

# Project: Enigma Machine and Turing-Welchman Bombe

Maria Camila REMOLINA GUTIÉRREZ  
maria.remolina\_gutierrez@telecom-sudparis.eu

Advisor: Prof. Eric RENAULT

June 19, 2019

## Abstract

This is the work proposal to follow in the course Project as part of the master M1 in Computer Science and Communication Networks at Télécom SudParis. The goal is to implement the Enigma machine used in World War II, followed by the Bombe machine that breaks the cipher, created by Alan Turing and Gordon Welchman at Bletchley Park.

## 1 Introduction

The enigma machine is a cipher machine used by the Nazi Germany during World War II in order to send secret coded messages. It was initially a commercial machine bought by banks and businesses. But then the military took it and added an extra security layer called plugboard. It was innovative at the time because it was not a substitution cipher, i.e. the same letter can get different results after encryption.

The machine works by a combination of moving rotors and inside wiring as seen in Fig. 1. The way to use it is that when the sender types a message, a bulb lights up indicating the correspondent coded letter. Then, in order to decode, the receiver types the coded message he received and the initial message lights up on the board. In war times the coded messages were transmitted over morse code.

In the military version of the Enigma Machine there were 5 possible rotors to pick from, each with 26 possible positions, then there were 10 possible switching in the plugboard that could choose a pair from the available 26 letters. That accounts for  $15896255217826360000$  different combinations for the setting of the army Enigma Machine [2]. The initial setting for each day was given to the bases in a sheet of paper that



Figure 1: Enigma Machine - Military Edition

had the monthly configurations of the machine. So they changed the setting everyday.

So in order to decipher the code, you needed either to have the code sheet or to break the message. This latter was what the scientists at Bletchley Park did; and what I will try to recreate in this project.

## 2 General Goal

To understand the code breaking process behind the enigma cipher.

## 3 Specific Goals

- To implement the Enigma machine with software
- To implement the Bombe machine that breaks the Enigma cipher
- To understand the mathematical and probabilistic techniques used to break a cipher with limited time and computational resources.

- To understand the weaknesses exploited that allowed to break the Enigma cipher

## 4 Implementation

I implemented a project in the Python programming language that replicates both the Enigma Machine and the Bombe Machine. I explain in the following subsections the details of each of them.

### 4.1 Enigma Machine

There are different models of the Enigma machine that were developed through time. In this project I chose one of the latest, i.e. with more complicated settings. This is the model M3 & M4 Naval (from February 1942) that has 8 possible rotors [8, 9].

#### 4.1.1 Enigma's algorithm

The enigma machine consists on a set of physical connections that allow electrical signals to travel through inner cross wirings that move place every time a key is pressed. In the most simple form we could express that any typed key in the keyboard is transformed to another letter that lights up in the bulbs panel. So we could say:

$$\text{SIGNAL}_{\text{OUT}} = P(S(P(\text{SIGNAL}_{\text{IN}})))$$

$$S(\text{SIGNAL}) = R_1(R_2(R_3(F(R_3(R_2(R_1(\text{SIGNAL})))))))$$

Where  $P$  represents the Plugboard;  $S$  represents the Set of rotors and reflectors;  $R_i$  represents the  $i$ -th rotor (in this case there are 3 but there could be more); and  $F$  represents the reflector. All of this process can be graphically understood with the image below (Fig. 2):

The machine has a symmetric property so in order to decipher the message one has to put the same rotor settings as when the person typing the message started. So this leads us to the configuration of the machine, i.e. the input.

#### 4.1.2 Input

The machine has different settings that changed each day and that were communicated via code sheets like the one in Fig. 3.

I handle this as a configuration text file in my implementation. I will describe the components as follows:

- Rotor selection and order: Depending on the machine there are up to 8 different rotors (I-VIII) from which to choose 3 or 4. So an input is which of them and in which permutation from left to right. Example - V IV I.

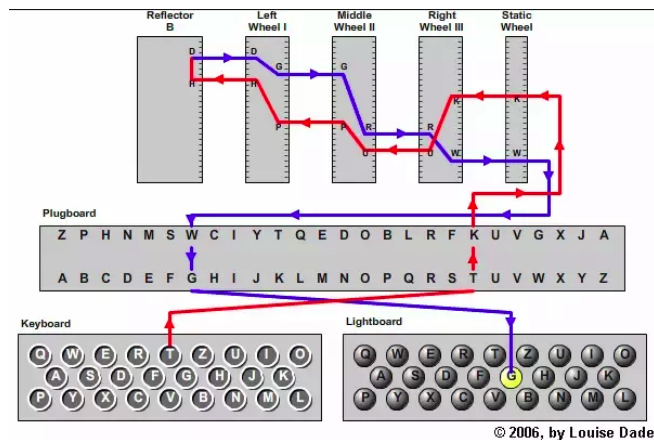


Figure 2: Enigma Machine Wiring

Geheim!  
Nicht ins Flugzeug mitnehmen!

**Sonder-Maschinenschlüssel BGS**

08 \*

Datum	Walzenlage	Ringstellung	Steckerverbindungen	Keuengruppen
31.	I II V	10 14 02	BF SD AY HG OU QC WI RL XP ZK	yqv vuc xxo gvf
30.	V IV I	04 25 01	DI ZL RX UH QK PC VY GA SO EM	mgy vts gvt csx
29.	III V II	13 11 06	ZM BQ TP YX FK AR WH SO NJ DG	aky vdv oyo tzt
28.	I III II	09 16 12	NE MT RL OY HV IU GK FW PZ XC	nfh vco tur wnb
27.	III II I	06 03 15	BF GR SZ OM WQ TY HE JU XN KD	bec jmv vtp xdb
26.	I III V	19 26 08	GS VD CQ LE HI BO JP UZ FT RN	wvu yem buz rjk
25.	II I IV	05 01 16	KA ZH QP GR MF LJ OT EN BD YW	ktv muq cqm cpm
24.	III II IV	22 02 06	PI KM JB YU QS OV ZA GW CH XF	zcd iwo urp glg
23.	IV III II	08 11 07	SX TD QP HU FB YN CO IK WE GZ	epm mgz vqg vsm
22.	I V II	13 02 26	GP XH IW BO NU MD SA ZK QR LT	aam mvv jqq wqm
21.	IV I V	17 24 03	XC AQ OT UZ HD RG KM BL NS JW	ltl blu frk xrh
20.	IV I III	15 22 12	PO TV QC ZS EX WR BJ DK FU LA	non lic oxr usr
19.	V I III	13 24 21	HA GM DI VK JP YU EF TB ZL XQ	ecd ciq uvr ppt
18.	IV V I	23 09 20	XM PZ SQ GR AJ UO CN BV TM KI	fjh sts uqu cft
17.	III II V	21 24 15	UT ZC YN BE PK JX RS GF IA QH	oub eci pyf rqi
16.	IV III V	07 01 13	IN YJ SD UV GF BH TK QE AR OP	kex paw flw onw
15.	I IV II	15 04 25	TM IJ VK OY NX PR WL GA BU SF	sdr pbu byv khb
14.	III II IV	10 23 21	WT RE PC FY JA VD OI HK NX ZS	mhz lff lnq giy
13.	V I II	14 04 12	AN IV LH YP WM TR XU FO ZB ED	rqh ucm ldi ods
12.	II V I	07 19 02	HR NC IU DM TW GV FB ZL EQ OX	asy xza uvo fmr
11.	I V IV	13 15 11	NX EO RV GP SU DK IT FY BL AZ	gyd iuq oob vef
10.	V II I	09 20 19	FN TA YJ EO EG PC VD KI XH WZ	pyz ace pru uyc
9.	I IV V	14 10 25	VK DW LH RF JS CX PT YB ZG MU	nyh fbd ohs jrp
8.	IV V I	22 04 16	PV XS ZU EQ BW CH AO RL JN TD	tck rts nro mkl
7.	V I IV	18 11 25	TS IK AV QP HW FM DX NG CY UE	mhw lwb mdm ybe
6.	IV I III	02 17 20	KZ FI WY MP DS HR CU XE QV NT	uwu ydk lrh mgd
5.	I V IV	26 09 14	VW LT PB FO ZK GS RI QJ HM XE	suv tsy nfp yjc
4.	IV III V	07 01 12	QS YA XW KR MP HT DU OV CL FZ	uby usi mhh mwb
3.	I II V	05 16 03	FW DL NX BV KM RZ HY IQ EC JU	tns von gtw axl
2.	III I II	12 22 17	DW UO PY GR FS EQ KT CL AI ZB	smz lbl pkc sym
1.	I III II	04 18 06	ZN OM CR UI KP WQ SE JV LX TF	ghr vqv cya ayl

DECLASSIFIED  
Authority NND 68303  
By NARA Date 11/4/94

Figure 3: Enigma Machine Wiring

- Rotor's ring settings (*Ringstellung*): Within each rotor there is an internal ring that maps position to a letter, but it can be switched so that the internal wiring mapping is offset. Example - N D V.
- Rotor's initial positions (*Grundstellung*): Within the machine, when locating the rotors, they have to be in a particular initial position. This is achieved by inserting the rotor and turning it to the respective starting point. Example: D Y A.

- Plugboard pairs: This corresponds to the symmetric correspondence of letters in the other plugboard panel. They are typically 10 pairs.
- Reflector selection: As with the rotors, there are 4 different reflectors from which to choose, all with different cross-connections.

Apart from all of these values that changes from user to user, I also have an input configuration file that loads the historically accurate wiring of each reflector and rotor. For the rotors there is an additional parameter called the turnover point. The rotors work as a clock in the way they rotate. The right-most represent the second, the middle the minutes, and the left-most the hours. However they don't turn all in the same letter ('Z' as one would expect), they all each have their own turnover point that triggers its right partner rotation. Those turnovers are obtained from [6]. Finally the message to cipher is also input as a text file.

#### 4.1.3 Processing

As for the processing of the plain text, here is a pseudo code with the simplified process:

```
for letter in text:

    # advance rotors according to their own position ,
    # and turnover points
    step_rotors()

    # enter plugboard switch , if no correspondance there
    # is no connection , so letter doesn't change
    plugboard_letter = plugboard.get(letter , letter)

    # enter rotors (inwards - right to left)
    rotor_names_inverted = rotor_names[::-1]
    pin_in = ALPHABET.index(plugboard_letter)

    for rotor_name in rotor_names_inverted:
        rotor = rotors[rotor_name]
        pin_in = rotor.process_inwards(pin_in)

    # enter reflector
    pin_out = reflector.reflect(pin_in)

    # enter rotors (outwards - left to right)
    for rotor_name in rotor_names:
```

```

rotor = rotors[rotor_name]
pin_out = rotor.process_outwards(pin_out)

# enter plugboard switch
rotor_letter = ALPHABET[pin_out]

# calculate ciphered letter
ciphered_letter = plugboard.get(rotor_letter, rotor_letter)

```

Some things to be aware of in the processing are that the rotors step before ciphering, not after. Also, the turnover points need to be checked each time the rotors step, some of them even have 2 different turnovers.

#### 4.1.4 Output

The output of the machine is straightforward, it corresponds to the cipher letter of each key press after going through the transformations. And for a whole text is just the concatenation of those letters. If this output is to be typed again in the machine with the same initial configuration, the operator on the other side of the line will obtain the original plain message.

#### 4.1.5 Challenges

The biggest challenge in implementing the Enigma machine is knowing exactly how it worked. Even though there is a great amount of information online, the depth of it is not that significant. Most of the resources tend to shallowly explain the behavior but there are several subtleties and details that were hard to find. I put in the references the most trustworthy and complete information I found on the matter [3, 4, 5, 6, 7, 10].

Another challenge was the translation of this information, because the original machine settings are in German. This implies that there are multiple acceptable translations for several parts of the machine that might be opposite among different sources. A concrete example of this goes within the rotor, where there are 2 different settings called *Ringstellung* and *Grundstellung*. They refer to the ring setting within each the rotor and to the ground setting or offset of the rotor within the machine, respectively. However figuring out which one was which presented a problem because different sources referred to them by different names, sometimes even opposite. So I had to solve this by recurring to the German words.

Finally, I would like to refer as well to different simulators that are found on the internet in which I could get a phenomenological understanding of the machine. I was able to try out different configurations and learning its way to function. I also confirmed

my solution against these simulators in order to test my implementation. This was a positive match.

- Cryptii Simulator: <https://cryptii.com/pipes/enigma-machine>
- Louise Dade Simulator: <http://enigma.louisedade.co.uk/enigma.html>

## 4.2 Bombe Machine

### 4.2.1 Enigma Exploits

In order to break the enigma, mathematicians Alan Turing and Gordon Welchman, with the collaboration of the staff at Bletchley Park, design the Bombe Machine. It exploited the Enigma Machine's flaws to crack the Nazi's communications. However, the machine itself was not all, it exploited behavioral patterns in the communication as well. Here I list the main weaknesses that lead to deciphering the Enigma:

- The biggest flaw of the Enigma Machine is that because of the internal wiring and design, a letter can never be encrypted to itself. This opened the door to cryptoanalysts to locate possible message deciphers. [11]
- Another breakthrough was when they realized about specific patterns that would repeat themselves in a conversation. For example the daily weather forecast, the heil Hitler salute, or war jargon. These parts of the conversation typically followed a standard format that allows to create "cribs", an intelligent guess of part of the code. [12]
- In addition to the previous facts, sometimes reckless operators forgot to change the daily settings, which lower the odds of combinations and gave more data to the analysts.

### 4.2.2 Bombe's algorithm

- with those typical messages we can scan the transmissions and locate where it is within the text (where no letter ciphers to itself)

- with all the possible positions we create a set of hypothesis called a menu, a diagram like fig x. from there, we need to find an enigma configuration that corresponds to the menu. that was the role of the bombe <https://www.youtube.com/watch?v=2dKG21u2aSo>

### 4.2.3 Input

### 4.2.4 Processing

### 4.2.5 Output

### 4.2.6 Challenges

-getting info, very sensitive, most of it classified. Some of it released in 2010, less than 10 years ago -less than 200 bombes, as opposed to enigma machines that were even commercial

...

Finally, and as in the previous step, I got some insights from two Bombe simulators online. However as the is very complex, using them is not as straightforward and required a lot of learning. After all, each bomb had several operators. So I just had a glimpse on their configuration files, specially the menu input. Here are the references:

- Ekhall & Hallenberg Simulator: <http://www.lysator.liu.se/~koma/turingbombe/> [13]
- 101 Computing Simulator: <https://www.101computing.net/turing-welchman-bombe/>

## 4.3 Code

The code of both machines is available in the project attached to this report. The structure is as follows:

### PICTURE OF CODE STRUCTURE

There is a program called `example.py` that shows how to use the programs and when executed, outputs the result of both enigma and bombe machines to their respective results files. It also outputs to console like this:

## 5 Discussion

- historic relevance - not only breaking it, it needed strategy, not letting them know they cracked the code - conclusions, goals achieved

### 5.1 Future Work

## References

- [1] A. Hodges. *Alan Turing: The Enigma*. Princeton, N.J: Princeton University Press (2012).
- [2] Numberphile. *158,962,555,217,826,360,000 (Enigma Machine)*. YouTube (2013).



```

mcrg@thegirlwhocan:~/Documents/GitHub/enigma_bombe/enigma_bombe/src$
python example.py
-----
ENIGMA
-----
Clear text:
HIERICIAMMARIAANDTHISISMYPROJECTIPRESENTHERETHEENIGMAMACHINEUSEDINWOR
LDWARTWOASWELLASTHEBOMBEMACHINETHATBREAKSTHECIPHER

Ciphred Text:
SEZALVGDPKKDBSNUMMPQLOAIXTDLANSIXVMAJFAJCFNKRZURAYCYGTUCNARZDXFLZJWQ
AREVTDJNXNQDXSOQSBUVXQJJOPOHUGCMHELD FBSFDJFWTVRSXNJ

-----
BOMBE
-----
Bombe finished and it output the possible plugboards to the file: ..
/data/example_plugboards.txt

```

Figure 4: Example Code Standard Output

- [3] D. Rijmenants. *Technical Details of the Enigma Machine*. Obtained from: <http://users.telenet.be/d.rijmenants/en/enigmatech.htm>
- [4] T. Sale. *Military Use of the Enigma*. Obtained from: <https://www.codesandciphers.org.uk/enigma/enigma3.htm>
- [5] Cryptomuseum. *Enigma Cipher Machines*. Obtained from: <https://www.cryptomuseum.com/crypto/enigma/index.htm>
- [6] Wikipedia. *Enigma Machine*. Obtained from: [https://en.wikipedia.org/wiki/Enigma\\_machine](https://en.wikipedia.org/wiki/Enigma_machine)
- [7] Wikipedia. *Cryptanalysis of the Enigma*. Obtained from: [https://en.wikipedia.org/wiki/Cryptanalysis\\_of\\_the\\_Enigma](https://en.wikipedia.org/wiki/Cryptanalysis_of_the_Enigma)
- [8] L. Dade. *Enigma Machine: How It Works*. Obtained from: <http://enigma.louisedade.co.uk/howitworks.html>
- [9] Wikipedia. *Enigma Rotor Details*. Obtained from: [https://en.wikipedia.org/wiki/Enigma\\_rotor\\_details](https://en.wikipedia.org/wiki/Enigma_rotor_details)
- [10] G. Ellsbury. *The Enigma and The Bombe*. Obtained from: <http://www.ellsbury.com/enigmabombe.htm>
- [11] Numberphile. *Flaw in the Enigma Code*. YouTube (2013).
- [12] Computerphile. *Tackling Enigma (Turing's Enigma Problem)*. YouTube (2013).
- [13] M. Ekhall, F. Hallenberg. *US Navy Cryptanalytic Bombe - A Theory of Operation and Computer Simulation*. Proceedings of the 1st Conference on Historical Cryptology, pages 103-108, Uppsala, Sweden, 18-20 June, 2018.