



PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Ft. Road, BSK III Stage, Bengaluru – 560 085

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Title: Problem Solving with C Laboratory		
Course code: UE19CS152		
Semester : II sem	Section: B	Team Id: PID6
SRN:PES1UG19CS255	Name: Malhar Pattekar	
SRN:PES1UG19CS492	Name: Soham Panini	
SRN:PES1UG19EC262	Name: Samriddha Shukla	
SRN:PES1UG19CS311	Name: Om Vasmate	

PROJECT REPORT

Problem Statement:

Protecting the commercial, political and military information using a encryption device in order to send and receive information in early to mid -20 centuries.

Description:

- **Introduction.**

The encryption device simulator is an imitation of the encryption device named "enigma". The project code works on the same lines of decrypting and encrypting the data sent and received during the World War-II. The encryption device is based on the transformation of 26 alphabets by various permutations.

- **Basic Operation.**

An operator would be given a plain-text message to encrypt. After setting up his machine, he would type the message on the Enigma Simulator keyboard. For each letter pressed, one lamp lit indicating a different letter according to a pseudo-random substitution determined by the electrical pathways inside the machine. The letter indicated by the lamp would be recorded, typically by a second operator, as the cypher-text letter. The action of pressing a key also moved one or more rotors so that the next key press used a different electrical pathway, and thus a different substitution would occur even if the same plain-text letter were entered again. For each key press there was rotation of at least the right hand rotor and less often the other two, resulting in a different substitution alphabet being used for every letter in the message. This process continued until the message was completed.

- **Working. :**

Let's say we want to encrypt the word Hackaday. This is called the plaintext message. We'll treat all letters as uppercase. Using the above cipher we look in the top line for the H and we substitute the letter below it, a Z. Similarly, looking for the A in the top line, we see we should substitute it with a G. The encrypted text becomes ZGTBGNGV. This is called the ciphertext. To decrypt it we do the reverse, look for each letter in the bottom row and substitute with the corresponding letter in the top row, getting HACKADAY

initial position	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
	G	E	T	N	D	H	Q	Z	U	P	B	R	C	O	X	M	K	Y	A	W	F	I	L	S	V	J
first turn	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
	J	G	E	T	N	D	H	Q	Z	U	P	B	R	C	O	X	M	K	Y	A	W	F	I	L	S	V
second turn	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
	V	J	G	E	T	N	D	H	Q	Z	U	P	B	R	C	O	X	M	K	Y	A	W	F	I	L	S
third turn	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
	S	V	J	G	E	T	N	D	H	Q	Z	U	P	B	R	C	O	X	M	K	Y	A	W	F	I	L

The ability to change the mapping is important because once someone deduces that G is the substitution for A, they'll know that's true for every G in the ciphertext. An improvement would be for all those pairings to change. And even better, if they change each time a letter is encoded.

One way to easily implement that, and it's the way it's done in the Enigma, is to embed all that wiring in a wheel/rotor. By turning the rotor while leaving the letters stationary, the connections between letters change. After one turn, A would now be substituted with whatever Z formerly was, namely J. Repeating this step of substitution followed by turning the rotor for each letter, HACKADAY becomes ZJGZLVFA. Follow the diagram above and you notice that the first Z came from the H, but the Z in the fourth position came from the K.

ADDING A PLUG BOARD: This helps to swap the letters or allocates one letter for the other and when the letter is called it is swapped by the letter allocated to you.

- **Conclusion:**

This is a user-friendly three-rotor simulator, where users can select rotors, use the plugboard and define new settings for the rotors and reflectors is available. The output appears in separate windows which can be independently made "invisible" to hide

decryption. This option produces 26 characters in less than one second. The encryption simulator helped a lot to learn about enigma and helped us to build on the same lines.

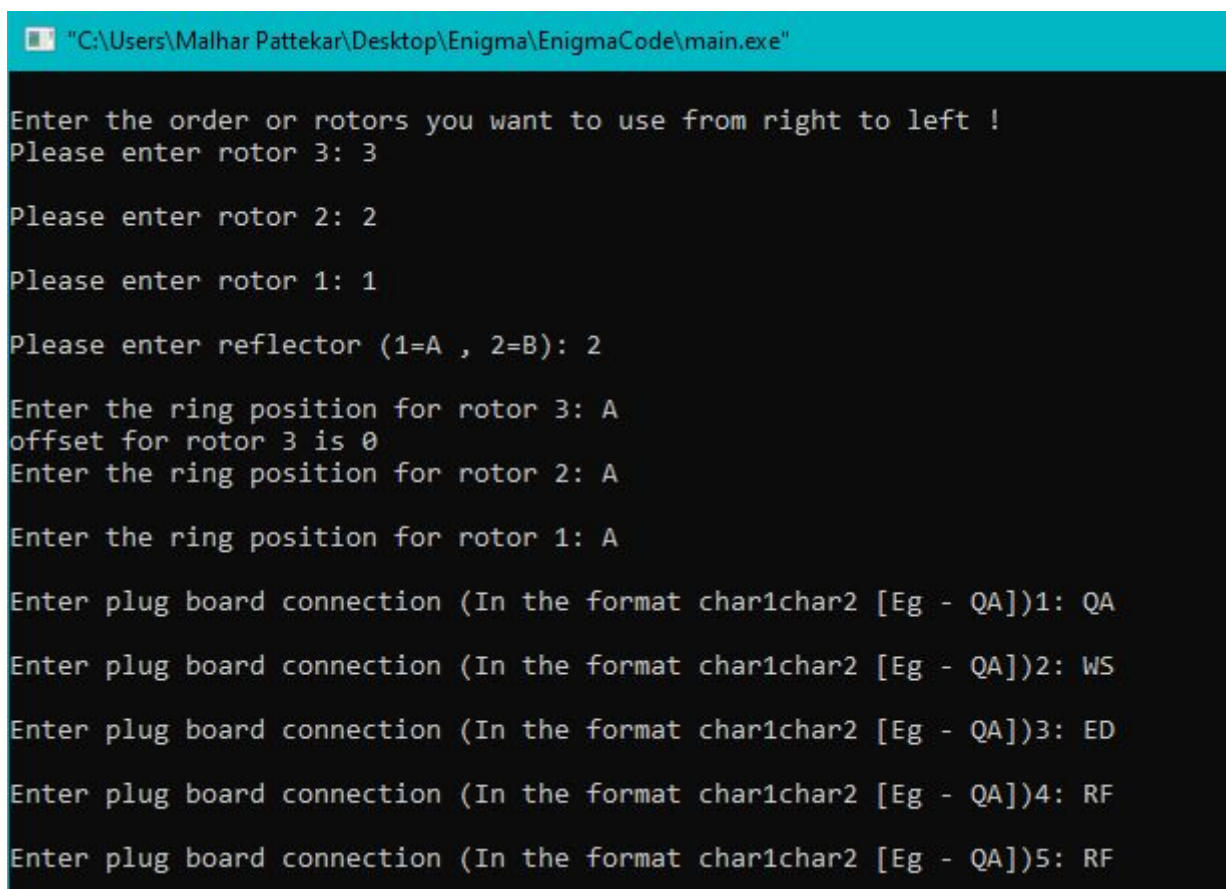
C-concepts used:

- Functions
- Arrays
- Basic mathematical operations
- String functions
- Conditional Statements

Learning Outcome:

We have understood the importance of security and privacy in today's day and age when all the make shifts have been perloc. Increased confidence in new and complex scenarios, stronger working relationships, positive shared experiences of overcoming real-world problems, technical problem solving, demonstrating pioneering industry leadership, confidence in self-directed learning.

Output Screenshots:



```
"C:\Users\Malhar Pattekar\Desktop\Enigma\EnigmaCode\main.exe"

Enter the order or rotors you want to use from right to left !
Please enter rotor 3: 3

Please enter rotor 2: 2

Please enter rotor 1: 1

Please enter reflector (1=A , 2=B): 2

Enter the ring position for rotor 3: A
offset for rotor 3 is 0
Enter the ring position for rotor 2: A

Enter the ring position for rotor 1: A

Enter plug board connection (In the format char1char2 [Eg - QA])1: QA
Enter plug board connection (In the format char1char2 [Eg - QA])2: WS
Enter plug board connection (In the format char1char2 [Eg - QA])3: ED
Enter plug board connection (In the format char1char2 [Eg - QA])4: RF
Enter plug board connection (In the format char1char2 [Eg - QA])5: RF
```

"C:\Users\Malhar Pattekar\Desktop\Enigma\EnigmaCode\main.exe"

Rotor positions 0 0 11
After encryption by rotor Y

Rotor positions 0 0 11
After encryption by rotor C

Rotor positions 0 0 11
After encryption by rotor R

Rotor positions 0 0 11
After encryption by rotor G

Rotor positions 0 0 11
After encryption by rotor X

Rotor positions 0 0 12
After encryption by rotor S

Rotor positions 0 0 12
After encryption by rotor Z

Rotor positions 0 0 12
After encryption by rotor J

Rotor positions 0 0 12
After encryption by rotor Q

Rotor positions 0 0 12
After encryption by rotor Q

Rotor positions 0 0 12
After encryption by rotor U

Rotor positions 0 0 13
After encryption by rotor D

Rotor positions 0 0 13
After encryption by rotor K

Rotor positions 0 0 13
After encryption by rotor N

Rotor positions 0 0 13
After encryption by rotor B

Rotor positions 0 0 13
After encryption by rotor J

"C:\Users\Malhar Pattekar\Desktop\Enigma\EnigmaCode\main.exe"

Rotor positions 0 0 18
After encryption by rotor H

Rotor positions 0 0 18
After encryption by rotor L

Rotor positions 0 0 18
After encryption by rotor J

Rotor positions 0 0 19
After encryption by rotor Q

Rotor positions 0 0 19
After encryption by rotor Q

Rotor positions 0 0 19
After encryption by rotor X

Rotor positions 0 0 19
After encryption by rotor Z

Rotor positions 0 0 19
After encryption by rotor S

Rotor positions 0 0 19
After encryption by rotor M

Rotor positions 0 0 20
After encryption by rotor W

Rotor positions 0 0 20
After encryption by rotor F

Rotor positions 0 0 20
After encryption by rotor G

Rotor positions 0 0 20
After encryption by rotor E

Rotor positions 0 0 20
After encryption by rotor Z

Rotor positions 0 0 20
After encryption by rotor P

The encrypted string is XUDYNEZJMP

Name and Signature of the Faculty