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 magnitude of 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

What the project is about - talk through the motivations and a rough outline of the project
The project aims to show that a significant speed up has occurred in the field of computation since the inception of this cryptographic technique. Context of the project - what are some real world things that need to be taken into account
What was achieved - go through deliverables and then go through issues and counter
2-3 pages

Since the time of Alan Turing, the developments of computational power cannot be understated. We have developed from electromechanical devices that have a single function, to elec-

tronic devices that are capable of renders and simulations of electromechanical devices within themselves, the BOMBE is an example of an electromechanical device and this can be rendered and simulated within a modern mobile device, let alone the cutting edge, which at the time, the BOMBE was. Within more cutting edge technology we are able to study models of the universe, and other academic pursuits that were inconceivable at the time of Alan Turing.
Alan Turing is the man that is credited with the creation of the BOMBE, this electromechanical device is then credited with breaking the German Enigma codes that are encrypted using the Enigma machine, this is another electromechanical device with a single function, it is used to encrypt messages that are then sent to another enigma machine that will then encrypt the message. This is how almost all of the German orders were relayed. The breaking of this code is said to have shortened the second world war by several years and thus save innumerable lives.
The motivations of this project are two fold, the first is to show the effectiveness of both the Enigma machine and the BOMBE. This will be done via the implementation of both in a modern setting. The second is to outline the developments that have occurred since the inception of the Enigma and BOMBE, by showing the significant increase in computational efficiency, an analogue for this would be how fast the BOMBE breaks the Enigma code. The most simple solution will be a BOMBE representation using modern coding techniques without any parallel improvement or any other modern techniques. This will be purely a copy of the enigma, in C++ code. This will outline the improvements between an electromechanical device and the sequential electronic devices that have been common from around the 1980s. The second, more complex solution will incorporate parallel programming, this was introduced in the early 21st century, and is still an improving technology.

II RELATED WORK

Survey relevant literature
Relate to your own project
May be able to reuse parts of the literature survey
use anything relating to parallel computing, the introduction of common single core computing, the introduction of C++ code, the development of the intel tool set, any historical papers about the enigma code, anything about the BOMBE, anything about the measurement of computational improvements.
2 pages

III SOLUTION

Overview of architecture and design - use parts from the design report
Description of tools used - design report
Outline of algorithms to be used - design report
Features of the implementation process - issues and struggles with the implementation, how they were overcome
Testing - validation and all other testing done
Verification and validation
Stages of the life cycle undertaken - software engineering
4-7 pages

A Enigma

Paragraph talking about why C++ was chosen as the language to implement the Enigma in.

The electromechanical device known as the Enigma machine is best known for its use by the Third Riche, it was designed by a company called XXXX in XXXX, and then sold to the German armed forces. The machine is designed to allow secure cryptographic messages between two of these machines, the first will encode the message using decided settings, the second will, using the same setting, decode the message. This is done through a set of alphabet mapping, a one to one relationship between one alphabet and another, one letter is mapped to another and this relationship is bidirectional. No two letters from one alphabet map to the same letter in another. The Enigma machine is made up of 7 parts, there are only 5 important parts plus an additional component and some wiring to make it work. The important parts are the plug-board, the three rotors, and the reflector. The other parts are a converter that allows all the separate components to work together, along with a lamp-board to show the encrypted letter, along with wiring to connect them all together.
The plug-board is a mapping from one alphabet to another that is a manual process of re-plugging 26 wires that map from one to the other. This does not change for each letter that is input into the device, unlike some of the components we will see later. The reflector is much the same as the plug-board, except that it cannot map a letter to itself. It is a simple one to one mapping that does not change over time, in the earlier designs of the Enigma machine, it could not be rewired, and instead had to be swapped out for another reflector with different wiring, in a similar way to that of the rotors that we discus later on, with later designs came the ability to rewire it. The rotors are the core of the machine. These are three modular, circular, mappings that are altered each time a letter is encrypted by the machine. The way they work is that the inner ring and outer ring each have contacts on them, when the outer ring is rotated then the mapping will change. there are 26 different mappings that can be used. The three rotors are chosen from a set of available rotors and each one has a unique ordering of letters on both its rings. Thus the order in which the rotors are input into the machine is important.
The enigma machine will take the first letter of the message that is to be encrypted from a key-board and first pass it through the plug-board, it will then go to a converter which allows the signal to be passed to the first rotor, another mapping, then the second, and the third, in each of these case the inner ring of the previous rotor is connected to the outer ring of the next rotor. Then the reflector is reached, the mapped letter is then passed to the inner ring of the third rotor and this is a reverse mapping. The outer ring of the third rotor is connected to the inner ring of the second rotor and thus the signal will pass back through the rotors in this fashion. Once the plug-board is reached, the reverse mapping is done and a light will trigger on the lamp-board showing the letter that the input letter has been mapped to. After this the rotors are then stepped, this means that the first rotor will rotate one position for each letter that is encrypted. The second rotor will only step when specific notches are reached on the first rotor, the third is the same but for notches on the second rotor. Based on this method the first rotor will rotate more than the second and the second will rotate more than the third.

The recreation of the enigma machine was to be done in C++, this was chosen as it has sig-

nificant parallel tools available for it, in the form of the Intel c++ compiler. A tool that has shown great capabilities in parallelism. This would be very helpful in the creation of the advanced BOMBE recreation and developing in two separate languages didn't make sense.
The recreation revolves around a settings file, this is a text file that contains all the setting that will be used by the enigma machine to encode the message that it is given through the command line interface, these settings include; the plug-board mapping, the reflector mapping, the rotors that will be used and their corresponding initial displacement. This file is then updated automatically to reflect the changes that have occurred to the settings as the message is encrypted, each letter will have a different set of setting as the displacement of the rotors changes, this will be reflected in the settings file. The c++ file that makes up the core of the code that has realised enigma, is called the main file, this will manage all the changes that occur as well as managing user interaction with the device and having a central location with which to manage the other c++ files that realise different parts of the enigma machine.
The plug-board, reflector, rotors, as well as a converter file have all been created in their own separate files so as to improve the modularity of the system, as well as provide a more focused development route; each file can be developed without removing functionality for the others.
The main file contains the instructions on how to use the system, it contains all file dependency instructions, as well as containing the set up function and main function. The set up function will be run before the system is usable, it will find the set up file, using the information that it contains, it will then make the system usable through the main function of the main file. As the set up function is run, it will call the set up functions of all the other parts of the enigma in other files, the plug-board file will store its mapping, the reflector will store its mapping and the rotors file will store the rotors that are currently in use and make sure that the displacement is correct for each one. Once this has been done the main function is used, this will take in user input through the command line, the only thing that will be input is the plaintext, this is the message to be encrypted. The main file will then iterate through each of the letter in the message and encrypt them individually, using unique settings for each letter. The first letter will use the settings that have been read in by the set up file. the first stage is to convert the letter to its corresponding number, for example A is equal to 1 and Z is equal to 26, this makes use of the converter file, this file contains functions to convert between letters and numbers as well as another function that does the reverse. This was done because numbers are much more manageable than letters in terms of computation. once this has been done the number will be sent to the plug-board. The plug-board stores the mapping from one set of 26 numbers to another set of 26 numbers, this is a computational realisation of the alphabet to alphabet mapping that is done by the electromechanical enigma device. Once this has been done in the require direction, the plug-board has two directions mapping forwards and mapping back, we will use the mapping forward function now, and the mapping back function at the end to receive the encrypted message. We have now finished the mapping through the plug-board and will move onto the first rotor. The first rotor will have been chosen by the user in the set up file as well as its initial displacement from its standard position. This is another alphabet to alphabet mapping, but with a twist. The twist is that each letter does not have the same mapping, once a letter has been encrypted the enigma machine will increment the displacement of the rotor and once the first rotor has reached a specific displacement it will increment the displacement of the second rotor which in turn may reach a specified displacement and cause the third rotor to increment its displacement. This is only done once the current letter has been fully encrypted, meaning once it has passed forwards through

every rotor as well as back through them. Once the letter has been mapped by each of the rotors it reaches the reflector, this is very similar to the plug-board, it can be re-programmed in a very similar way. This mapping is then done, with the caveat that a letter cannot be encrypted as itself at this point, and the return journey begins. The letter will travel back through each of the rotors, this time being mapped in the opposite direction, then to the plug-board, which will also map in the backwards direction as mentioned before. This concludes the encryption of this letter, in the electromechanical enigma this would cause a letter to light up on the device to show which letter the input letter has been encrypted as, in this device, the covert file will be used, this is because throughout this process the 'letter' has actually been the number corresponding to that letter so as to simplify the process, we will map from number to letter and output this encrypted letter. Once we have a fully encrypted letter the rotors will change their displacement as the process will start again for the next letter in the message. This will continue for all letters in the message until we reach the end, at this point the new setting that the system is using will be output to the settings file, this is done so that we are not reusing the same setting each time we encrypt something as the set up function will be reading in the same values again and again from an unchanged settings file.

Paragraph talking about the testing that was done, go through the exhaustive search that was done along with the proof that all components work as expected individually.

Briefly go through the software engineering development that was done.

B BOMBE

Paragraph talking about why C++ was chosen to implement the BOMBE and another talking about the benefits of the intel tools.

Paragraph talking through how the electromechanical version of the BOMBE worked, and any history that is relevant to the paragraph. Another paragraph about any extra details that need to be known such as the known pair attacks

The first version of the BOMBE that was recreated in C++ code never actually existed, this was made as a benchmark for the others. In this version we will use a brute force approach with no limitations, that is we will try every combination of setting possible to attempted to map from the input encoded message and the guessed message that it corresponds to. This is done through a similar method to that which was used to make the Enigma Machine. In fact a copy of the exact enigma machine code was used in this implementation. The only difference is the main file and settings file, the main file will rewrite the settings file with each possible setting and then it will be tested, if the results are wrong then the next setting is used, this is done until the settings that were used in this case are found, they will then be output by the main file so that they might be used by an enigma clone to break any other messages that were sent in the same block as the message that was just broken and thus have almost identical settings. The brute force approach is slow. This was known from the start, that is why it was done. This is the slowest that a BOMBE could be using modern techniques.

Paragraph about the exact re-creation of the BOMBE using all the eliminations possible to minimise the search space.

Paragraph talking about the parallelised version of the BOMBE and how it was designed, don't talk about speed up or any issues that were faced.

Paragraph talking about the testing and validation that was done for each model individually. Talk through the tests but not the results.

Briefly go through the software engineering development that was done.

IV RESULTS

Evaluation method description
Experimental settings - talk about the grid search done
Results generated by the software
2-3 pages

V EVALUATION

Discussion of strengths and weaknesses of solution and of lessons learnt - evaluate issues
Limitations of the solution - given more time... etc.
Critical appraisal of the way the project was organised - critique my own organization
2-3 pages

VI CONCLUSION

An overview of the project
Brief description of the main findings
Discussion on how the project can be extended
1-2 pages

A 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*