is included with standard GNU/Linux, Microsoft Windows and macOS installs of Python.

The name *Tkinter* comes from *Tk interface*. Tkinter was written by Fredrik Lundh.
Tkinter is free software released under a Python license.

Tcl/Tk is not a single library but rather consists of a few distinct modules, each with separate functionality and its own official documentation. Python’s binary releases also ship an add-on module together with it.

Tcl is a dynamic interpreted programming language, just like Python. Though it can be used on its own as a general-purpose programming language, it is most commonly embedded into C applications as a scripting engine or an interface to the Tk toolkit. The Tcl library has a C interface to create and manage one or more instances of a Tcl interpreter, run Tcl commands and scripts in those instances, and add custom commands implemented in either Tcl or C. Each interpreter has an event queue, and there are facilities to send events to it and process them. Unlike Python, Tcl’s execution model is designed around cooperative multitasking, and Tkinter bridges this difference (see Threading model for details).

Tk is a Tcl package implemented in C that adds custom commands to create and manipulate GUI widgets. Each Tk object embeds its own Tcl interpreter instance with Tk loaded into it. Tk’s widgets are very customizable, though at the cost of a dated appearance. Tk uses Tcl’s event queue to generate and process GUI events.

Themed Tk (Ttk) is a newer family of Tk widgets that provide a much better appearance on different platforms than many of the classic Tk widgets. Ttk is distributed as part of Tk, starting with Tk version 8.5. Python bindings are provided in a separate module, tkinter.ttk.

CHAPTER 3

IMPLEMENTATION

3.1 WORKING OF THE ENIGMA MACHINE

An Enigma machine is made up of several parts including a keyboard, a lamp board, rotors, and internal electronic circuitry. Some machines, such as the ones used by the military, have additional features such as a plugboard. Encoded messages would be a particular scramble of letters on a given day that would translate to a comprehendible sentence when unscrambled.

When a key on the keyboard is pressed, one or more rotors move to form a new rotor configuration which will encode one letter as another. Current flows through the machine and lights up one display lamp on the lamp board, which shows the output letter. So, if the "K" key is pressed, and the Enigma machine encodes that letter as a "P," the "P" would light up on the lamp board.

Each month, Enigma operators received codebooks which specified which settings the machine would use each day. Every morning the code would change.

For example, one day, the codebook may list the settings described in the day-key below:

Plugboard settings: A/L – P/R – T/D – B/W – K/F – O/Y

A plugboard is similar to an old-fashioned telephone switch board that has ten wires, each wire having two ends that can be plugged into a slot. Each plug wire can connect two letters to be a pair (by plugging one end of the wire to one letter’s slot and the other end to another letter). The two letters in a pair will swap over, so if “A” is connected to “Z,” “A” becomes “Z” and “Z” becomes “A.” This provides an extra level of scrambling for the military.

To implement this day-key first you would have to swap the letters A and L by connecting them on the plugboard, swap P and R by connecting them on the plugboard, and then the same with the other letter pairs listed above. Essentially, a one end of a cable would be plugged into the "A" slot and the other end would be plugged into the L slot. Before any further scrambling happens by the rotors, this adds a first layer of scrambling where the letters connected by the cable are encoded as each other. For example, if I were to encode the message APPLE after connecting only the "A" to the "L", this would be encoded as LPPAE.

Rotor (or scrambler) arrangement: 2 — 3 —1

The Enigma machines came with several different rotors, each rotor providing a different encoding scheme. In order to encode a message, the Enigma machines took three rotors at a time, one in each of three slots. Each different combination of rotors would produce a different encoding scheme. Note: most military Enigma machines had three rotor slots though some had more.

To accomplish the configuration above, place rotor #2 in the 1st slot of the enigma, rotor #3 in the 2nd slot, and rotor #1 in the 3rd slot.

Rotor orientations: D – K –P

On each rotor, there is an alphabet along the rim, so the operator can set in a particular orientation. For this example, the operator would turn the rotor in slot 1 so that D is displayed, rotate the second slot so that K is displayed, and rotate the third slot so that P is displayed.

3.2 BLOCK DIAGRAM

Fig.3.1 shows the block diagram of the Enigma Machine.

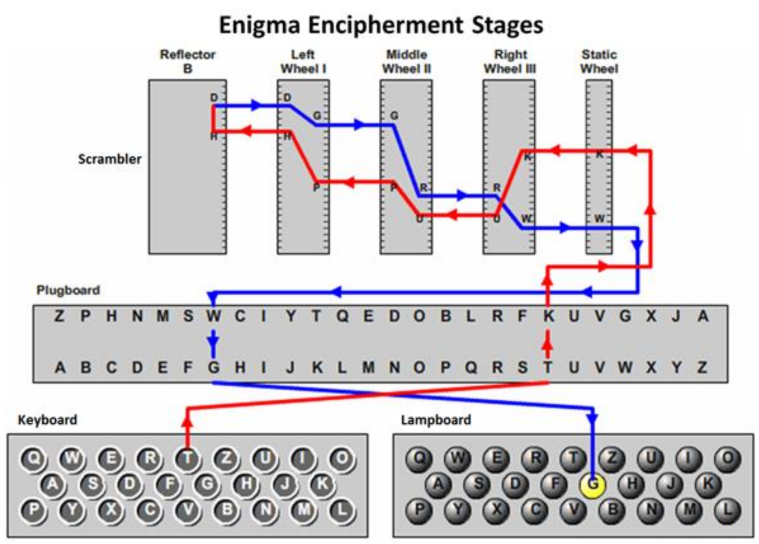


Fig.3.1 Block Diagram

3.3 PSEUDO CODE

3.3.1 Rotor

```
define plaintext message
```

```
define normal alphabet and scrambled alphabets
for each character in plaintext message:
    for each scrambled alphabet:
        get index of character in scrambled alphabet
        get new character at that index in normal alphabet
        replace character with new character
```

```
    concatenate transformed character to ciphertext message
```

3.3.2 Switchboard

```
define plaintext message
```

```
define list of switchboard swap pairs
for each character in message:
    for each pair in swap pairs:
        if character in swap pair, swap its value
```

```
    add character to ciphertext
```

3.3.3 Reflector

```
define plaintext message
```

```
define list of reflector swap pairs
for each character in message:
    for each pair in swap pairs:
        if character in swap pair, swap its value
```

```
    add character to ciphertext
```

3.3.4 Incrementing Rotor Wheels

```
function increment rotors:

    for each rotor/scrambled alphabet, left to right:
        get index of left notch in left alphabet
        get index of right notch in right alphabet
        if left index equals right index:
            cycle left alphabet forward 1 character

    cycle right-most alphabet forward 1 character
```

CHAPTER 4
CRYPTANALYSIS

Each of the three rotors display a number or letter and when the rotors turn, a new set of three numbers/letters appears. With the initial set of three numbers/letters (meaning the numbers/letters on the sender's machine when they began to type the message), a message recipient can decode the message by setting their (identical) Enigma machine to the initial settings of the sender's Enigma machine. Each rotor has 26 numbers/letters on it. An Enigma machine takes three rotors at a time, and the Germans could interchange rotors, choosing from a set of five, resulting in thousands of possible configurations. For example, one configuration of rotors could be: rotor #5 in slot one, rotor #2 in slot two, and rotor #1 in slot three.

How many configurations are possible with an Enigma machine with these specifications?

In the first slot, there are 5 rotors to pick from, in the second there are 4 rotors to pick from, and in the third slot there are 3 rotors to pick from. So there are 5x4x3 = 60 ways to configure the five rotors.
There are 26 starting positions for each rotor, so there are 26x26x26 = 17576 choices for initial configurations of the rotors' numbers/letters.
Since there are 26 letters in the alphabet, there are 26! ways to arrange the letters, but the plugboard can only make 10 pairs, so there are 20 letters involved with the pairings, and 6 leftover that must be divided out. Furthermore, there are 10 pairs of letters, and it does not matter what order the pairs are in, so divide also by 10! and the order of the letters in the pair does not matter, so divide also by 2^{10}. The resulting number of combinations yielded by the plugboard is as follows:

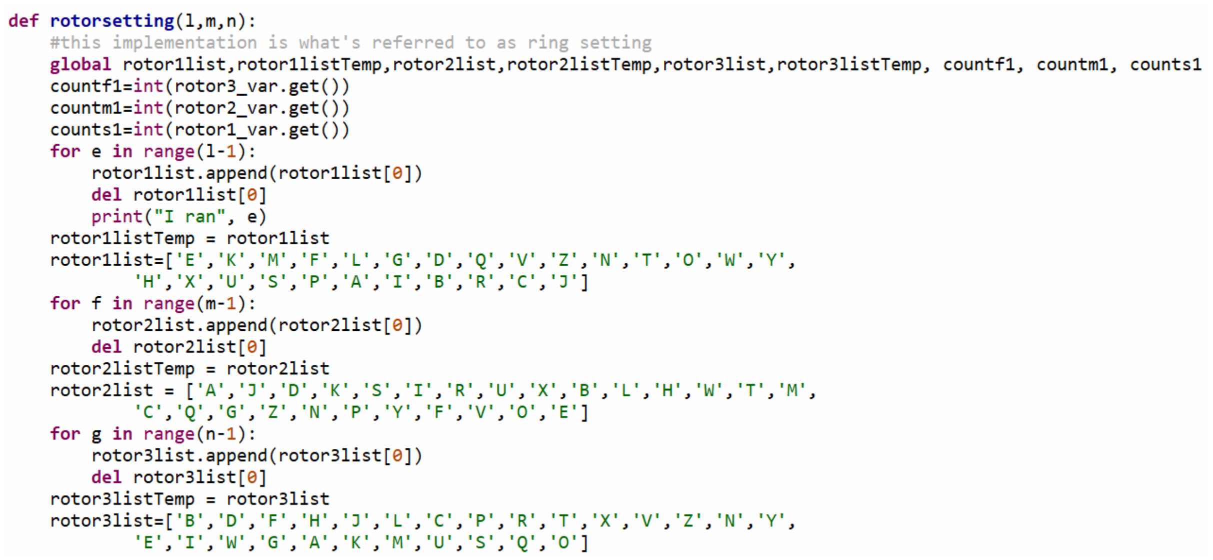
$$\frac{26!}{6!\,x\,10!\,x\,2^{10}} = 15{,}07{,}38{,}27{,}49{,}37{,}250$$

All of the components put together yields: 60x17576x150738274937250 =
15,89,62,55,52,17,82,63,60,000 total number of ways to set a military-grade Enigma machine.

CHAPTER 5

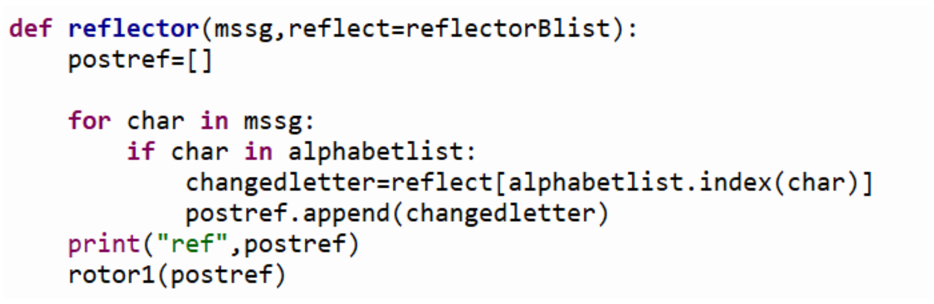
RESULTS AND CONCLUSION

5.1 SOURCE CODE



```python
def rotorsetting(l,m,n):
    #this implementation is what's referred to as ring setting
    global rotor1list,rotor1listTemp,rotor2list,rotor2listTemp,rotor3list,rotor3listTemp, countf1, countm1, counts1
    countf1=int(rotor3_var.get())
    countm1=int(rotor2_var.get())
    counts1=int(rotor1_var.get())
    for e in range(l-1):
        rotor1list.append(rotor1list[0])
        del rotor1list[0]
        print("I ran", e)
    rotor1listTemp = rotor1list
    rotor1list=['E','K','M','F','L','G','D','Q','V','Z','N','T','O','W','Y',
          'H','X','U','S','P','A','I','B','R','C','J']
    for f in range(m-1):
        rotor2list.append(rotor2list[0])
        del rotor2list[0]
    rotor2listTemp = rotor2list
    rotor2list = ['A','J','D','K','S','I','R','U','X','B','L','H','W','T','M',
          'C','Q','G','Z','N','P','Y','F','V','O','E']
    for g in range(n-1):
        rotor3list.append(rotor3list[0])
        del rotor3list[0]
    rotor3listTemp = rotor3list
    rotor3list=['B','D','F','H','J','L','C','P','R','T','X','V','Z','N','Y',
          'E','I','W','G','A','K','M','U','S','Q','O']
```

Fig.5.1 Rotor Settings



```python
def reflector(mssg,reflect=reflectorBlist):
    postref=[]

    for char in mssg:
        if char in alphabetlist:
            changedletter=reflect[alphabetlist.index(char)]
            postref.append(changedletter)
    print("ref",postref)
    rotor1(postref)
```

Fig.5.2 Reflector

```
def plugboard(mssg,lstplugboard1=dfltPB1,lstplugboard2=dfltPB2):
    global reverse, finalmsg
    postPB=[]

    for char in mssg:

        if char in alphabetlist:

            if char in lstplugboard1:
                changedletter=lstplugboard2[lstplugboard1.index(char)]
                postPB.append(changedletter)

            elif char in lstplugboard2:
                changedletter=lstplugboard1[lstplugboard2.index(char)]
                postPB.append(changedletter)

            else:
                postPB.append(char)


    if reverse==False:
        print("PB",postPB)
        rotor3(postPB)
    else:
        finalmsg.append(postPB)

        reverse=False
```

Fig.5.3 Plugboard Logic

5.2 SNAPSHOTS/OUTPUT

5.2.1 Snapshots

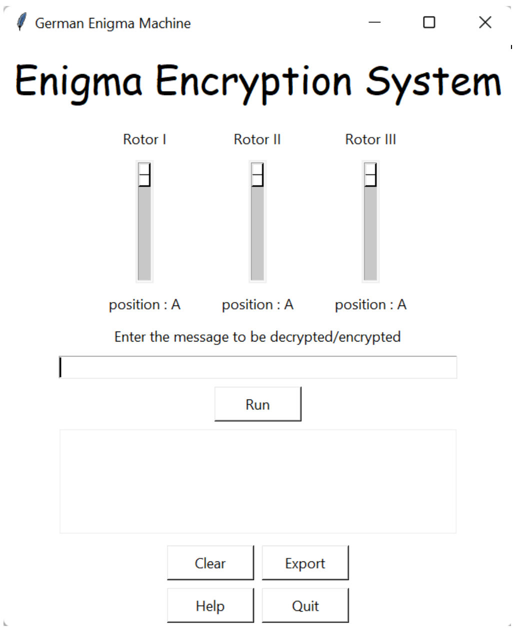
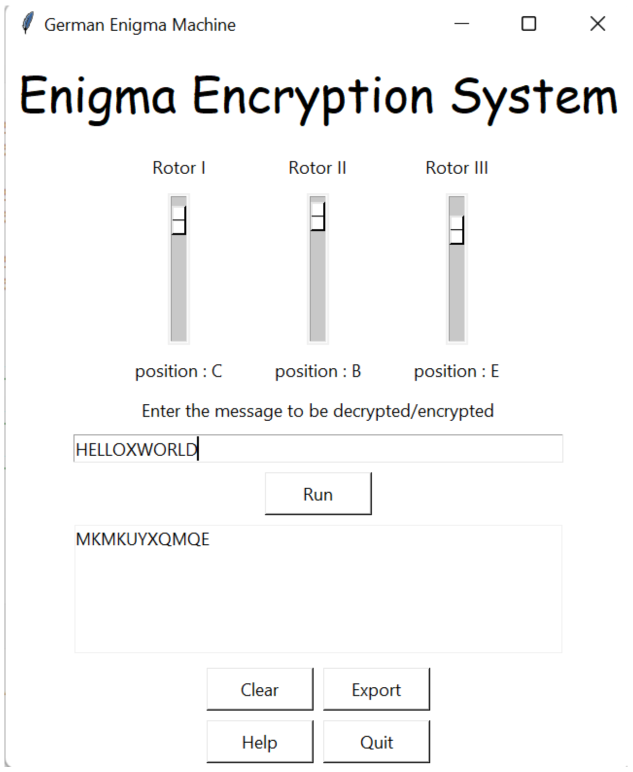


Fig.5.4 Initial Display


German Enigma Machine

Enigma Encryption System

Rotor I Rotor II Rotor III

position : C position : B position : E

Enter the message to be decrypted/encrypted

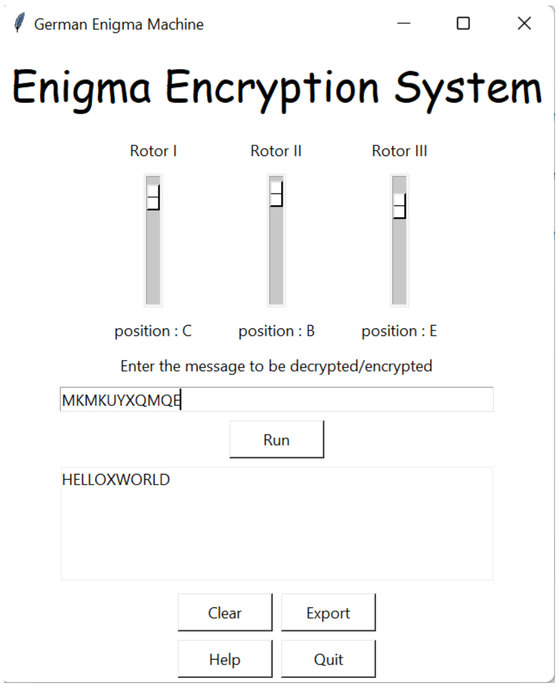
HELLOXWORLD

Run

MKMKUYXQMQE

Clear Export Help Quit

Fig.5.5 Encryption


German Enigma Machine

Enigma Encryption System

Rotor I Rotor II Rotor III

position : C position : B position : E

Enter the message to be decrypted/encrypted

MKMKUYXQMQE

Run

HELLOXWORLD

Clear Export Help Quit

Fig.5.6 Decryption

5.2.2 Test Cases and Outputs

S. No.	Input	Expected Output	Actual Output
1	Rotor: HAT Plaintext: HELLO	Rotor: HAY Ciphertext: WCIQT	Rotor: HAY Ciphertext: WCIQT
2	Rotor: PIP Plaintext: ATTACKATDAWN	Rotor: PJB Plaintext: XNKPXOVQOYRC	Rotor: PJB Plaintext: XNKPXOVQOYRC
3	Rotor: HAT Plaintext: WCIQT	Rotor: HAY Ciphertext: HELLO	Rotor: HAY Ciphertext: HELLO

Fig.5.7 Test Cases and Outputs

5.3 CONCLUSION

The Enigma Encryption System provides a user-friendly and reliable interface for the user to encrypt and decrypt the messages. It provides a strong encryption technique that cannot be easily decoded. The feature of exporting the generated messages makes it convenient to share it externally.

5.4 FUTURE ENHANCEMENTS

This system can be further made stronger by implementing more rotors to increase the complexity of the algorithm. It can be modified easily and deployed to a network server to enable all the participants on the network to communicate securely through this encryption scheme.

REFERENCES

- William Stallings: Cryptography and Network Security Principles and Practices.
- Charles Severance: Python for Everybody.
- www.stackoverflow.com
- https://www.codesandciphers.org.uk/enigma/rotorspec.htm