Enigma Machine Simulator

Elliot Brooks

Supervisors:

April 2024

# Abstract

Contents

[1 Abstract 2](#_Toc157955239)

[2 Introduction 4](#_Toc157955240)

[2.1 Aims and Objectives 4](#_Toc157955241)

[3 History & Background 5](#_Toc157955242)

[3.1 History of the Machine 5](#_Toc157955243)

[3.2 Design of the Machine 5](#_Toc157955244)

[3.2.1 Keyboard & Lampboard 5](#_Toc157955245)

[3.2.2 Rotors 5](#_Toc157955246)

[3.2.3 Reflector 6](#_Toc157955247)

[3.2.4 Plugboard 7](#_Toc157955248)

[3.2.5 Enigma Machine and Abstraction 7](#_Toc157955249)

[3.3 Design Flaws 10](#_Toc157955250)

[3.4 Possible Solutions 10](#_Toc157955251)

[3.5 Similar Products 10](#_Toc157955252)

[4 Design & Implementation 10](#_Toc157955253)

[4.1 Tools and Technologies 10](#_Toc157955254)

[4.2 Enigma Model Design 10](#_Toc157955255)

[4.2.1 Keyboard & Lamp board 10](#_Toc157955256)

[4.2.2 Reflector 10](#_Toc157955257)

[4.2.3 Rotors 10](#_Toc157955258)

[4.2.4 Enigma 10](#_Toc157955259)

[4.3 Design of the Application 10](#_Toc157955260)

[4.3.1 Command Line Interface 10](#_Toc157955261)

[4.3.2 GUI 10](#_Toc157955262)

[4.3.3 Visualization 10](#_Toc157955263)

[5 Results 11](#_Toc157955264)

[6 Evaluation 11](#_Toc157955265)

[6.1 Model Evaluations 11](#_Toc157955266)

[6.2 GUI Evaluation 11](#_Toc157955267)

[7 Conclusion 11](#_Toc157955268)

[8 References 11](#_Toc157955269)

[9 Table of Figures 11](#_Toc157955270)

[9.1 Figures 11](#_Toc157955271)

[9.2 Tables 12](#_Toc157955272)

[9.3 Equations 12](#_Toc157955273)

# Introduction

The Enigma Cypher Machine (*Enigma I)* is most well-known for its usage and eventual breaking, during World War 2 (WW2). The machine used several mechanical rotors, a reflect and a plugboard to redirect electrical signals, scrambling a plaintext message into cyphertext. Whilst most have heard of the machine, either from a study of cryptography or through history, many do not understand how the machine functions and its significance in cryptography. By creating a visual tool to demonstrate the inner workings of the Enigma, the aim of the project was to offer a deeper insight into the functionality, significance, but also the weaknesses of the machine.

There already exists a handful of examples of Enigma simulations which tend to offer a skeuomorphic approach to modelling the machine. The challenge I wanted to overcome was to create a tool that teaches the user how the machine functions logically. I began my project by thoroughly researching *Enigma I* to understand the inner workings of the machine and to compile this research into a simpler, abstracted model using Java. I created both a multi-platform command line interface (CLI) and a multi-platform graphical user interface (GUI) to allow the user to interact with the model, the former offering a step-by-step visualisation of the encryption of a message. To complete the project, I developed a second “Enhanced Enigma” model which aims to correct the two key cryptographic weaknesses of the machine. I identified the accuracy of my model by comparing with similar products as well as some real-world messages found amongst the German forces during WW2. I also identified the effectiveness of the two Enigma models by using some basic cryptoanalysis.

## Aims and Objectives

The following list adapted from the original project description, however there are some additional objectives I have included myself. For each task I set my own criteria to assess the success of each task’s implementation.

|  |  |
| --- | --- |
| Aim / Objective | Criteria for Success |
| Create a standalone package that simulates the Enigma Machine | The package should simulate only the Enigma machine and should be able to be used in different applications. It should also produce the correct output for a given input. The package should allow the Enigma Machine to be fully configurable as the ‘Enigma I’ |
| Create a new “Enhanced Enigma” machine with an aim of fixing cryptographic weaknesses | The enhanced machine should function similarly to the original Enigma and should be more secure cryptographically |
| Create a basic command line interface | The CLI should be allow full configuration of the machine and allow the user to enter a plaintext message |
| Create a GUI with a visualisation tool | The GUI should have the same capabilities, if not more than the CLI. In addition, it should aim to visually reflect the machine.  The visualisation tool should provide a simplified and informative representation of the encryption steps |
| Support multiple platforms | The application should be compatible with Windows/MacOS/Linux |

# History & Background

## History of the Machine

The Enigma cypher is one of the most famous cipher machines (Crypto museum n.d.) due to its role in WW2 and the work undertaken at Bletchley Park to crack the code. The Enigma was used extensively by the German forces to transmit coded messages for more secure communication. Whilst most people refer to this machine as “The Enigma”, Enigma is a brand name for a series of cypher machines (Crypto museum n.d.). The one used most in WW2 was *Enigma I* and was the key focus of this project.

To use the Enigma machine, it had to be configured to an exact setting so that the machine could be correctly decoded by the intended parties. These settings were distributed to the German forces each month in a code book, containing each day’s settings.

TODO : Add more history here

## Design of the Machine

The Enigma Machine was a rotor-based machine and worked with a hybrid of mechanics and electrical signals. *Enigma I* contained 5 key components that worked together to produce cyphertext which were: the keyboard and lampboard, the rotors, the reflector, and the plugboard. The inclusion of all these components aimed to make the machine as unpredictable as possible as well as generating a large key-length of roughly 67 bits (nearly 159 quintillion different settings) (van Manen and Robertsson 2016).

The use of the machine was a simple process. A user would receive a message along with a key denoting the choice of rotors and their respective settings, the plugboard settings and a reflector choice. Each time a key was pressed on the keyboard, a light on the lampboard would emit light as demonstrated in Owen’s animation (Owen 2021) and the user would write down the corresponding letter. Due to the symmetric design of the Enigma machine, as long as two people had the same settings then they could simply input the cyphertext into the machine and receive the plaintext and vice versa. However, this design choice was overlooked and compromised the Enigma’s security (See 3.4) (Thimbleby 2016).

TODO : figure of genuine enigma machine

### Keyboard & Lampboard

The keyboard and lampboard were the interface which the user could encode/decode messages. The former was comprised of the 26-letter alphabet omitting any special or numeric characters. Each key on the keyboard could be pressed which would cause a ratchet mechanism to move a lever (pawl) to step the rotors (Hamer 1997). The lampboard was a copy of the keyboard but instead of keys, there were small glass panels which would allow the bulbs underneath to shine through. These panels were also printed with the 26-letter alphabet and upon a keypress, any given lamp could light up to show the plaintext character’s corresponding cyphertext.

### Rotors

The rotors were the heart of the Enigma machine and were responsible for most of its unique properties. They were metal ratchet discs with 26 different positions representing each letter of the alphabet. Each position had a corresponding metal contact (Owen 2021) on both sides of the disk to allow electrical current to flow through the rotor. Inside the rotor, fixed wires were implemented which directed the current from one contact to another, encoding the input.

Rotors also exhibited another property; each rotor had a notch at a fixed position which would allow the levers (pawls) mentioned earlier to ‘step’ the rotor (Hamer 1997). This stepping caused the rotor to rotate by one position. In the machine, three rotors were placed in series to allow current to pass through all three, causing a letter to be scrambled 3 times from one key press. Once a rotor would reach its turnover position (the character shown to the user once the notch position is lined up to the pawl, the latter is not seen by the user of the machine), the rotor to left of the turnover rotor would be stepped as well. The right-most rotor would step every key press, the middle rotor would step with a period of 25 (Grime 2013) (Smart 2016) (one might expect 26, however due to a quirk of the machine known as double stepping, the middle rotor could step twice in a row (See 3.4 for further explanation)). The rightmost rotor would step with a period of 262. This rotational property of the machine allowed for the encoding for a given letter to change each keypress.

The Enigma rotors could be swapped around and placed in any order in the three slots available. Usually, users were given a box containing 5 different rotors (although later models had more and could place them in a choice of 4 slots), [CITE] the choice of which to use formed part of the Enigma’s key.

TODO : Give background of the below table

|  |  |  |  |
| --- | --- | --- | --- |
| Rotor | Encoding (Position A - Ring setting A) ABCDEFGHIJKLMNOPQRSTUVWXYZ | Notch | Turnover |
| I | EKMFLGDQVZNTOWYHXUSPAIBRCJ | Y | Q |
| II | AJDKSIRUXBLHWTMCQGZNPYFVOE | M | E |
| III | BDFHJLCPRTXVZNYEIWGAKMUSQO | D | V |
| IV | ESOVPZJAYQUIRHXLNFTGKDCMWB | R | J |
| V | VZBRGITYUPSDNHLXAWMJQOFECK | H | Z |

Table 1 "Enigma I" rotor encodings (Crypto museum n.d.)

Finally, the rotors had an additional setting known as the ring setting. This allowed the internal wires to be shifted independently from the letter indicators. Whilst generally, the ring setting formed part of the key, it is important to note that the effect of the ring setting did not play much of a part in making the machine more complex as cryptanalytically it has the same effect as the rotation.

Much of the complexity of the Enigma was due to these rotors. Alone, with an *Enigma I* model, there were 1054560 different ways to configure the rotors (See Equation 1)

Equation 1 Permutations for rotors (excluding ring setting)

### Reflector

The reflector was a similar component to that of the rotors and together formed the subsystem that does most of the scrambling. The main differences regarding the reflector are that it does not rotate, and the current does not pass through but rather is ‘reflected’, travelling back in the opposite direction. This reflection allows the reflector to act as a similar substitution cypher akin to the rotors.

The design of the reflector was to enable whole machine to be self-reciprocal, combining encryption and decryption into one operation. This, along with the fact that the reflector could not encode a letter to itself (the design of the rotor prevented the current being passed back through the same metal contact) were crucial flaws exploited (Thimbleby 2016) by the team at Bletchley Park to crack the code during WW2.

TODO : Give background of below table

|  |  |
| --- | --- |
| Reflector | Encoding ABCDEFGHIJKLMNOPQRSTUVWXYZ |
| UKW-A | EJMZALYXVBWFCRQUONTSPIKHGD |
| UKW-B | YRUHQSLDPXNGOKMIEBFZCWVJAT |
| UKW-C | FVPJIAOYEDRZXWGCTKUQSBNMHL |

Table 2 "Enigma I" reflector encodings (Crypto museum n.d.)

### Plugboard

The plugboard formed the final part of the Enigma’s encryption key and was located at the front of the machine. It displayed another representation of the 26-letter alphabet, each of which had a plug socket. These sockets, and the cables that came with the machine allowed two letters to be connected to each other on the plugboard. This created yet another scrambling of the letter, such that if socket A and socket E were connected then any current passing through the plugboard in wire A, would be directed to wire E and vice versa. Any letter left unconnected to another would result in the plugboard having no effect. The plugboards letter swapping effect only occurred twice in each encryption, once at the start of the encryption (after the keypress) and once at the end (before the lamp on the lampboard lights up).

TODO : Diagram / photo of plugboard

## Enigma Machine and Abstraction

### Enigma’s Encryption

Often the best way to understand the mechanisms of the machine is to focus on a single letter’s encryption. As mentioned in 3.2, the user begins with defining the machines key. Upon a keypress on the keyboard, the first notable event is that the rightmost rotor will rotate. Depending on the key, potentially the other rotors will rotate as well but only ever by one position. Once this step has completed, an electrical signal will be induced passing through the plugboard towards the rotors. This input to the rotors will either be the same as the original key press or a different letter due to the plugboard. This electrical signal then passes through rotors right-to-left with the signal being redirected at each rotor. The result of these three redirections is then transmitted to the reflector where the signal is reflected at a different contact. The signal then passes through all three rotors for a final time, this time from left-to-right, before being redirected by the plugboard again. Finally, the signal is transmitted to the lampboard where the cyphertext is displayed. Any plaintext letter can be scrambled up to 9 times before displaying the cyphertext.

By representing each of the components of the Enigma as a transformation, such that represents the plugboard, represents the rotor in the left, middle or right position and represents the reflector. An equation such as Equation 2 can be formulated to describe Enigma’s encryption steps. It is important to note that due to the plugboard’s symmetry, .

Equation 2 The encryption steps of Enigma I

### Abstractions

By abstracting the machine into a logical model, its behaviour and potentially its weaknesses become clearer.

The plugboard acts as a simple cypher that aims to swap two letters such that letters connected by a cable on the physical machine are encoded to each other, otherwise no letter scrambling takes place.

A diagram of lines and dots

Description automatically generated

Figure 1 Wiring diagram to show a plugboard's potential encoding for a reduced alphabet size e.g. A encodes to D and vice versa. Notice the plugboard wiring diagrams tend to create X shapes.

The reflector acts as a substitution cypher with some additional constraints being self-coding and reciprocal coding. Like other components this can be drawn using a wiring diagram that shows the input and output from the same side.

A diagram of a diagram

Description automatically generated

Figure 2 Wiring diagram to show an example reflector encoding for a reduced alphabet size e.g. A encodes to F and vice versa.

The rotors of the machine are substitution cyphers with no additional constraints. They take an input letter and produce either the same or a completely different letter. In addition, the behaviour of the rotor can be described as a function.

TODO : Check this formula with tutor, maybe make it more descriptive?

Equation 3 A function to represent the encoding behaviour of the rotor where x and x` are letters and represents the rotor’s ring setting and rotation.

A rotor can be described using a wiring diagram, but multiple diagrams are needed to convey the effect of the rotor stepping as each time this rotation takes place, the encodings will be shifted.

A diagram of lines and dots

Description automatically generatedA diagram of lines and dots

Description automatically generated

Figure 3 Two Wiring diagrams to show an example rotor encoding. The diagram on the right shows the same rotor as on the left, but with a rotation of 1. The left diagram shows F encoded to F and the diagram on the right shows E encoded to E. The “wire” is still the same, but the rotation of the rotor will lead to an F input/output being interpreted as E input/output.

By abstracting all the electrical and mechanical features of the machine, logical diagrams demonstrating the letter scrambling that takes place in the Enigma machine can be created with much more ease. Similar diagrams can be seen in work by Smart (Smart 2016) and Thimbleby (Thimbleby 2016). The diagrams from Figures 1, 2 and 3 represent each component of the machine and can be combined to create a representation of the entire Enigma machine (See Figure 4).

A diagram of a network

Description automatically generated

Figure 4 A wiring diagram representing a single state of an Enigma I machine. Input is received on the right-hand side before being scrambled by components in the following order: plugboard, rotor III, rotor II, rotor I, reflector UKW-B, rotor I, rotor II, rotor III, plugboard. The electrical signal received back from the plugboard represents the encoded letter. The names of each rotor/reflector do not match the encodings but are given as an example.

A diagram of a network

Description automatically generated

Figure 5 An example of an encryption/decryption taking place in an Enigma I machine. In this case A has been encoded to C.

## Design Flaws

## Possible Solutions

## Similar Products

A section to show the other types of enigma simulators that can be found on the internet which form the inspiration for my project. Explain how I would like to take this further.

# Design & Implementation

## Tools and Technologies

Description and list of tools used within this project.

* Java to create the application and the model.
* JavaFX – graphics package used to create the GUI + Visualizations.
* GitHub – version control and development tasks
* Maven – build project across multiple platforms.
* XML – Custom component definitions
* Gluon Scene Builder – Used for UI building and FXML creation.

## Enigma Model Design

Talk about the idea of the package, this would be a good place for UML diagrams and a description of how this model is going to work.

### Keyboard & Lamp board

Explanation of how this part does not really need to be modelled, and instead can be represented as input and output.

### Reflector

Description of how this will be modelled.

### Rotors

Description of how this will be modelled.

### Enigma

Description of how this will be modelled.

## Design of the Application

Explain how the application will work, show diagrams in order to show how the interfaces will interact with the model.

### Command Line Interface

A fairly short section, just explain how the interface works and what information will be shown.

### GUI

Larger section explaining the various design stages of the GUI, as well as explaining the MVC implementation. Important to talk about design choices in this section as this is the bulk of the project.

### Visualization

A second part to the GUI, explain the design choices and implementation as well as a section “interpreting the diagram.”

# Results

# Evaluation

## Model Evaluations

* Test Coverage
* Actual German messages
* Permutations compared with improved model.

## GUI Evaluation

* Usability
* Cross-platform

# Conclusion

# References

n.d. *Crypto museum.* Accessed 2024. https://www.cryptomuseum.com/index.htm.

Grime, J. 2013. "Maths from the talk "Alan Turing and the Enigma Machine"."

Hamer, David H. 1997. "Enigma: Actions involved in the ‘double stepping’of the middle rotor." *Cryptologia 21, no. 1* 47-50.

Owen, Jaren. 2021. "How did the Enigma Machine work?" *Youtube.*

Smart, Nigel P. 2016. "The enigma machine." *Cryptography Made Simple* 133-161.

Thimbleby, Harold. 2016. "Human factors and missed solutions to Enigma design weaknesses." *Cryptologia 40, no. 2* 177-202.

van Manen, Dirk Jan, and Johan O.A. Robertsson. 2016. "Codes and ciphers." *Part I: Decoding nature’s disorder: GeoExpro 13* 38-41.

# Table of Figures

## Figures

[Figure 1 Wiring diagram to show a plugboard's potential encoding for a reduced alphabet size e.g. A encodes to D and vice versa. Notice the plugboard wiring diagrams tend to create X shapes. 7](#_Toc157955103)

[Figure 2 Wiring diagram to show an example reflector encoding for a reduced alphabet size e.g. A encodes to F and vice versa. 8](#_Toc157955104)

[Figure 3 Two Wiring diagrams to show an example rotor encoding. The diagram on the right shows the same rotor as on the left, but with a rotation of 1. The left diagram shows F encoded to F and the diagram on the right shows E encoded to E. The “wire” is still the same, but the rotation of the rotor will lead to an F input/output being interpreted as E input/output. 9](#_Toc157955105)

[Figure 4 A wiring diagram representing a single state of an Enigma I machine. Input is received on the right-hand side before being scrambled by components in the following order: plugboard, rotor III, rotor II, rotor I, reflector UKW-B, rotor I, rotor II, rotor III, plugboard. The electrical signal received back from the plugboard represents the encoded letter. 9](#_Toc157955106)

## Tables

[Table 1 "Enigma I" rotor encodings (Crypto museum n.d.) 6](#_Toc157955113)

[Table 2 "Enigma I" reflector encodings (Crypto museum n.d.) 7](#_Toc157955114)

## Equations

[Equation 1 Permutations for rotors (excluding ring setting) 6](#_Toc157955121)

[Equation 2 A function to represent the encoding behaviour of the rotor where x and x` are letters and represents the rotor’s ring setting and rotation. 8](#_Toc157955122)