Enigma and BOMBE simulation

Student Name: A.L. Gillies

Supervisor Name: M. Johnson

Submitted as part of the degree of MEng Computer Science to the

Board of Examiners in the School of Engineering and Computing Sciences, Durham University

Abstract —

Context/Background - Can antiquated ideas and designs be brought up to speed in the modern day? The aim of this paper is to show the plausibility of this using the Enigma and the BOMBE, two cryptographic devices from the second world war, these will be modernised and tested to prove this hypothesis.

Aims - Using the known execution speed of the antique Enigma and BOMBE, the aim of this paper is to show, through implementation and testing of a modern interpretation of the Enigma and BOMBE, the speed up that has occurred since the inception of both.

Method - After both the Enigma and BOMBE are implemented the BOMBE will then be improved on using parallelisation techniques. Each will then be tested. The time taken for a standard Enigma machine to encrypt, the computerised version to encrypt and the parallelised version to encrypt will be compared to one another to give a good indication of the improvements over time. Another comparison will be made between the standard Enigma, the computerised Enigma and the parallelised Enigma, this should re-enforce the indication of computer improvements.

Proposed Solution - The Enigma machine will be implemented through C++ code, this will form the baseline for a modern interpretation of the machine as it will not run anything in parallel. This code will then be parallelised using different standardised tools and techniques, these will then be used as the final comparison to show the significant increases in performance that this seemingly antiquated idea can expect across different inputs.

Keywords — Enigma, BOMBE, Parallel, Modern, Computation, Evaluation, Reinterpretation, Comparison, C++, Antiquated.

I INTRODUCTION

A Description and Purpose

This project is to be an investigation into the possible improvements that can be gained on an antiquated technique by parallelisation against unparallelised and a time accurate representation. This will be done by first creating a modern equivalent of the antiquated technique. The modern equivalent will then be improved upon using parallelisation techniques. This is to be done on the BOMBE using the Enigma machine.

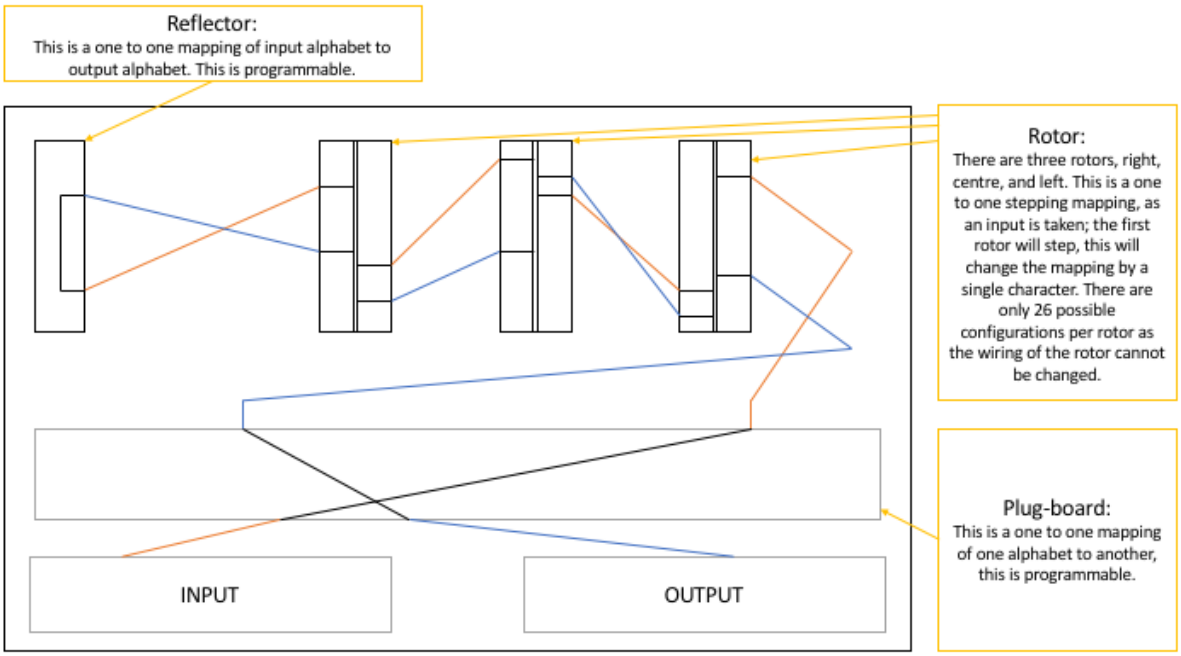
B What is the Enigma Machine?

The Enigma machine was an early to mid 20th century cryptography device, it was used in both commercial and military applications. Most famous for its use by Nazi Germany during the second world war, the electro-mechanical rotor device was made up of three rotors, a plugboard, and a reflector. The user interface for the device is a keyboard and a lightboard, once a letter is pressed on the keyboard, it is then encrypted and the corresponding encrypted letter will light up on the lightboard.

The device encrypts any given letter is the same way. Once a letter has been pressed on the keyboard a signal is passed through the plugboard this is a plug-based one to one mapping of all the letters on the keyboard to a unique letter of the alphabet. This output is then passed to the first of three rotors, a rotor is a set one to one mapping that will then step to the next letter once used. The order of either alphabet on the rotor cannot be changed but the mapping of the input letter to the output letter will change after each time it is used, this is called the stepping mechanism. The first rotor will step each time a letter is pressed and the second will only step if the first rotor has done a complete rotation, the third will only step if the second has done a complete rotation. The final component is the reflector this is similar to the plugboard that it is a one to one mapping but has the issue that it cannot map a letter to itself. After the reflector the signal is sent back through all three rotors and then through the plugboard to light up one of the letters on the lightboard.

The device has several settings that can be changed. Firstly the plugboard is reprogrammable, the plugboard is a one to one mapping between two alphabets. Secondly, the device typically has three rotors that can be used at any one time, there are eight rotors that can be put in any one of the three position, left, right and centre. Having chosen three of the rotors and their positions in the device, the initial rotation of each also has to be chosen. Finally, the reflector is also programmable, this is another one to one mapping between two alphabets.

Shown below is a diagrammatic representation of the Enigma Machine.



C What is the BOMBE?

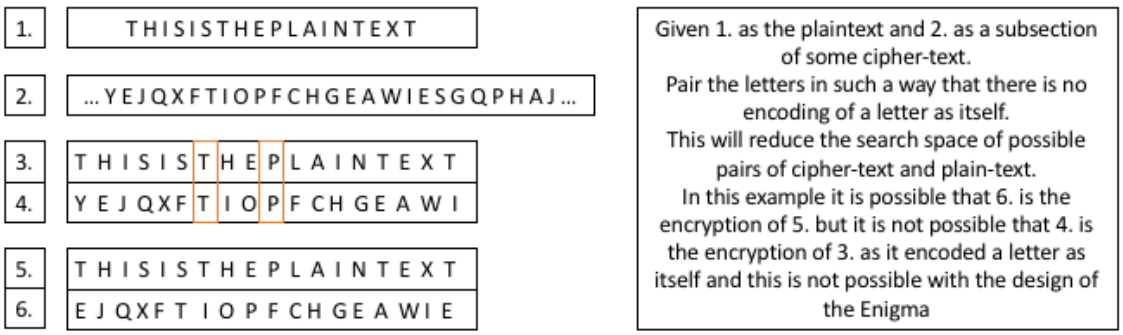
The BOMBE is the antithesis of the Enigma machine, it was specifically designed by British cryptologists in 1939 to counter the Enigma encoded messages that were used by the Axis at the time. The BOMBE was designed to to discover the settings used by the Enigma operators that were changed daily.

The BOMBE was an electro-mechanical device that replicated the action of several Enigma machines wired together; the British BOMBE contained 36 Enigma representations. These representations were not exact clones, they were made up of seven parts; namely six rotors and a reflector substitute. The first three rotors behaved exactly as they do in the enigma, and input entering the first rotor, being remapped, then the result is fed into the second and remapped and so on until the reflector substitute is reached, this would have the same mapping as its Enigma counterpart but does not feed it's result back into rotor three, it instead feeds the result into rotor 4 which is a refection of rotor three, this has the same effect to the mapping but allowed for the BOMBE to be easier to design and manufacture. Once rotor four had remapped the input, the output would go on to rotor five which is a reflection of rotor two, and so forth. The BOMBE does not have the plugboard mapping integrated as this was done separately.

The BOMBE was based on the probably-phrase attack; it is possible to exploit the relationship between a section of plaintext that was either guessed or already known, and the ciphertext to which it might correspond. The idea of the BOMBE was to remove any superfluous Enigma set-ups using the fact that the Enigmas reflector could not encode a letter as itself. Once the 'crib' or the guessed plaintext had been shown to be the test ciphertext then the settings of the Enigma would be known and any other messages sent by the Enigma machines would be known for the rest of the day as the British cryptologists knew the settings used for the current messages and it was just a case of feeding the ciphertext into their own Enigma machine with the current settings and reading the output.

The crib is shown to correspond to the ciphertext through several steps. The first is to use the fact that the Enigma had a flaw in the encryption process to full effect, if at any point the same letter is in the same position in the ciphertext and crib then this encryption is invalid thus the crib will be moved one letter over and retested, this removed a lot of unnecessary testing in the later stages.

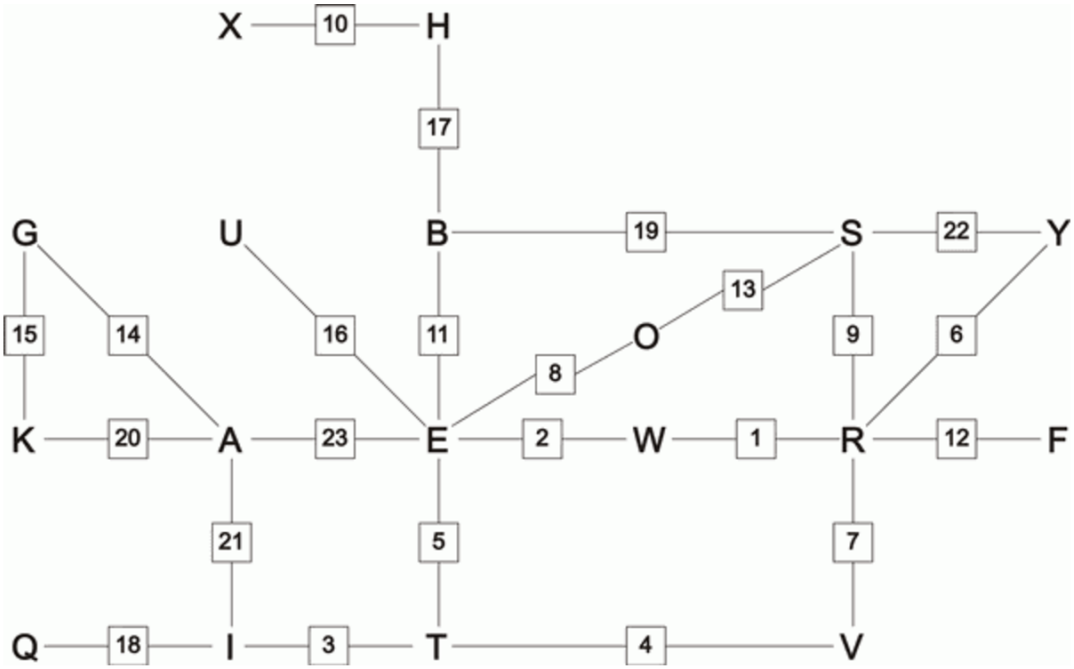
To give a visual representation of the first stage.



The second stage is to create a connection network for each step, this is done by pairing and numbering a possible ciphertext-plaintext pair, the first paring will be labelled as number

one, the second as number two, and so forth until there are no more pairs. After we have these pairings, we will create the network where at any step, indicated by the pair number, there is a map from the corresponding plain-text character and cipher-text character. This is then done for every character pairing in the plain-text, cipher-text pairing.

An example of what the connection network could look like is shown bellow.



Using the connection network the search space can be reduced by using the implications of some transformations on others. We first must recall that each transformation consists of two 'stecker swappings' and a 'scrambler encryption' these are the set of rotor transformation and reflector respectively. Using the connection network above, here is an example of the search space reduction.

For example, let us test the hypothesis that E is mapped to K through the first three rotors, also known as E is "steckered" to K. Now consider how E is transformed into A at relative position 23, this is shown on the connection network above.

If E is "steckered" to K, K will be input to the reflector or "scrambler" at relative position 23 which will output some other letter L1. Since we know that E transforms into A at relative position 23, we know that A must be "steckered" to L1.

But if A is "steckered" to L1, L1 will be input to the "scrambler" at relative position 21 which will output some other letter L2. Since we know that A transforms into I at relative position 21, we know that I must be "steckered" to L2.

But if I is "steckered" to L2, L2 will be input to the "scrambler" at relative position 3 which will output some other letter L3. Since we know that I transforms into T at relative position 3, we know that T must be "steckered" to L3.

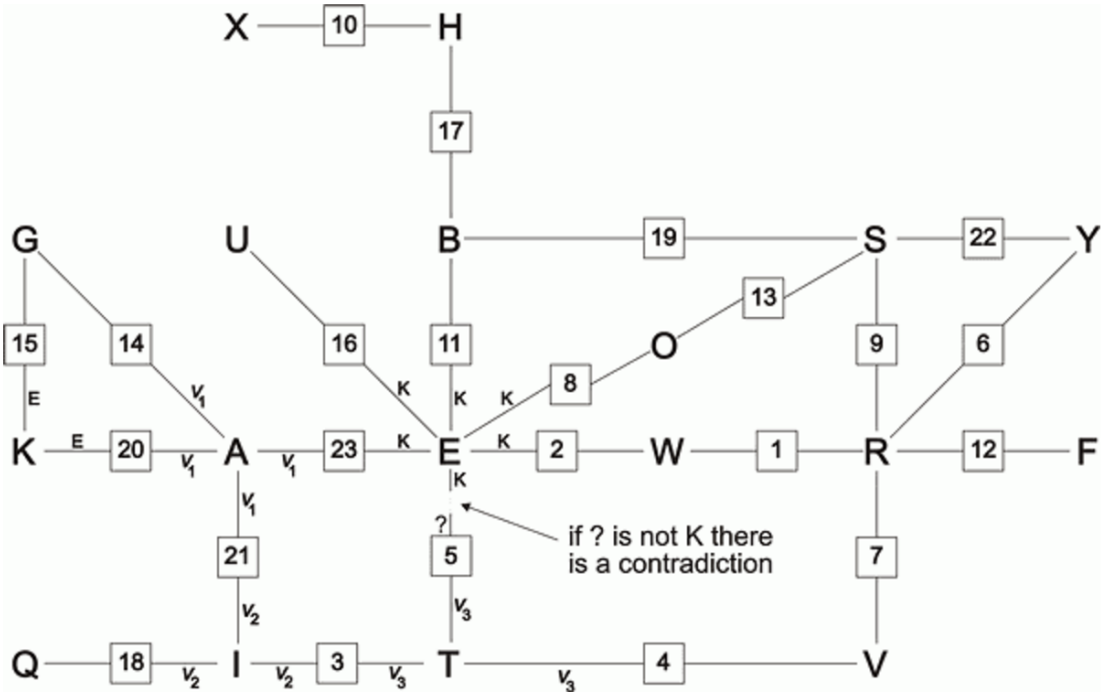
But if T is "steckered" to L3, L3 will be input to the "scrambler" at relative position 5.

Now consider the output of the ”scrambler” at relative position 5 when it receives L3 as its input. Since we know that T transforms into E at relative position 5, we know that the output letter of the ”scrambler” at relative position 5 must be ”steckered” to E.

Suppose, when given L3 as its input, the ”scrambler” at relative position 5 outputs J. By our original hypothesis E is ”steckered” to K, but by our chain of implications E is ”steckered” to J, so in this case E is ”steckered” to both K and J. But the construction of the Enigma does not permit one letter to be ”steckered” to two letters, so E cannot be ”steckered” to both K and J, so our original hypothesis has resulted in an impossible consequence and so must be false.

But now suppose, when given L3 as its input, the ”scrambler” at relative position 5 outputs K. Then by our original hypothesis E is ”steckered” to K, and by our chain of implications E is ”steckered” to K, so in this case E is ”steckered” only to K, thus our original hypothesis has resulted in a consistent consequence and so may be true.

The altered connection network bellow shows the hypothesis that E is ”steckered” to K via the the steps of implication that we have been through.



A further method to reduce the search space was found by Gordon Welchman, a contemporary of Alan Turing. The reciprocal property of the stecker meant that if, for example, W was steckered to Q then Q must be steckered to W, and that is was possible to reflect this property in a device which came to be called ’The Diagonal Board’. A steckering is a complete ’set’ of rotor transformations in one direction consisting of three rotors.

The effect of the diagonal board, when attached to the BOMBE, was to to increase the feedback into the double ended scramblers and thus reduce the the necessity of obtaining loops in the crib-ciphertext pairing. This in turn made it possible to use shorter ’menus’, the connection diagram, which were less likely to include a turnover of the Enigmas middle rotor during the enciphering of the plaintext.

At this point the search space is sufficiently reduced to allow for the BOMBE to be used and have a result within a reasonable time. The top rotor was set to rotate at 120rpm with the second rotor 1/26th of the speed and the third 1/26th of that. At the set speed the BOMBE will exhaustively search the problem space and will either reach the end of the search space or reach a 'stop'. A 'stop' was known as a setting where the transformations determined by the rotor order, rotor position, and some stecker swapping are consistent with the transformations that yielded the ciphertext from the crib.

The BOMBE would 'know' that it had reached a stop when all output wires through the test registry were 'live'. This test registry was present on the diagonal board. At this point all the rotor positions were known.

D What are the parallelisation techniques?

With the way the BOMBE works, over a set alphabet in a sequential nature, parallelisation would be a marked improvement as it would allow for a search over the set alphabet in a non-sequential way and thus improve speed. There are several techniques that could be employed to improve the speed of the search. Single Instruction Multiple Data is a idea that is rooted in doing the same instruction to multiple data sources, for example, if one wanted to add six to a lot of numbers, an unparallelised program would load the first number then add to it and send it back, a parallelised program would load as many of the numbers as it could, add six to all of them in one go then send them back. The SIMD technique is characterised by vectorised loads and saves. This would be a possible improvement as the BOMBE doesn't change it's instruction just the data it applied to. Multiple Instruction Multiple Data is another parallelisation technique that is much harder to implement but has the added benefit that different operations can be done to different datasets so two almost unconnected tasks can be done simultaneously.

E Deliverables

E.1 Minimum Objectives

- Simulate Enigma and show its correctness, by encrypting plaintext and comparing this to its known ciphertext to show this correctness, this should be done over a significant database of plaintext and ciphertext.
- Simulate BOMBE and show its correctness, taking some of the aformentioned stock ciphertext and running the BOMBE on this input such that the plaintext is output as the result.

E.2 Intermediate Objectives

- Prove the correctness of the Enigma simulation by encrypting plaintext and comparing this to its known ciphertext to show this correctness. This should be done over a significant database of plaintext and corresponding ciphertext.
- Prove that the BOMBE simulation is correct by taking all of the aformentioned stock ciphertext and running the BOMBE on this input such that the plaintext is output as the result.
- Both simulations should also be proved to be effective on random inputs and also in relation to one another; the Enigma machine should be run on a random input and the output should then be used as the input on the BOMBE which should in turn return the random plaintext.
- Evaluate average computational time on inputs of a set size.

E.3 Advanced Objectives

- Increase the scope of the BOMBE simuation by running the simulation in parallel. This means that different parts of the ciphertext could be run in parallel or the whole cipher text being run with each parallel part testing different configurations of the Enigma machine.

II DESIGN

A Tools

The only tools I will need are the Intel compiler based on C++, this will allow unparallelised code to be written in C++ meaning that it can be compiled by a standard C++ compiler which is more widely available for a wider range of platforms. The Intel tools will allow for parallelisation techniques to be applied to the BOMBE simulation, at this point only the Intel compiler will be able to compile the code to fully utilise the additions to the code that parallelization require. C++ is the chosen base language mainly because it has an expansive library of parallel tools, this is the Intel compiler. Bash scripts will also be used, short scripts to automate IO and calling, to make the process of running and decoding easier. Bash scripts will be used for simplicity on the system and will not be a requirement for the system to work.

B Development and Maintenance

The project will be designed and produced in an agile manner, it will also incorporate a modular progression. Each system, Enigma and BOMBE, will be produced independent of one another. This will allow verification of correctness to be acquired at several different stages throughout the implementation and not as a final progression. The Enigma machine will be designed and implemented first, it will then be verified. Following that the standard BOMBE will be implemented and using the already verified Enigma as a means to show to correctness of the BOMBE, it will also be tested on known (ciphertext, plaintext) pairs as well. This will lead onto the final stage, the parallelisation and testing of the BOMBE, it will continuously be verified during this process to produce a fast and correct BOMBE implementation that will then be tested against it unparallelised counterpart, leading to a conclusion.

The maintenance of this system will most likely not extend far past the deadline date, as by this point both systems will be fully working in the latest version of the intel compilers and will not be changed for any updated version of the compilers. The systems will work on any operating system that can install the latest version of the intel compilers at the time of the deadline.

C System Architecture and Design

The system will consist of two stand alone C++ programs, each will have an input file that will tune parameters and inputs. Each system will have a shell script that will verify the inputs, recompile on the current system, tune the parameters and then run the system with these settings. This will be the same for both the Enigma and BOMBE. The settings file will be in the form of a basic .txt file, this will have a standard format. The settings file will contain; which rotors to use in which positions, what their offset is, the plugboard and reflector mappings, as well as the plaintext.

Neither the Enigma or the BOMBE will allow for continuous input of characters as the plaintext or ciphertext respectively as this would be the dictate of the speed of the system, the characters cannot be input fast enough for there to be no machine idle time between two adjacent characters. Thus rendering a timed run completely dependant on how fast the user can type and not on the system itself.

The Enigma simulation will output its results to a text file as this will be useful for comparison as well as using the output of the Enigma simulation as the input for the BOMBE simulation as well as having files for both the cipher-text and plain-text that will form the basis of the testing of both simulations. The BOMBE will also output a text file as a result as, again, this will be useful for testing at a later date.

The parallel BOMBE will be written using the intel compilers, all of the parallelisation will be done through intel tools. The Intel standard pragmas, instructions to parallel, are very capable of the requirements that this projects has. The idea behind the parallelisation is to split the search space into parts and have each part run on a different BOMBE representation thus allowing a multicore machine to execute the search for stops over all of it's cores.

D Functional Requirements

Unique ID	Deliverable	Description	Priority
DL1	Simulate the Enigma Machine	Have a working Enigma machine that can take in settings and plaintext and return ciphertext.	High
DL2	Simulate the Bombe	A working BOMBE that takes in settings and ciphertext and returns decoded plaintext.	High
DL3	Show the correctness of the Enigma Machine	Show through a series of tests and validations that the plaintext fed to the Enigma Machine and the Ciphertext returned are paired through the settings of the enigma	Medium
DL4	Show the correctness of the BOMBE	Show that the known plaintext of the ciphertext input into the BOMBE is exactly the same as the plaintext that is returned by the BOMBE	Medium

DL5	Evaluate average computation time over set inputs for both Enigma and BOMBE	Using a wide range of known plaintext - ciphertext pairs find the average computation time, the time take on average to do a single encode or decode for both simulations	High
DL6	Parallelise The BOMBE and evaluate average computation time	Using known parallelisation techniques increase the speed at which the BOMBE will decode a given ciphertext	High

E Evaluation

Each system will be evaluated individually first then in conjunction to one another. To evaluate the Enigma, it will be tested over known (plaintext,ciphertext) pairs to show output is valid. This database will be taken from a valid third party simulator. The BOMBE is tested over known and unknown ciphertext, the output of the known ciphertext will then be reencrypted using the settings that were used to decrypt it and they will be compared to show correctness. Once correctness has been established then decryption time will compared between the parallelised and standard BOMBE over the same ciphertexts, and an average improvement will be shown.

F References

References

Enigma – Britannica Academic (n.d.).
URL: *http://academic.eb.com.ezphost.dur.ac.uk/levels/collegiate/article/32677*

Navy M3/M4 Enigma Machine Emulator (n.d.).
URL: *http://enigma.louisedade.co.uk/enigma.html*

Stoler, M. S. (2007), 'Re-engineering the Enigma Cipher'.

The Turing Bombe - Cribs and Menus (n.d.).
URL: *http://www.ellsbury.com/%5C/bombe1.htm*