Enigma: Encryption & Decryption Methods

Final report

Submitted for the BSc in

Computer Science

April 2020

by

James Duncan

Word count: 15,781

Abstract

This project was undertaken to enhance the developer's knowledge of cryptography and to demonstrably attempt to remedy the core flaw in the WW2 Enigma machine that directly led to its breaking by the Turing Bombe.

The principal objective was to explore methods of improving the Enigma using modern computing, specifically to establish whether it is possible to secure the Enigma against the Bombe. A secondary objective was pursued to undertake an objective comparison against modern encryption.

The scope of the project was to resolve the inherent self-encipherment security flaw to ensure protection from the Bombe, and undertake a comparison of the capabilities of the Enigma to modern day methods, focussing on Symmetrical Encryption.

Options to improve the Enigma are discussed and a binary switchboard was chosen and added to the Enigma. Due to later conclusion, an Enigma block cipher had the potential to be a more reliably successful solution.

A set of encryption methods was developed specifically for this project, the Enigma machine, improved Enigma, AES, SHA-256 and MD5.

JavaScript was the chosen development language whilst management was governed by the use of a considered selection of tools such as GitHub and Trello.

It was concluded that it is possible to fix Enigma's self-encipherment, however, due to the solution chosen, fixing the issue by the chosen method caused significant data integrity issues.

Extensive research into areas encompassed by this project was undertaken into symmetric encryption of the Enigma and modern methods including the DES family and AES, whilst also including key areas of modern cryptography such as hashing and Asymmetric Encryption.

Acknowledgements

I would like to thank Dr Alexander Turner who supervised my project throughout its development and gave advice when required; I wish him all the best at his next University. I would also like to thank Dr John Atanbori for picking up project supervision after Alexander Turner left the University, he has been of great value during the write up of this final report. Lastly, I would like to thank my family for being incredibly supportive of my work throughout the year.

Contents

1 Introduction

Cyber Security is an ever-expanding field that has grown from the advent of the technological world. The act of keeping data safe, although much more relevant today it has been around for thousands of years. The Enigma machine is one of the most famous encryption devices in history; its development, use and breaking has helped improve modern day encryption. As the need to protect user data increases the amount of data breaches also increases, this is most evident when looking at recent breaches - for example, Facebook recently made an "admission to storing user passwords in plain text" *(Ranger, 2019)*.

In this report the project will take a step back and look at the Enigma machine in detail, comparing it to modern encryption methods whilst highlighting key advancements and weaknesses. There is specific focus on symmetric encryption algorithms which use a singular key as the main comparison point - although some references will also be made to asymmetrical encryption. This project will also investigate different ways of improving the Enigma machine in an attempt to make the machine more secure and put it into a state that cannot be broken by the Turing Bombe machine.

Below you will find the main research question and a list of objectives, which sets out the main areas covered in the literature review. Further sections of this report go into the developmental process of the Enigma Machine, Modern Encryption techniques (AES, Hashing) and self-developed enhancements to the Enigma (The Switchboard). These developments are compared and contrasted to each other whilst evaluating the security added or lost by attempting to improve the Enigma.

1.1 Objectives

1.1.1 Objective 1 – Enigma (Mk1 & Mk3 Kreigsmarine)

Before any analysis of the Enigma can be taken, an Enigma machine must be built. The machine must function exactly as the real device used in World War 2 (WW2) including all rotors, plugboard and ring settings. For this reason, two Enigma machines will be built, the first is the Mk1 Enigma machine used throughout the war and the second is the German Naval Enigma (Kreigsmarine) with the additional fourth dial adding a multitude of additional combinations. These machines should allow the user to input all the settings for the machine and type a message to another who can then decrypt the message on the other side.

1.1.2 Objective 2 – Cribbing & Improved Enigma

Once the Enigma machines have been built analysis can start based on the Enigmas' key weaknesses, for this reason a cribbing device was developed to check for a key word in a message (plaintext attack). Based on the analysis an improved Enigma machine that aims to fix the major flaws of the original was developed to compare against the original Enigma and modern-day encryption methods. The improved machine was based on the original Enigma with nearly identical functions whilst also making use of modern-day computer speeds and capability.

1.1.3 Objective 3 – Modern Encryption & Comparisons

Before any direct comparisons can be drawn between the Enigma machines, some modern-day encryption methods were developed including MD5 and SHA-256 Hashing (one-way encoding) and AES Encryption (Modern standard). Further analysis of all methods can then be collated and attacked in various ways to test the strength of the encipherments.

1.1.4 Additional Objectives (Morse code)

As an additional objective a Morse code translator was added to the development as this is historically how messages were transmitted across large spaces in WW2 and before the Internet.

1.2 Research Question

"Does making improvements to serious security flaws in the Enigma machine using modern computers ensure security against the bombe machine, and how does this Enigma stand up to Modern Encryption methods? "

1.3 Report Organisation

The report covers a series of discussions and investigations that lead to the development of the Enigma, improved Enigma, AES and Hashing Algorithms.

The background section (Section 2) discusses the basis to the project specifically looking at the functions of the Enigma including all of its breaks such as the Bombe machine. However, this section also covers both the DES family and AES, explaining how these symmetric ciphers work and why DES replaced AES. Also covered are Hash functions (specifically SHA-256) and attacks on their functions such as collision and rainbow table attacks.

Section 3 explores the requirements of the software developed for this project and key decisions made from Section 2 research, this is required to determine what form the improved Enigma needs to take.

Section 4 explores software design in depth, breaking down key areas required for the development of the full solution.

Section 5 explains the development process taken when developing the Enigma, improved Enigma, AES, SHA-256 and MD5 Hashing. This includes how the project was managed and key areas of code that need in depth explanation.

Sections 6 & 7 both explore how the developed software proves or disproves the research question before conclusion.

2 Background (Literature Review)

This Section aims to brief anyone unfamiliar with the Enigma and Modern encryption to provide context to the full project and its scope. It starts by introducing the Enigma machine before exploring the key concepts used to develop an Enigma, including the method of encryption, the strength and operation during WW2. The exploration of the Enigma aims to highlight it's key weakness for further development towards the research question. This section also covers the use of modern encryption and its vulnerabilities, specifically covered is AES, Hashing methods (SHA-256 & MD5), DES and Asymmetrical encryption.

2.1 Enigma Mk1 & Enigma M3 Kreigsmarine

Enigma Mk1 (Taken by James Duncan 19th December 2019, Bletchley Park)

2.1.1 Context

After the German defeat in World War 1, the British revealed that they had broken the ADFGVX Cipher that the Germans had believed unbreakable during and even after the war. Breaking this cipher gave the allies a distinct advantage as they were able to understand secret messages being sent between enemies that believed these messages secure. Understandably, after this knowledge became public the Germans' sought a new method of encryption and they selected the Enigma machine.

There was nothing secret about the basic Enigma – "it had been exhibited in 1923 soon after its invention" *(HODGES, 2019)* by Arthur Scherbius, who sold them commercially to banks and other companies that wanted the best security to ensure customer data was not read by an unintended audience. In 1935 "the British had created their own version by adding extra attachments to it, this machine was known as the Typex machine" *(HODGES, 2019).*

On adoption, the Germans made significant improvements to the device before deployment for military use. The Scrambler (or 'Rotors') had different wiring than the commercial Enigmas so that a commercial entity could never decode a military variant's message, unless it had also modified its internal wiring. "By 1925 Scherbius began mass production of military Enigmas that went into service the following year" *(Ward and Singh, 2001).*

The German military Enigma went through multiple additions that only added to its security before and during the Second World War (WW2). These changes included the addition of rotors, reflectors and creating a four-rotor version of the Enigma. However, the one change that was the most significant was the addition of the plugboard that added a higher number of possible keys (Starting Positions). The Enigma that didn't have a plugboard was known as the Unsteckered Enigma (Korpal, no date).

Attempts to crack the German Enigma were limited at first with few countries being notably interested in cracking the device, however, Poland, fearful of Germany (later invaded by Germany starting WW2) came up with many ways of cracking the early code. German additions to the machine strained the intelligence resources Poland could put into the project and as a result they handed the project to Britain.

Britain, fully appreciating how much of a difference it would make to the war, focussed cracking efforts on using a machine of their own inspired by the Polish codebreakers, Bomba. It is said that the British achievements in cracking the Enigma machine shortened WW2 by up to two years *(Callahan, 2013)*.

2.1.2 Enigma Method

The German Enigma is actually an incredibly simple device that provided the portability and speed required to send an encrypted message from anywhere during war time. It was simple enough to act as a black box method for the soldiers at war; soldiers did not need to know the inner workings of the Enigma to send or receive messages.

The device sent an electrical signal through the desired connected plug and then the rotors scrambled the letters into different positions. The current then went through a reflector which sent the current back through the rotors and lit up the encrypted letter to complete the circuit *(Ward and Singh, 2001)*.

The rotors are the main component of encryption with each rotor having different properties depending on how the user sets the rotor up or how they were wired. A simplified version of the rotors can be found below in Figure 1.

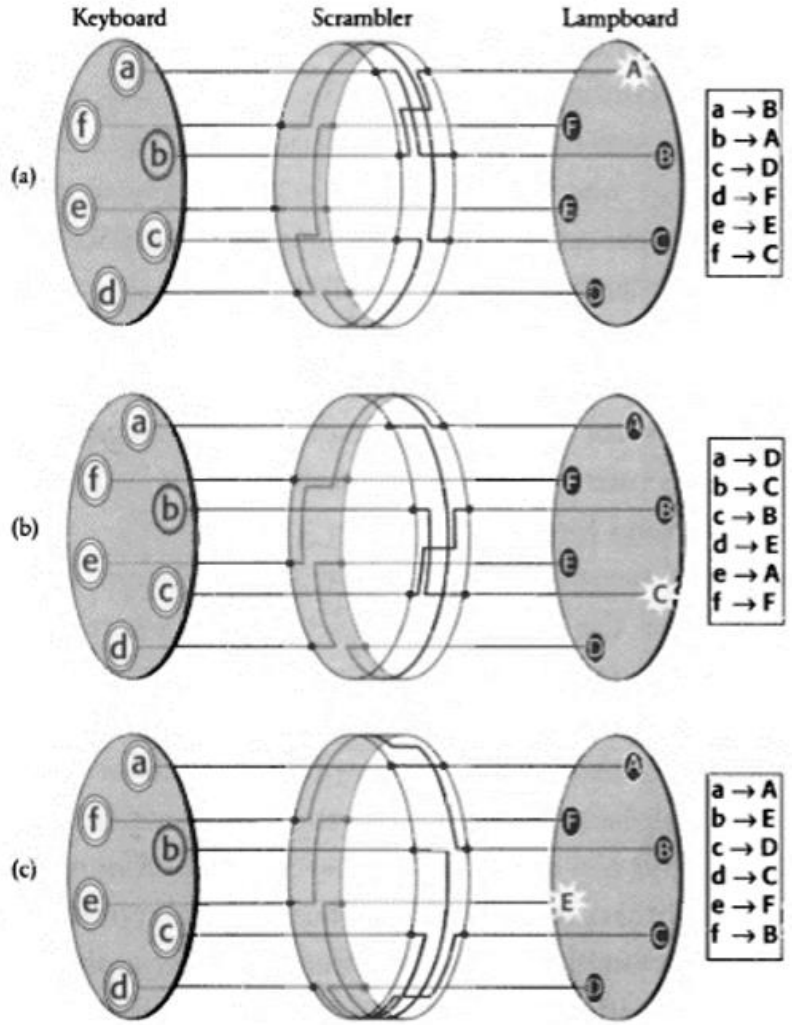


Figure 1: Example Rotor Wiring (Ward and Singh, 2001)

Each rotor has 26 starting positions (representing each letter of the alphabet) which are connected via a wire to another letter. When a letter is keyed in on the keyboard say "B" it is changed to the letter where the wire is connected say "A". In essence this letter has been transformed to a different one. After a key has been pressed the rotor wiring will change meaning, and if the same letter is pressed again it will not encrypt as the same letter as before. This will continue until the rotor made a full rotation.

The issue with this is that if a single rotor is used the strength is limited and patterns could emerge after the rotor completes a full rotation (with only 26 different positions for each ciphertext character). "Cryptographers are keen to avoid repetition because it leads to regularity and structure in the ciphertext - symptoms of a weak cipher" *(Ward and Singh, 2001)*. For this reason, multiple rotors were used, the output of a rotor goes into each subsequent rotor. This created a much more respectable rotor strength of 26*26*26 = 17576 *(HODGES, 2019)*.

Each rotor is wired differently depending on the rotor used. The Enigma 1 used five different rotors and the Enigma Kreigsmarine had a roster of eight which could be positioned in any order. Each rotor had a different point where the rotor would push the next rotor to the next position (the notch position) and the Kreigsmarine variant had some rotors which even double stepped. Reflectors could

also be switched out although they could not move in early models of the machine. Further rotors were introduced later on increasing the amount of selection the Germans had. In turn, the selection of rotors increased the key strength of the machine. Details of these wirings can be found in Figures 2.

Wheel	ABCDEFGHIJKLMNOPQRSTUVWXYZ	Notch	Turnover	#
ETW	ABCDEFGHIJKLMNOPQRSTUVWXYZ			
I	EKMFLGDQVZNTOWYHXUSPAIBRCJ	Y	Q	1
II	AJDKSIRUXBLHWTMCQGZNPYFVOE	M	E	1
III	BDFHJLCPRTXVZNYEIWGAKMUSQO	D	V	1
IV	ESOVPZJAYQUIRHXLNFTGKDCMWB	R	J	1
V	VZBRGITYUPSDNHLXAWMJQOFECK	H	Z	1
UKW-A	EJMZALYXVBWFCRQUONTSPIKHGD			
UKW-B	YRUHQSLDPXNGOKMIEBFZCWVJAT			
UKW-C	FVPJIAOYEDRZXWGCTKUQSBNMHL			

Wheel	ABCDEFGHIJKLMNOPQRSTUVWXYZ	Notch	Turnover	#
ETW	ABCDEFGHIJKLMNOPQRSTUVWXYZ			
I	EKMFLGDQVZNTOWYHXUSPAIBRCJ	Y	Q	1
II	AJDKSIRUXBLHWTMCQGZNPYFVOE	M	E	1
III	BDFHJLCPRTXVZNYEIWGAKMUSQO	D	V	1
IV	ESOVPZJAYQUIRHXLNFTGKDCMWB	R	J	1
V	VZBRGITYUPSDNHLXAWMJQOFECK	H	Z	1
VI	JPGVOUMFYQBENHZRDKASXLICTW	HU	ZM	2
VII	NZJHGRCXMYSWBOUFAIVLPEKQDT	HU	ZM	2
VIII	FKQHTLXOCBJSPDZRAMEWNIUYGV	HU	ZM	2
UKW-B	YRUHQSLDPXNGOKMIEBFZCWVJAT			
UKW-C	FVPJIAOYEDRZXWGCTKUQSBNMHL			

Figure(s) 2: Enigma Rotor Wiring (Crypto Museum, 2009)

Mark 1 Enigma (Top), Enigma Kreigsmarine (Bottom)

Due to the nature of these notch positions it made the rightmost rotor move very fast (each keypress), the middle move far less often and the final rotor (Slow Rotor) barely moved as it required the middle rotor to move to its notch position. These notch positions could also be changed to a different letter; this was known as the Ring Setting (Ellis, 2009) which increased the cryptographic strength and gave yet more options to customise the keys.

Once you had all of the rotors set up in the correct position, orientation and ring setting you could encipher a message. Thanks to the reflector being reciprocal you could also decrypt messages sent using that set up of the machine. “In essence the reflector made the encipherment and decipherment a mirror process”*(Ward and Singh, 2001).*

This is the point where Scherbius stopped adding to his Commercial Enigma, which was deemed secure but additional security was added to the military machine, the plugboard. The plugboard significantly increased the amount of possible settings that can be used to set the machine (see 2.1.3) yet it is a remarkably simple concept. Each Enigma fitted with a plugboard had a set of 6 cables (changed to 10 later in the war *(Ellis, 2009)*) these cables could connect one letter to another. This had the effect where if one cable was connected to say “A” and “B” any “A” typed would go through the plugboard and enter the rotors as “B” and vice versa.

This entire process can be given as a product of permutations. “Let S denote the plugboard transformation. L, M and R the transformations of the left middle and right rotors respectively. U the reflector transformation and P a simple rotation (A > B). The encryption of E, at rotor positions x, y and z can be represented as,

$$E = S(P^x R P^{-x})(P^y M P^{-y})(P^z L P^{-z}) U (P^z L^{-1} P^{-z})(P^y M^{-1} P^{-y})(P^x R^{-1} P^{-x}) S^{-1}$$

" (Gabbasov, 2015). You can find the full wiring diagram for the Enigma with plugboard below in Figure 3.

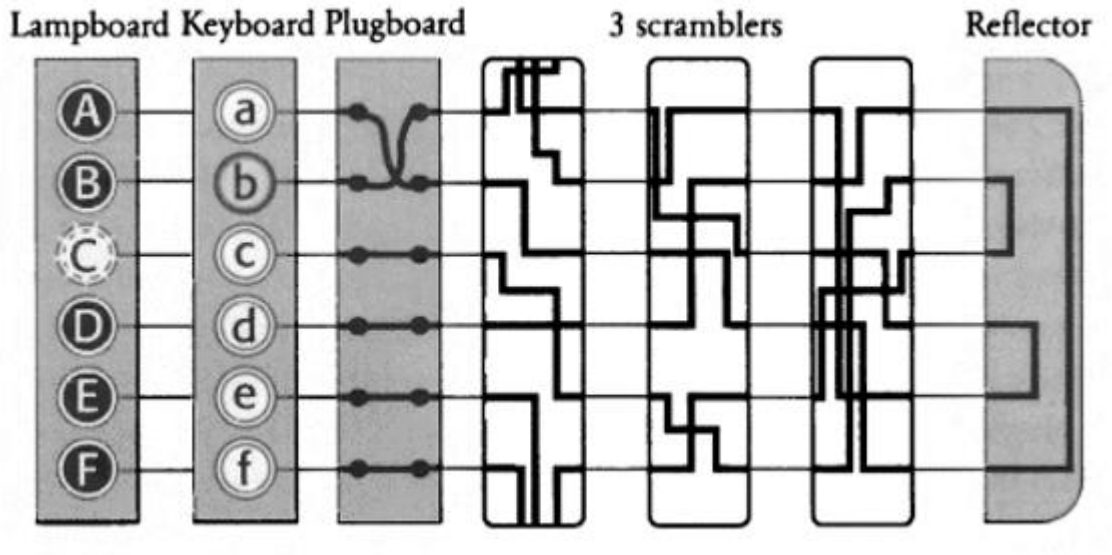


Figure 3: Enigma Transformation (Ward and Singh, 2001)
-Note: "scramblers" referred to above are equal to rotors

2.1.3 Enigma Strength

The Germans considered the Enigma to be unbreakable but just how strong was it really?

To determine this, it's best to split down the different components of the collective key for the Enigma. The key is made up of, the type of Machine (Rotor set), Rotor Orientation (which rotors are located in which slot of the Enigma), the ring settings, start positions and finally the plugboard *(Carter, 2008).*

As mentioned previously the plugboard gave a significant boost to the key space of the Enigma (how many possible keys can be used in the Enigma). For example, 1 cable can connect to 26 positions in the first input then 25 different positions to the end of the cable. This total key space generated by the plugboard can be calculated and is shown by the table in Figure 4.

c cables	total number of settings
0	1
1	325
2	44,850
3	3,453,450
4	164,038,875
5	5,019589,575
6	100,391,791,500
7	1,305,093,289,500
8	10,767,019,638, 275
9	53,835,098,191,875
10	150,738,274,937,250
11	205,552,193,096,250
12	102,776,096,548,125
13	7,905,853,580,625

Figure 4: Amount of Key space generated by the plugboard alone (Callahan, 2013)

Initially the Germans used six plugs which gave a plugboard key space (pairings) of 100,391,791,500 however increasing the number of plugs to ten increased this to 150,738,274,937,250. This could have been further increased to eleven plug cables however, as above, past that point adding more plugs actually decreases the key space.

The rotor order only added to this large key space making the machine much stronger. The amount of different positions is calculated by taking the maximum number of rotors used in the machine in the case of the Enigma mark 1 with three, and taking the factorial of that number which in this case is 6 *(Gabbasov, 2015)*. This was only further enhanced later when more rotors were added to the roster taking that number even higher. The different settings of each rotor can be calculated by the number of letters on the first rotor multiplied by the max number on the second rotor and then by that on the third rotor. In all cases the machine used a max number of 26 letters meaning the space generated by this is 17576 *(Hallenberg, 2015).* The possible ring settings are calculated much like the rotor setting and equals 676:

$$\begin{gathered}\#\text{possible keys} = (\#\text{possible plugboard settings}) \times (\#\text{possible rotor orders}) \\ \times (\#\text{possible rotor orientations}) \times (\#\text{possible ring settings})\end{gathered}$$

$$\Rightarrow \#\text{possible keys} = \left(\frac{26!}{(26-(2\times 6))!} \times \frac{1}{6! \times 2^6}\right) \times (3!) \times (26 \times 26 \times 26) \times (1 \times 26 \times 26)$$

$$\Rightarrow \#\text{possible keys} = (100391791500) \times (6) \times (17576) \times (676)$$

$$\Rightarrow \#\text{possible keys} = 7,156,755,732,750,624,000$$

Having broken down all elements in the key, we can calculate the full key space as denoted above for the Enigma mark 1 *(Korpal, no date)*.

As the total number is so high, you can appreciate why the Germans thought that this machine could not be broken as manual scrutiny of all these settings would take years. with the advent of modern computers this number is very easy to brute force through in a day, however, no such system existed at the time. The total number was only increased by increasing the number of possible rotor orders, additional rotors and the number of plugs to ten instead of six. Some networks were even stronger such as the Kreigsmarine's with 8 possible rotors and a reflector that can be inserted at any letter further increasing the key space by a factor of 26 *(Ward and Singh, 2001).*

2.1.4 Enigma Operation

As Scherbius wanted, his machine went into widespread military use. A protocol had to be established in order to ensure secure communication. Not changing the key would mean that thousands of people could break the code systematically or someone might get lucky. Key Distribution is an issue with Symmetric ciphers (the same key is used on both sides) as you want to ensure no one can intercept the key so that eavesdropper could not decrypt all the messages sent using that key. The Germans got around this issue by distributing a codebook that sets out a month of key settings for each day *(Gabbasov, 2015)*. These keys were global for the network and hence were a desired item for the allies during the war. Capturing one would mean complete access to the network without having to break the cipher which was achieved a few times during the war.

In day to day use, the Germans set up their machine in accordance with the daily setting and then sent a message using a message key. The message key was a set of three letters that related to the setting of the rotors, the idea was to set up the machine differently to the daily key so different networks could not decrypt data they were not meant to see. During the period of 1930 – 1938 the Germans encoded the message key twice. For example, the user (A) would set the Enigma to the daily setting using the encrypt message key to say "EIN" and twice the output of the message could be "XHTLOA". User (B) would then decrypt this using the daily key getting the result "EINEIN". Both would then set their Enigmas to this setting and continue the rest of the communications *(Gabbasov, 2015)*.

After 1938 this was changed as it introduced serious security flaws because some of the German users used the same code over multiple messages. "For example, if the first three letters read "HIT" then the next would likely read LERLER making the keyword "Hitler"" *(Callahan, 2013)*. This was exploited heavily by the Polish, so the Germans switched to just typing the message key once to establish a new key. Both design weaknesses and operator sloppiness were the two main weaknesses of the Enigma.

2.1.5 Enigma Early Breaks

The repetition of the message key was a serious security flaw, this resulted in the message key having relations to each other that could be mapped and searched through given enough messages. For example, if the encipher of EIN as the message key was "XHTLOA" you know that "X" in position 1 relates to "L" in position 4, "H" in position 2 relates to "O" in position 5 and "T" in position 3 relates to "A" in position 6. Sometimes, it would happen that the 1st,2nd, or 3rd letter would be the same to its counterpart in the 4th, 5th and 6th position - these positions were called "females".

Using the "female" method it was possible to whittle down the possible Enigma settings if you had enough "females" in the messages to find the daily key. From there you could decrypt all of the traffic for the day. This however, would take way too long to catalogue all of the core positions that resulted in a "female", so the Polish that were leading the efforts to break the Enigma created Zygalski sheets *(Turing, 2018)*. These were simply tables of all the core-positions, in which "instead of printing 'has a female' or 'has no female', there would either be a hole punched, or not" *(HODGES, 2019)*. These sheets were placed on top of each other until it suggested the correct rotor positions in use.

This method although much faster than manually testing the positions, still required a lot of resources, so the Polish thought that a machine might be able to do the work of checking these sheets in a brute force manner. By November 1938, Polish Cryptanalysts had built six of these machines (one for each rotor order) which were subsequently called Bomba's *(HODGES, 2019)*. Unfortunately, this method was short lived as in December the Germans added more rotors to the roster which increased the amount of possible orientations significantly. The Polish could not keep up with the resource drain so held a meeting with allied forces, including Britain, sharing their progress in breaking the machine.

2.1.6 Bombe Method

After the Polish briefing, British efforts to break the Enigma Machine accelerated and work began on making a Bomba that avoided the repetition of the message key. As the British correctly predicted, the Germans would change it after finding out it facilitated a security vulnerability into the system.

The new Bombe exploited the Enigma's key weakness - that no letter can be encoded as itself. It exploited cribbing (plaintext attack) where you can correctly match a piece of plaintext to a piece of ciphertext without decrypting the message. The Germans unwittingly helped the British find these cribs - for example, "experience showed that the Germans would send a regular enciphered weather report at 6AM each day" *(Ellis, 2009)*. Cribbing works by taking a ciphertext and moving the plaintext alongside it until all letters in the ciphertext are different to the plaintext. If a matching letter is found it's an invalid crib and if it is not found then it's a possible valid crib. An example of a cribbing process can be found in Figure 5.

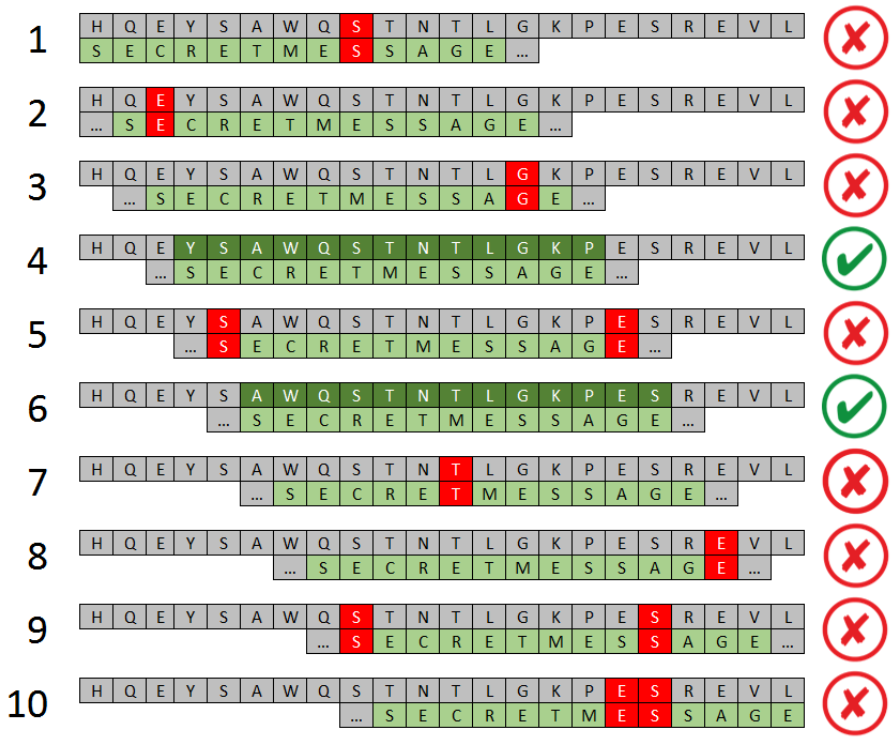


Figure 5: Cribbing Method (Computing, 2019)

It was the British mathematician Alan Turing that found a way to ruthlessly exploit this system to crack the Enigma machine. Turing created a loop design that showed how a crib could be turned into an electrical circuit. It worked by having a set of virtual Enigma machines set up with a ground setting that tries to transform the first letter of the plaintext into the ciphertext. The other virtual machines attempt to transform setting +1 and setting +3 into their relative ciphertext *(Ward and Singh, 2001)*. Assuming each virtual machine encrypts the letter in relation to the crib the machine will stop, and the settings can be checked in a testing machine. An example of this can be seen in Figure 6.

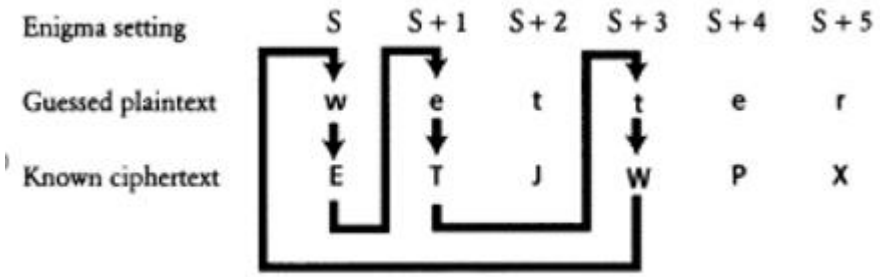


Figure 6: Turing Bombe Cribbing Method (Ward and Singh, 2001)

The Turing Bombe also had a way to nullify the plugboard, "The first Enigma has the electric current entering the scramblers and emerging at some unknown letter which we shall call L1. The current then flows through the plugboard, which transforms L1 into E. This letter E is connected via a wire to the letter E in the second Enigma, and then the current flows through the second plugboard it is transformed back into L1. In other words, the two plugboards cancel each other out. Similarly, the current emerging from the Scramblers in the second Enigma enters the plugboard at L2 before being transformed into T. This Letter T is connected via a wire to the letter t in the third Enigma, and as the current flows through the third plugboard it is transformed back into L2. In short, the plugboards cancel themselves out throughout the whole circuit, so Turing could ignore them completely" *(Ward and Singh, 2001)* . The full wiring diagram of the Turing Bombe can be found in Figure 7.

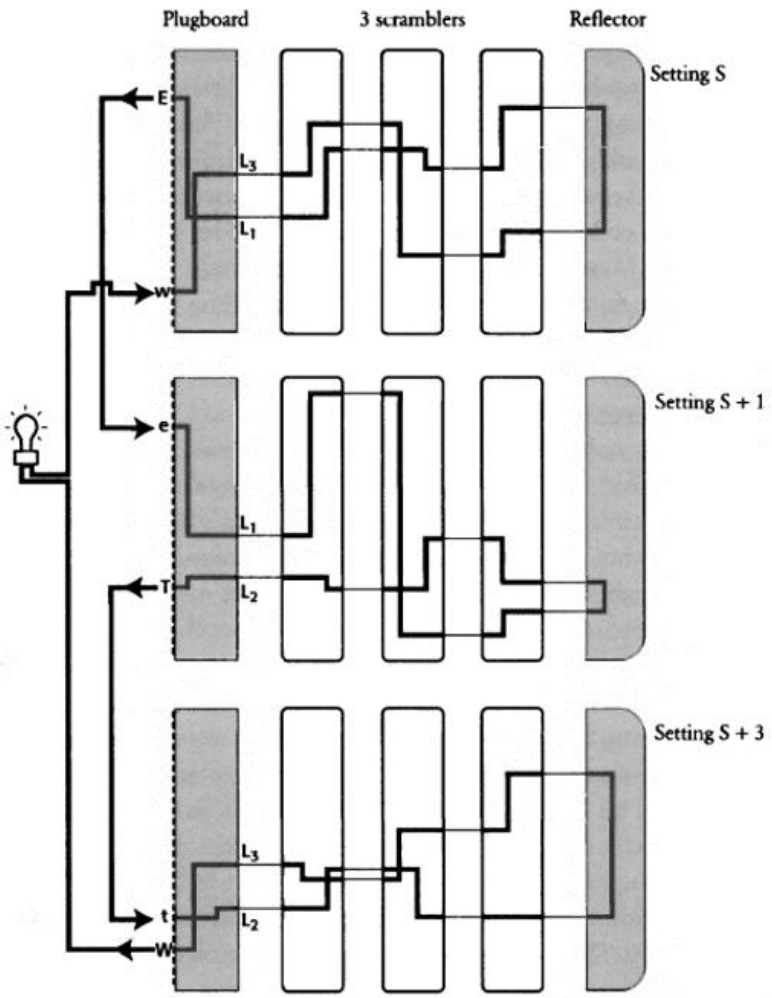


Figure 7: The Turing Bombe (Ward and Singh, 2001)

This machine allowed the British to break the Enigma Machine. It took additional effort to break the Lorenz Cipher (German Army) despite the same idea of using Cribs to break the code being implemented.

2.1.7 Morse Code

Morse code although not directly a part of Enigma was the prime way the Germans communicated their encoded messages over long distances. There is nothing secret about Morse code it is just an alternative way of representing the alphabet. The British set up listening stations (Y stations) all over their territory to tap into the German network *(Ellis, 2009)*.

2.2 Modern Encryption

2.2.1 Context

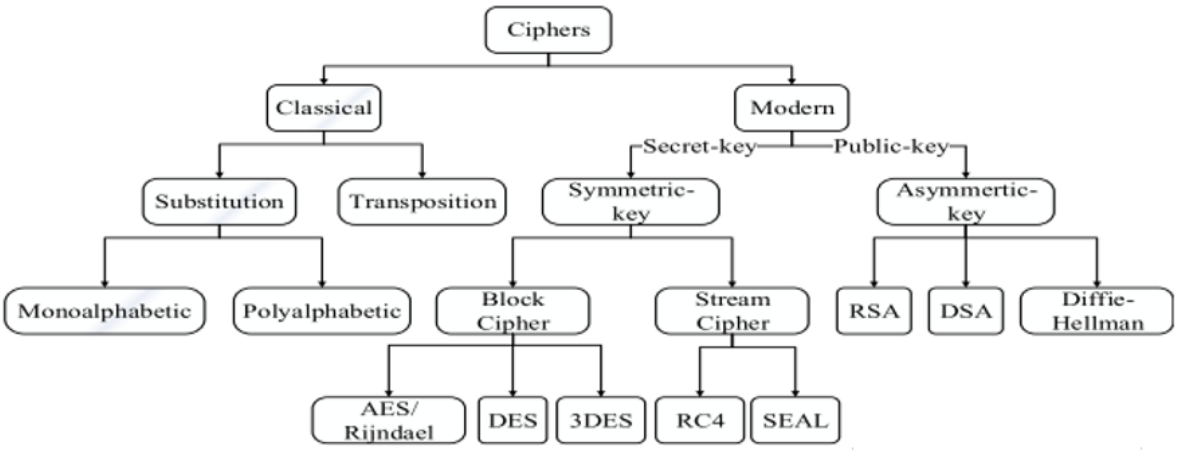


Figure 8: Types of Cipher (Singh and Supriya, 2013)

Up until now we have only been talking about Historical Symmetric Encryption. The Enigma itself is a polyalphabetic symmetric cipher that uses a singular key between encryption and decryption. However, since Enigma there have been major advancements to Encryption due to the advent of modern computers and the internet. In terms of Symmetric Key encryption that has been the focus of this report thus far Block Ciphers were introduced. These normally take in a set block size of data and encrypt that data through the algorithm chosen. This gave rise to DES (Data Encryption Standard) which was eventually surpassed by AES (Advanced Encryption Standard). There are also stream ciphers which often take the data and use the XOR Operator to merge the key and the data together on a bit by bit basis. Although Asymmetric key encryption (Public Key Encryption) is not a primary research point in this paper it is worth knowing how it works and you can find below a section on Asymmetric Encryption. A breakdown of ciphers can be found in Figure 8 which clearly shows the types of ciphers and the relative modern algorithms. Omitted from this list are Hash functions which, although not a method of encryption as they are one-way functions, are key players to many different ways of storing data securely online and providing data integrity. For example, password authentication uses Hash functions to create a hashed version of a password which is then used to verify the correct password; this allows for no database to store plaintext passwords.

2.2.2 DES, Double DES & Triple DES

DES (Data Encryption Standard) is a block Cipher. Block ciphers take a fixed set of plaintext (in a block of various sizes depending on the algorithm) and transforms it by performing a set of operations on the block often using the key for a set number of rounds. The exact procedure block ciphers use is down to the mode and algorithm used. “No Block Cipher is perfectly secure, it is always possible for someone to brute force to get a key, however, a good block cipher should make this computationally impossible” *(Bellare et al., 2015)*.

DES was developed in the 1970s by IBM and got official approval by NIST (The United States National Institute of Standards and Technology) in 1977 *(Morkel, no date)*. DES uses a key length of 64 bits however this was in actual fact lower as DES used some of this key as parity bits (the least significant bit), so the total key length was actually 56 bits in length. The Block size for DES is the full 64 bits.

DES utilised this key and block size with the structure of a Feistel Network (designed by Horst Feistel who was key in the development of DES) *(Bellare et al., 2015)* to complete the encryption. The

structure of this Feistel network can be complex to understand so provided in Figure 9 you can see how a Feistel network encrypts the block.

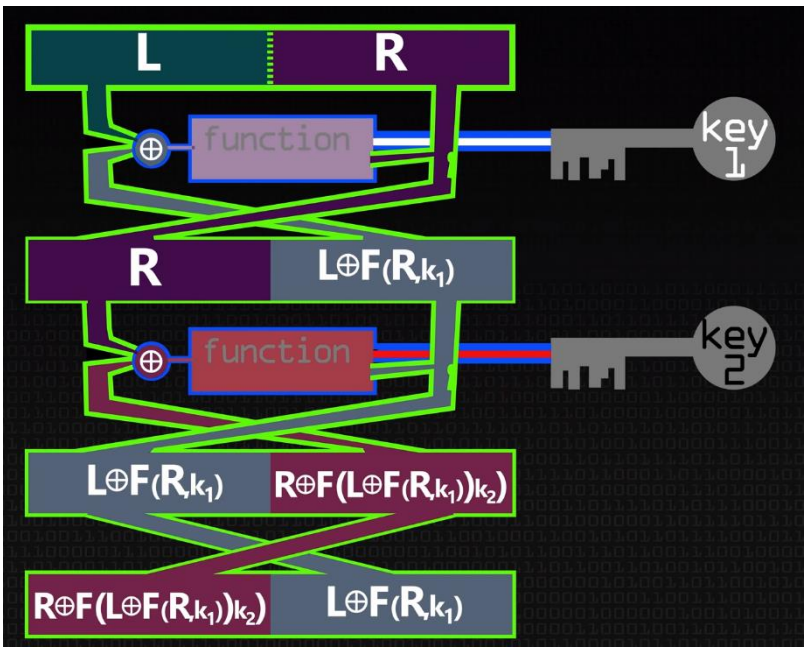


Figure 9: Feistel Network (Computerphile, 2012)

Figure 9 shows 2 rounds of Feistel encryption. First the key is expanded to create a set of round keys for each round in the cipher and these keys are used later in the network in the encryption function. The network first starts by splitting the block it is enciphering into two sides (Left and Right). The left side uses XOR with a function (which is different for each implementation of a Feistel network) and in this example the expanded key for the round is directly applied with the XOR function. The left output is then moved down to the right side of the cipher and the right side of the cipher to the left, this allows for the right side to be encrypted in the next round. In the second round the left-hand side data (in the example right) is brought through the same process that was used above and outputted much in the same way as the previous round. Finally, on the final round the sides are swapped and the ciphertext is returned. This network, no matter what the function (even if it's a one way hash function), will always be able to decrypt by applying the inverse to the network *(Computerphile, 2012).* It is this network that DES makes use of for encryption and the full DES Encryption diagram can be found in Figure 10.

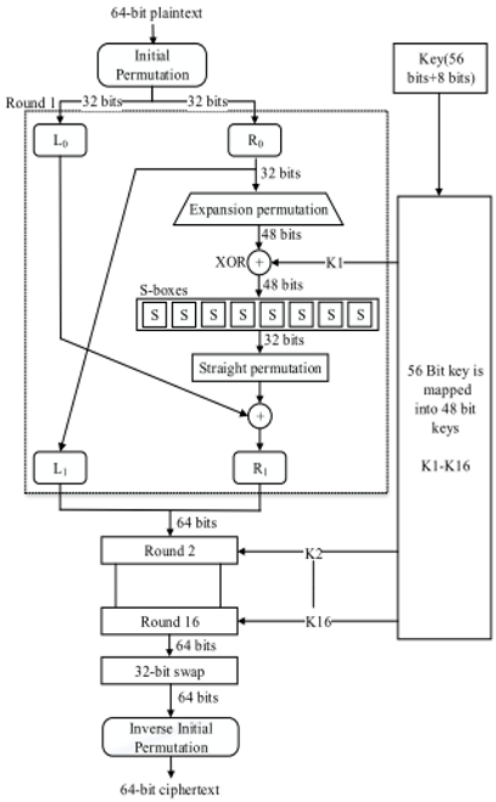
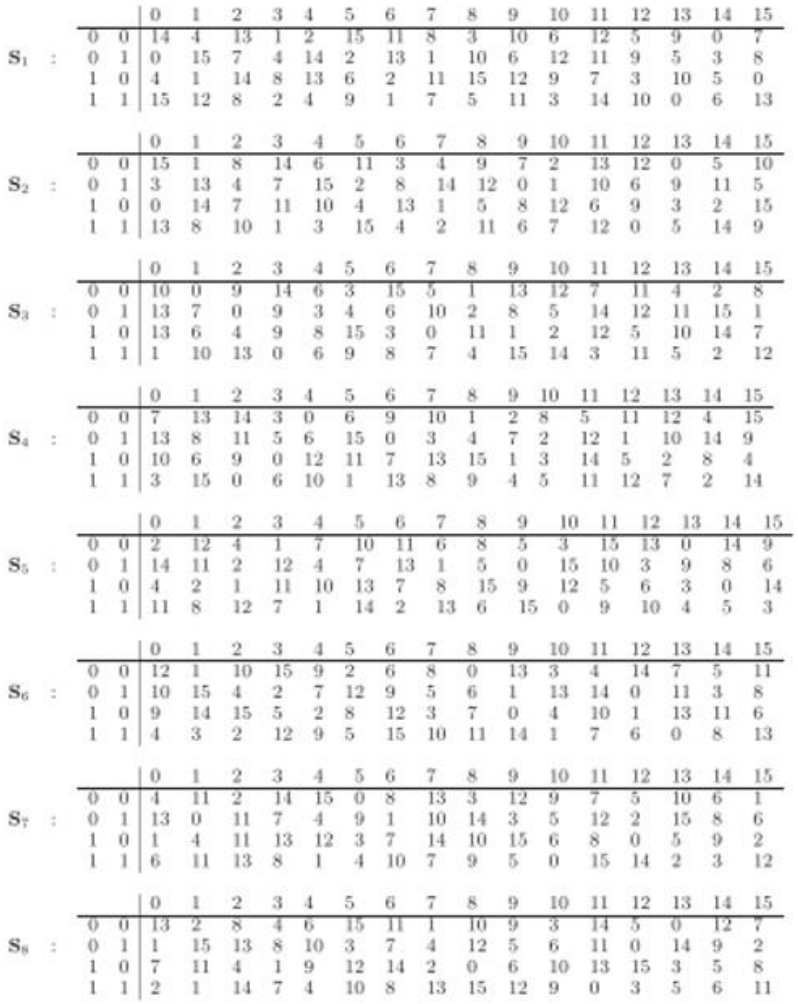


Figure 10: DES Encipher Map (Singh and Supriya, 2013)

The DES Encipherment shows clear inspiration from the Feistel Network cited, however, it has a series of key differences. The initial plain text is the IP (Initial permutation (The Block)) which is split into the left and right (32-bit sections), the section to be enciphered is then expanded to 48 bits and XOR is used with the round key. From here it differs from the Feistel network as the permutation goes through a set of Substitution Boxes (S-Boxes). The S-Boxes' job is to substitute each bit with a different bit, it does this by assigning each bit a reference that can be looked up in a table. Each S-box function takes 6 bits and returns 4 bits in turn compressing the full side back down to 32 bits *(Bellare et al., 2015)*. These S-boxes are fully invertible as the inverse table can be supplied. Figure 11 shows the S-box tables for DES. Finally, the output of these permutations is used with the XOR and the left-hand side creating the output for the first round. DES goes through 16 of these rounds *(Singh and Supriya, 2013)* before finally swapping the sides for the block of ciphertext.



S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1	0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1	1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1	0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1	1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1	0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1	0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1	1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1	0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1	1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1	0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1	1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1	1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1	0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1	1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figure 11: The DES S-Boxes (Bellare et al., 2015)

DES remained secure before it was broken in 1988 by the EFF (Electronic Frontier Foundation) who created a machine that brute forced 19 billion keys per second to try and guess the correct key *(Morkel, no date)*. This machine found the correct key in 4.5 days which makes the block cipher computationally weak and thus broken. Another brute force attack was recorded in 1999 where it was possible to test 250 billion keys per second which resulted in DES being broken in just a few hours (Morkel, no date).

Due to this breach in security for DES, it was necessary for an improved version of DES to be created. Many solutions were attempted such as double DES (2DES) which used two separate keys to encode the data one after each other. However, this was short lived as an attack called “meet in the middle” *(Bellare et al., 2015)* made this computationally possible. “Meet in the middle” attacked the data from both sides of the cipher to find the key in the middle halving the computational power required from one end of the cipher. As a result, Triple DES was developed.

Triple DES (3DES) has two modes that can be used; the first is a simple function that runs DES three times with three separate keys one after another (3DES3) and the second one is a mode that uses two keys (3DES2). In 3DES2 encryption is applied using key 1, the output of the previous step is decrypted using key 2. Finally, encryption of the output of step 2 is encrypted again using key 2. This is known as Encrypt-Decrypt-Encrypt (EDE) *(Aleisa, 2015)*. In both cases using more keys increases the 56-bit key length for each key applied, in turn this increased security.

2.2.3 AES

After DES was broken, there was a need for a new cipher that would remain futureproof for years to come. In January 1997 NIST announced a competition to develop a new encryption standard to replace DES and Triple DES *(Daemen and Rijmen, 2002)*. Many different teams entered the competition to become the Advanced Encryption Standard. This cipher would be endorsed by industry and government so had to be very secure. NIST wanted a cipher with the security of Triple DES but much more efficient and expandable. On 2nd October 2000 NIST announced the winner to be the Rjindael algorithm and it would be deployed without modifications *(Daemen and Rijmen, 2002).* AES/Rjindael is the primary Symmetrical encryption algorithm used today, it is a block cipher much like DES, but it does not follow the same Feistel network as DES but instead opts to use finite fields in a grid that it manipulates through multiple different mathematical functions.

AES's block Length is 128 bits however it supports keys lengths of 128, 192 or 256 bits. "It would be possible to define a higher block size or key length but currently there seems no need for it" *(Daemen and Rijmen, 2002).* The Algorithm is referred to as AES128, AES192 or AES256 depending on the key size used *(Singh and Supriya, 2013).* The key size also determines how many rounds are used in the system. A visual map of how AES encrypts a block is provided in Figure 12.

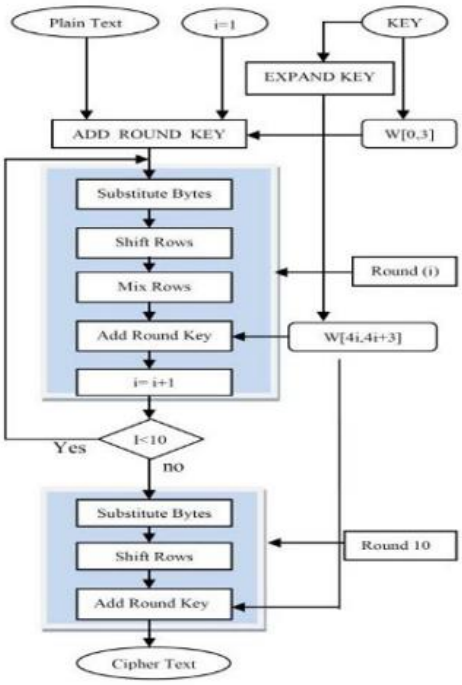


Figure 12: AES/Rjindael Network Map (Singh and Supriya, 2013)

AES first takes the key and expands it into the relative round keys depending on the size of the key. AES then takes the plain text and organises it as a matrix of the order 4x4 (of 8 bits/byte each) called the state *(Singh and Supriya, 2013)*, this state is then manipulated in the following ways. Before the first round, the round key is added using a bitwise XOR operation *(Daemen and Rijmen, 2002)* (Figure 13).The output of this state goes into the first round where it goes through an S-Box style method as seen in the DES algorithm (table for this can be found in Figure 14). This S-box method ensures no bit is flipped to the exact opposite of itself and ensures each bit is mapped to a different byte in the finite field *(Computerphile, 2019)*. The location of the swap is calculated on where the bit is located in the finite field (array) *(Bellare et al., 2015).* The outputs of the S-Box are then put into the shift rows step. This step is incredibly simple; the first row of the state stays the same, the second is shifted once to the left, the third twice to the left and the final 3 times to the left *(Daemen and*

Rijmen, 2002) (shown in Figure 15). The next step is Mix columns (Mix Rows in Figure 12) which takes each column and multiplies it with a set multiplication matrix (shown in Figure 16). Finally, the round key is added in another bit wise XOR. These functions are then repeated for a multiple number of rounds with the exclusion of the final round which doesn't mix the columns *(Daemen and Rijmen, 2002)*.

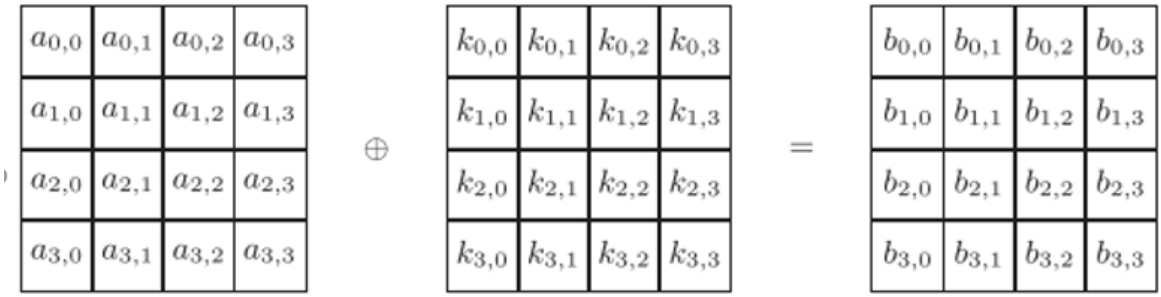


Figure 13: Bitwise XOR For AES/Rjindael (Daemen and Rijmen, 2002)

63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 14: AES S-box (Bellare et al., 2015)

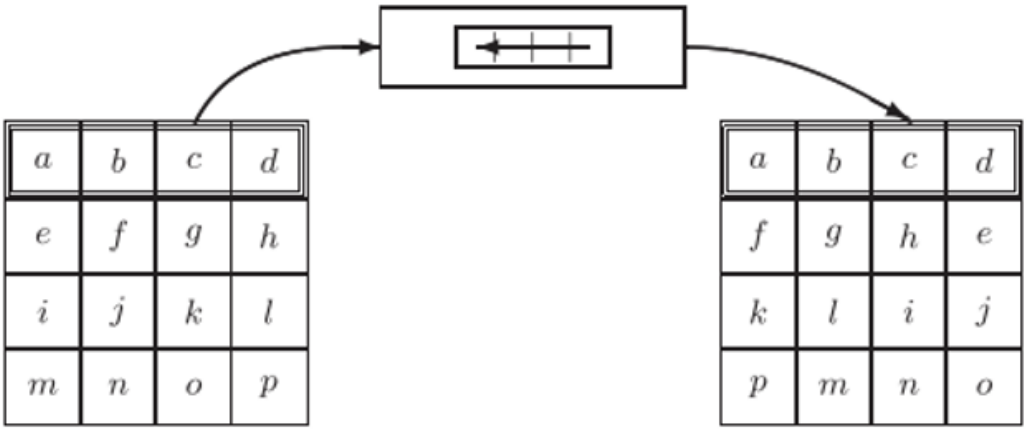


Figure 15: AES Shift Rows Operation (Daemen and Rijmen, 2002)

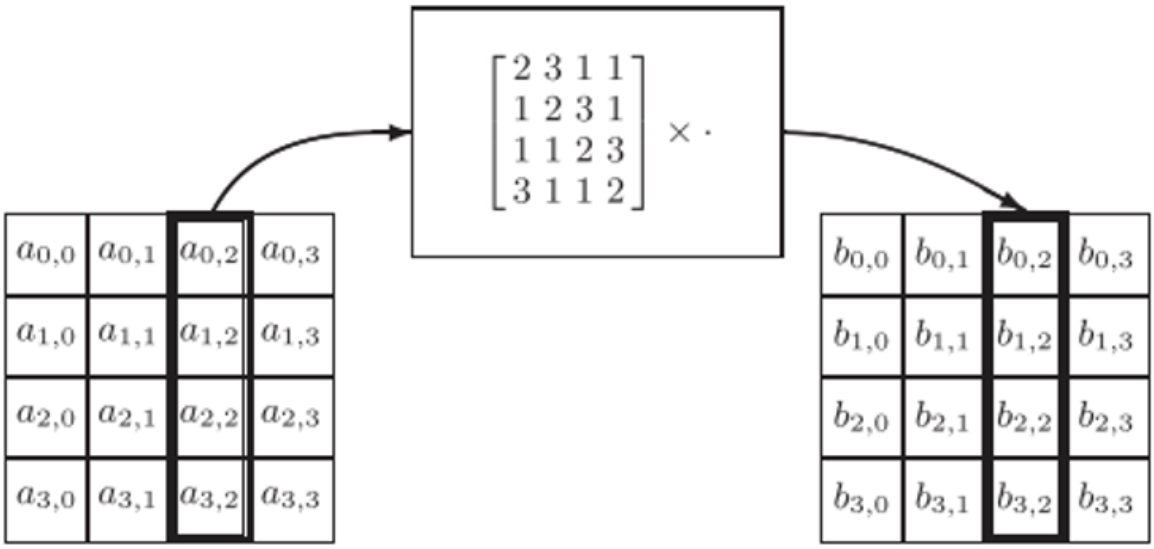


Figure 16: AES Mix Columns (Daemen and Rijmen, 2002)

Rjindael is a remarkably simple algorithm that results in very efficient encipherment of the block and allowed small processors to make use of AES and how it functions. So just how secure is AES? Well the best attack for it on a quantum computer halves the computational power required leaving potentially the 128 bit key weak, however, this is still computationally impossible on higher key lengths *(Computerphile, 2019).* Additionally, "the time required to check all possible keys at 50 billion keys per second in AES for a 128 bit key length is over 5000 years where as Triple DES with a 56 bit key would take 400 days" *(Aleisa, 2015)*

2.2.4 Asymmetric encryption

Asymmetric encryption also known as Public Key Encryption uses two keys instead of one to encrypt the data. This form of encryption is primarily used to ensure a secure connection between two individuals on the internet without a malicious interception of the message. "The major advantage of Asymmetric is key exchange" *(Meyers and Jernigan, 2018)* , you don't need to swap keys in asymmetric encryption, however, it is slow and doesn't effectively transport a lot of data. For that reason, both Asymmetric and Symmetric encryption are often used in tandem with each other to provide high levels of security.

The key principle of Asymmetric encryption is that both parties have a pair of keys, a public key and a private key (aka as secret key). Their public key can be decrypted by their private key and vice versa *(Morkel, no date)*. Both parties publish their public key online and share it with everyone but keep their private key secret from all parties. If Bob wanted to send a message to Alice, Bob would encrypt the message using Alice's public key then send over the data over the internet to Alice. Alice could then decrypt the message using her private key as this decrypts the ciphertext and allows her to read the message, this process is shown in Figure 17.

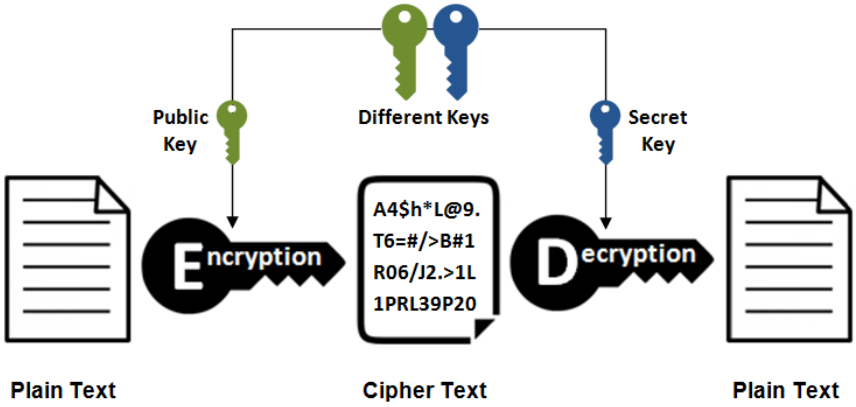


Figure 17: Asymmetric Encryption *(SSL2BUY, 2015)*

Asymmetric encryption also allows for a user to be specifically identified as if the user encrypts a message with his private key it could only over come from him and can only be decrypted by his public key *(Rouse, no date)*.

This whole process works if both the keys used are mathematically related. The primary algorithm in use today for this is RSA (Rivest, Shamir, Adelman. The creators of the algorithm) although this may change as RSA uses the relationship between prime numbers and how large primes are computationally very hard to factorise. However, if a computer fast enough (Quantum Computer) was set to this it could make very easy work of factoring the large prime numbers *(Morkel, no date)*.

RSA typically uses large keys of 1024 bits or 2048 bits, however for the benefit of this explanation in Figure 18 the numbers are significantly smaller. First the algorithm must decide on two numbers p and q (p = 2, q = 7) these are multiplied together to find the modulus number (14) a list is then created of numbers between 1 and 14 and all numbers that have common factors with 14 are removed (leaving 6 numbers in 14s case) this is now an array with length 6. We then find the coprime with the length and the modulus to find the key number (5). Now that we have both numbers the encryption can begin. The decryption number is worked out by taking the common modulus number length (6) and the encryption number (5) and working out what number can be multiplied by 5 and factored to leave a remainder of 1. This can be tested with a list of say 5 and filtered to the numbers that satisfy the condition (11). *(J, 2017)*

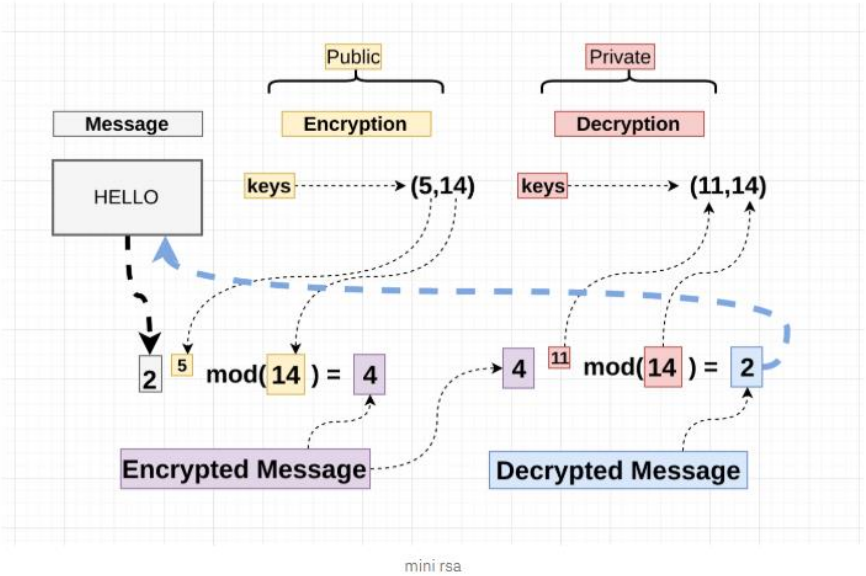


Figure 18: RSA Algorithm (J, 2017)

2.2.5 Hash Functions

Hash functions although not a method of Symmetric or Asymmetric encryption are used widely in cryptography for multiple reasons. Hashing is not like encryption methods covered in this report as these functions are only one way by design. If a malicious actor could reverse a hash function then it would be totally useless for the multiple jobs it does *(Daniel, 2018).*

Hashing works by taking in any length of data and returning a uniformly sized piece of data (often known as a fingerprint) *(Sobti and Geetha, 2012)*. There are many hash algorithms that can be used including SHA-256, MD5, BLAKE2 and more *(Daniel, 2018)*. They all have a different algorithm attached to them for different jobs. One of the most common uses or hashing functions is online password storage and it is better to store passwords as hash than it is to store the password in plaintext. This works because the hash function will always generate the same output if the same input is supplied and if the hashed password matches the one in the database then that user is authenticated *(Sobti and Geetha, 2012)*.

The primary reason for including hashes in this paper is for data integrity. For ensuring data is unaltered you can hash it and send the data over a channel (say the internet) then the receiver can hash the same file and if the hashes match the data has not been tampered with *(Meyers and Jernigan, 2018)*. For ensuring data integrity, it is important that whatever algorithm you use has considered the possibility of Hash Collisions.

A Hash Collision is where you can find another bit of data that gives the same computational hash; if this is possible then the hash algorithm is considered broken as data integrity is no longer possible with the algorithm. A malicious hacker given a hash must not be able to find a pair such that hash 1 is the same as hash 2 *(Sobti and Geetha, 2012)*. An example of this can be found in Figure 19. Many algorithms have fallen short on this including MD5 and SHA1 *(Marc, 2017).*

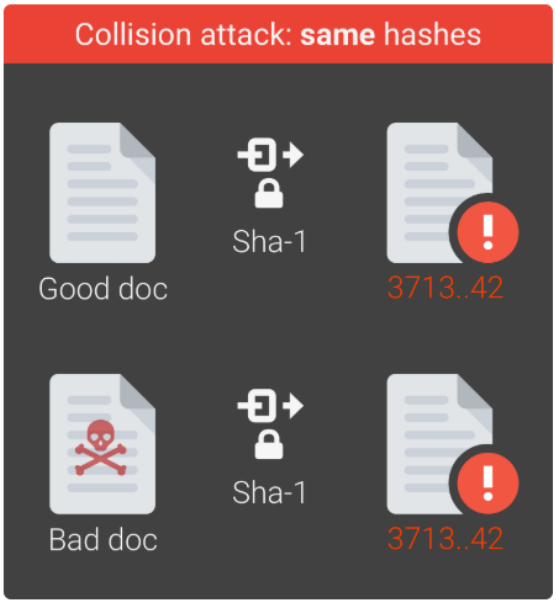


Figure 19: Hash Collision (Marc, 2017)

Another consideration is that the hash must be impossible to derive the original plaintext much like standard encryption. However, this proves an issue when it comes to password storage due to the individuals that create passwords. A hash always generates the same output meaning the people that use common passwords all have the same hash, this is exploited heavily in what is called a rainbow table attack. A malicious attacker might hash a bunch of common passwords and scan the list of hashes leaked from a database with the ones he/she pre-hashed. If the passwords match then you have essentially decrypted that hash which is a major problem. This problem is often alleviated by adding what is called a salt, which is a random set of characters that is appended or prepended to the plaintext before it is hashed, therefore the hash is completely different from the standard plaintext hash *(Arias, 2018).*

The hashing algorithm SHA-256 works similarly to block ciphers where it takes a block of data made up of the original input, padding (zeros) and the message length all in binary. It will then declare a set of common variables and keys that can be used in the hash rounds (64 for SHA-256). It will then perform a set of shifting and XOR operations per round using the keys and predeclared variables. An example of a round can be found in Figure 20 and the formulas used for the round can be found in Figure 21. This research would benefit from further investigation into how different hashing algorithms work where required as all hashing methods differ.

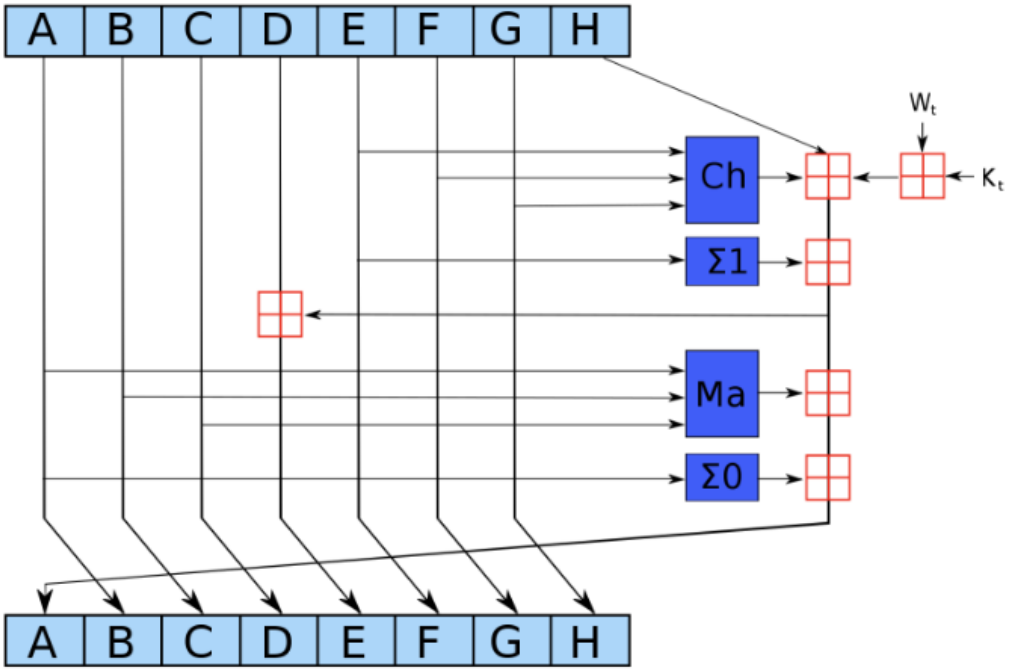


Figure 20: Singular round of SHA-256 (Anand, 2017)

```
Ch(E, F, G) = (E AND F) XOR ((NOT E) AND G)
Ma(A, B, C) = (A AND B) XOR (A AND C) XOR (B AND C)
    ∑(A)    = (A >>> 2) XOR (A >>> 13) XOR (A >>> 22)
    ∑(E)    = (E >>> 6) XOR (E >>> 11) XOR (E >>> 25)
     +      = addition modulo 2^32
```

Figure 21: Code of shift functions for SHA-256 round (Anand, 2017)

2.3 Conclusion

This section covered significant relevant research to the field of cryptography - and, specifically, in this project, the Enigma's story, internal function, breaks, strength and operation were explored.

The main weakness of the Enigma was also noted as no letter being able to self-encipher, meaning the cribbing method where a piece of plaintext could be matched to a piece of ciphertext could result in the Bombe cracking the Enigma in relatively short order. The following sections will keep this point firmly in mind considering the improved Enigma and how it eliminates this issue.

This section also studied the world of modern encryption and how it works in depth including the DES family and AES. AES is the main comparison point from here on in as it is the current standard and yet to be broken.

Hashing is also discussed here including SHA-256's full function, although MD5 later developed is not. The hashing section also analyses attacks on hashing and why MD5 is no longer used.

Finally, Asymmetric encryption is discussed as it is the main form of today's key distribution and internet encryption used.

The next section discusses how the improved Enigma can fix the key flaws of the original Enigma and discusses the relevant requirements related to the further development of the encryption methods.

3 Requirements

This section aims to highlight the key deliverables that must be developed for the project, explains how the Enigma machine functions from a programming perspective and expands core decisions that lead to the development of the improved Enigma. These deliverables are explained in depth with most illustrated with UML Use case diagrams. This section also explains core requirements that the developed software must abide by with explanations on how it conformed to them before delving into any constraints that delayed the project.

3.1 Product Deliverables

This project is primarily targeted at making post-war improvements to the Enigma to see if the Germans could have made it stronger. However, it also has an educational perspective due to the multiple implementations and visual look of the Enigma, and the possibility to show how the Enigma enciphered a letter though each step of the machine.

The primary deliverable for this project is an Enigma machine which must function as Enigmas used in WW2 which enciphers a letter using all the functions of the Enigma key including the rotors and plugboard. Both the Mk1 Enigma and the Kreigsmarine variant were developed for this project. The use case diagram for this is supplied in Figure 22 Below.

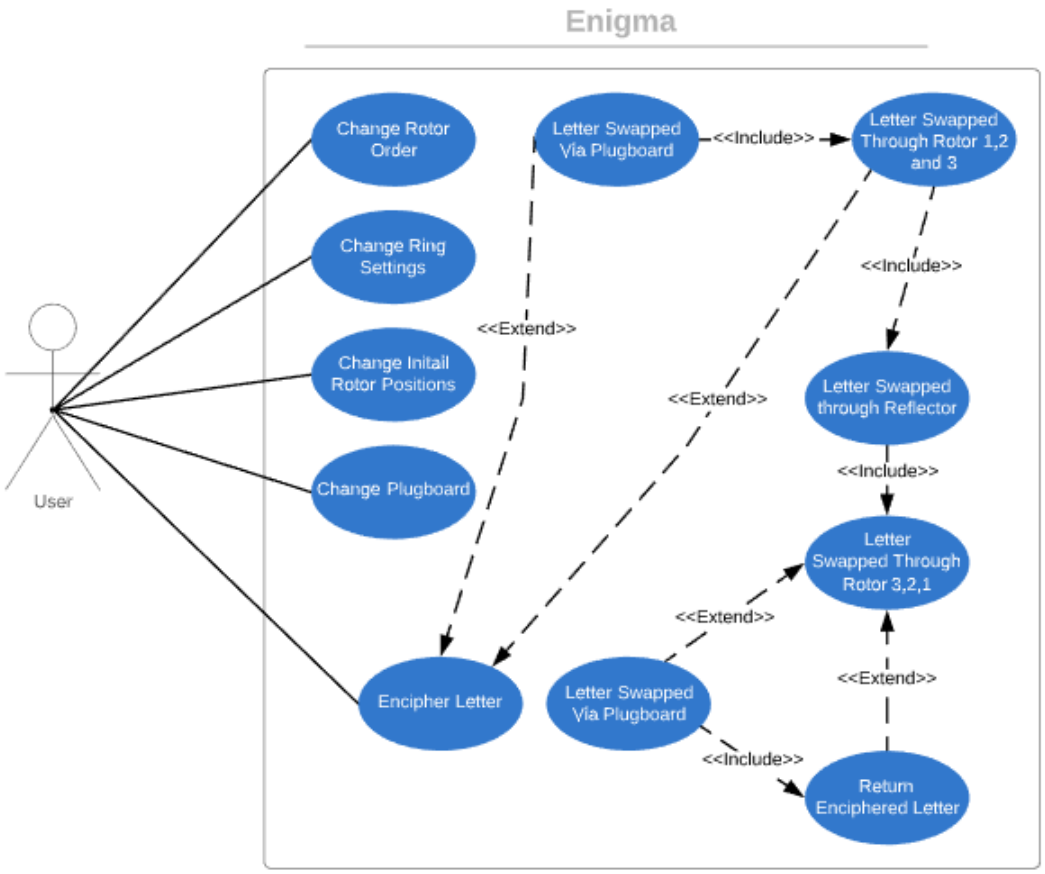


Figure 22: Enigma Use Case Diagram

In order to prove or disprove the research question the developer had to analyse different ways of improving the Enigma's core design. This was a significant departure from what was stated in the Project Initiation Document (PID) as the initial plan was to create the Typex machine. However, this limited the freedom of the rest of the project to devise new ways of improving the Enigma, so the

scope was changed as a result. Multiple different solutions were drawn up to consider for implementation of the new improved Enigma which can be found in Table 1 below:

Improvement	*Advantages*	*Disadvantages*
Add More Rotors	- Adds Key Space	- N/A But doesn't solve key weaknesses
Create a switchboard based off the plugboard	- Adds Key Space - Solves Same Letter Enciphering Issue - Possible during WW2	- Tedious to set up - Possible Decryption problems
Remove the limitations of plugs	- Potential increase in key space depending on plugs used	- Key Space could be massively reduced (See Figure 4)
Bitwise XOR A separate key (Stream Cipher)	- Increase Key Space - Solves Same Letter Enciphering Issue	- Key Distribution issues (per message key)
Add Special Characters	- Add Key Space	- Possible Decryption issues
Create an Enigma Block Cipher	- Solves Same Letter Enciphering issue - Adds Key Space - Adds huge amount of complexity and security	- Time restraints - Possibly slow - Key Distribution issues (Per Message Key)
Enable Missing Rotor Connections	- Solves Same Letter Enciphering issue - Possible during WW2	- Possible reduction in Key Space - Possible Decryption issues - Unlikely to encipher as same letter

Table 1: Enigma improvements

The following methods were seriously considered; creating an Enigma block cipher where the Enigma key swaps setting per round for a set of rounds (based on an external key) until the plaintext has fully diffused into the ciphertext. This seems likely to be the most secure route to take, however the amount of time required to implement this would have exceeded the time given for this project. This method should be considered in further detail later as it seems the most promising method of using modern computing to securely improve the Enigma. A bitwise XOR function was also considered, however it was felt that it wouldn't improve the Enigma more that it would have done by just adding an extra layer on top to get past the weaknesses of the Enigma itself. As a result, the XOR function was not chosen for the final design.

The Enigma that had some wires disabled in the rotor so that a letter might not swap would solve the issue of no letter being encrypted as itself. However, this also leads to a plausible reduction in key space and if the allied forces had obtained the new wiring diagrams then it is possible they could have designed a Bombe that could have skipped past these positions in a crib. It also seems like it would have been unlikely to encipher a letter as itself that often. This idea was not chosen although it had similarities to another idea based on the plugboard.

The switchboard idea is based on binary digits and a secondary plugboard. If a switch for a letter say “E” is on then the next time the user presses “E” it would encipher as “E” and switch the switch for “E” off and vice versa. This idea would solve the issue of no letter being enciphered as itself but could implement some decryption issues, although these could be solved by sending a binary message key within the message. This method was selected, and a use case can be found in Figure 23 below.

Figure 23: Improved Enigma Use Case Diagram

To check resistance against the Bombe machine it is necessary to check if a piece of ciphertext can result in a crib and, for that reason a cribbing device was developed. This requirement changed from the PID as it was noted by the developer that creating an entire Bombe was not required to test resistance against the Bombe machine and therefore only a cribbing device would suffice. You can see the use case diagram in Figure 24.

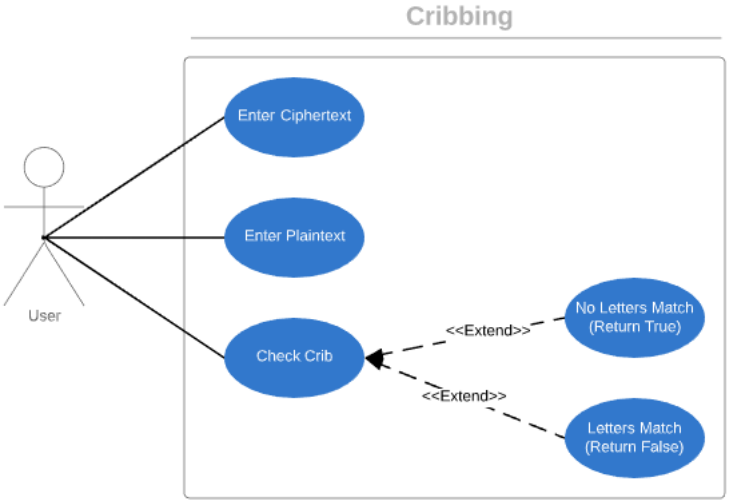


Figure 24: Crib Checker

The final deliverables for this project are the Modern encryption methods, mainly delivered in the form of JavaScript libraries (referenced in the implementation section), the SHA-256 transformation (see 2.2.5) and the AES transformation (see 2.2.3) as per the literature review.

The methods developed were AES, MD5 and SHA-256 and these are developed specifically for the comparison between Modern encryption and historical encryption such as the Enigma.

An additional objective for this project is to develop a Morse Code Encoder as messages were sent via radio in WW2 (intercepted by Y-Stations) akin to the WW2 version of the modern-day internet. A use case diagram can be found in Figure 25.

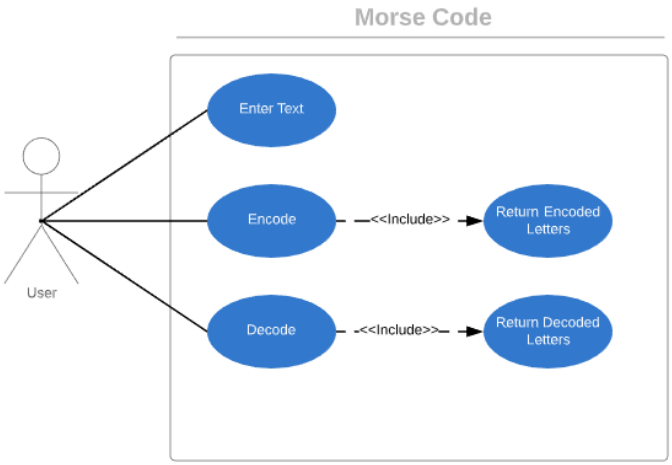


Figure 25: Morse Code

Additional python scripts were developed where required to generate tests for the evaluation. These scripts are talked about in the implementation and evaluation section of this report but are not specifically part of the main implementation of the project.

3.2 Functional Requirements

3.2.1 Interfaces

The interfaces must be easy to understand, and as a result all of the interfaces were designed to be as simple as possible whilst maintaining the complexity of the program running in the background. The developed software made use of web design allowing for clear labelling of the different sections for all methods in use. These interfaces should all follow the same overarching design to-not confuse the end user and to make it familiar. In the case of the Enigma it is vital that the user can tell the difference between all the inputs as the key is comprised of a number of different sections that appear similar (rotor position, ring position). The developer will make use of colour as a primary way of differentiation. This should not disadvantage other users so the colour scheme should be distinct for a colour-blind individual to have access.

3.2.2 Functional Capabilities

For development to be fully functional all encryption methods must function as their real-life counterparts. For example, the Enigma machine must use all of the encipherment settings used in WW2 and be able to swap all of the valid settings when required. This should allow full encipherment of the messages typed but also full decipherment by using the inverse of the encipherment. This should be true for all the encryption methods (AES, Morse Code, improved Enigma and Enigma Kreigsmarine). Hashing however should always result in the same hash that is generated from the plaintext in that it should not be able to be decrypted by running the inverse. The cribbing machine must run though all of the letters in the plaintext and compare them to the ciphertext, if there is a match it should return either true or false for the crib. Assuming the methods are correctly implemented the outputs of the machines should return the ciphertext ready for testing by the crib machine and the analysis.

3.2.3 Safety

No data is stored on the Enigma or other encryption devices as the entire program should remain client side. By not storing user data the project is abiding by all relevant Data Protection rules such as GDPR. It is worth noting that these old encryption methods are broken and as a result are insecure so no user should utilise these methods for encryption of any personal or sensitive data; this is explained in the instruction manual given to the user (found in appendix B). It is also worth noting that although the project is attempting to improve the security of the Enigma, the improved Enigma may have security weaknesses the developer did not find so should also not be used to encipher personal information.

3.2.4 Reliability

The developed encryption methods must error where appropriate, for example, on the Enigma if a plug is connected incorrectly the user must be warned about this otherwise the encipherment will be wrong. In the developed Enigma the message box will show an error if this happens and the user must refresh the page. The developed methods must be stable, and all testing should be completed to ensure that it doesn't crash and if there are any remaining bugs they should be noted. Hashing must reliably produce the same result every time or the algorithm is wrong, and cribs generated from the cribbing machine must output correct results.

3.2.5 Quality

The quality of the developed software must be flawless as an incorrect encipherment will skew the results, thus bugs must be ironed out before testing and evaluating the strength of the Enigma. The code quality should be such that another developer can look at the code and understand what is happening at each stage with clean code comments in case any modifications are required.

3.3 Design Constraints

The biggest constraint to this project is time, if there is further opportunity an improved Enigma based around a block cipher could be developed. Fortunately, the planning that was laid out in the PID allowed for accurate time management for all stages of the project, although throughout the process of the project this changed due to a mixture of project changes and external factors. The biggest external factor was undoubtedly the global pandemic (COVID-19) that shutdown the working environment for this project, shifting all learning and development to online. The developer was also considered high risk by the NHS. As a result, multiple adjustments were made to the project plan on the developer's side and on the University's such as giving a grace period of two weeks to all students to compensate. As referenced in the PID there may have been security restrictions on the content the developer wanted to research. However, none were found.

3.4 Conclusion

The project requirements and developmental progression are illustrated whilst the discussion above explores the different ways of developing a more secure Enigma to become the "improved Enigma". The chosen method was to implement a binary switchboard which self-enciphers a letter if the switch is on, and, not if the switch is off. Also explained are the functions that each device in development is required to have to be classed as complete and went into depth with the use of UML use case diagrams to illustrate the key software elements.

To conclude: this section provided a summary of key external factors that limited this project and their impact. The next section delves further into the designs in the UML diagrams above and explains how a developer may be able to develop the encryption methods.

4 Design

This section aims to lay out full development plans for the software, explaining any elements of development which may be difficult to understand, it explores the Enigmas plugboard and the improved Enigma's switchboard in depth. This section also explores the use of Modern encryption and how it may be implemented into the software before explaining how this software will be evaluated to answer the research question.

4.1 Software Design

4.1.1 Overall System

The design of the overall system is disjunctive as all of the different encryption methods make up a different system; for example, the Enigma does not rely on the AES encryption. The only exception for this rule is the cribbing device that relies on the Enigma's ciphertext to test its strength. Due to this separation, the user should be able to choose a method and how they use it.

4.1.2 Enigma

The Enigma's development must be comprised of all the different sections of the Enigma that allows it to encipher a message. The rotors each have their own alphabet of singular letters connected to each other via wire. For the development there is a need to simulate these wires and swap the letters.

First, it must be ensured that the correct alphabet is used for the rotor design (see Figures 2 for examples) then select the local alphabet wished to complete the swap with (for simplicity the English alphabet). From a development perspective this is now quite a simple problem as these alphabets can be stored in character arrays that can be compared by using the index of both of the arrays. This index can now act as the different wire mappings for the swaps of the Enigma, to swap a letter it is now possible to find the input letters index in the base array then find that index in the rotors alphabet (an example of this can be shown in Table 2). This can then be repeated for all three (or four) rotors in the Enigma depending on the version used. Using arrays also solves the problem of rotor order as the alphabet used by the rotor can be swapped by either having a selection which swaps in and out the array which is in use or by changing the alphabet manually in the code.

Index	*Alphabet*	*Rotor Alphabet*
1	A	E
2	B	K
3	C	M
4	D	F
5	E	L
6	F	G
7	G	D
8	H	Q
9	I	V
10	J	Z
11	K	N
12	L	T
13	M	O
14	N	W
15	O	Y
16	P	H
17	Q	X
18	R	U
19	S	S
20	T	P
21	U	A
22	V	I
23	W	B
24	X	R
25	Y	C
26	X	J

Table 2: Rotor Wiring Arrays

Ring settings can be selected by pre-shifting this array and keeping the notch at the same index; this allows for the notch to switch position depending on the number of shifts completed on the Enigma. These arrays allow for flexibility for the rotor, including the per encipherment rotor shift that can be completed by shifting the cipher alphabet forward a position for each number. These also allow for pre-set numbers that can be used as part of the key to take effect, say we have rotor 1 set to “15”, the ring to “4”, then the rotor alphabet will be shifted a total of 19 places before the first encipherment through the rotor.

The last remaining part of the key is the plugboard. This can be attempted in multiple ways however, the simplest way is to have an array of "plug-spaces". A plug space by default is always blank or "not connected" but when a plug-space has a letter selected it *is* connected. This seems simple however, what if the receiving plug is actually connected to a different letter? This would be a major bug in encipherment. For this reason, it is necessary for the plugboard to issue a ping to the letter that it is allegedly connected to. This can be shown in Figure 26 where the first plug ("G") issues a ping for the letter say "A" and if the second plug ("A") is linked to the first plug it will respond to the ping saying its valid, otherwise it will return an error.

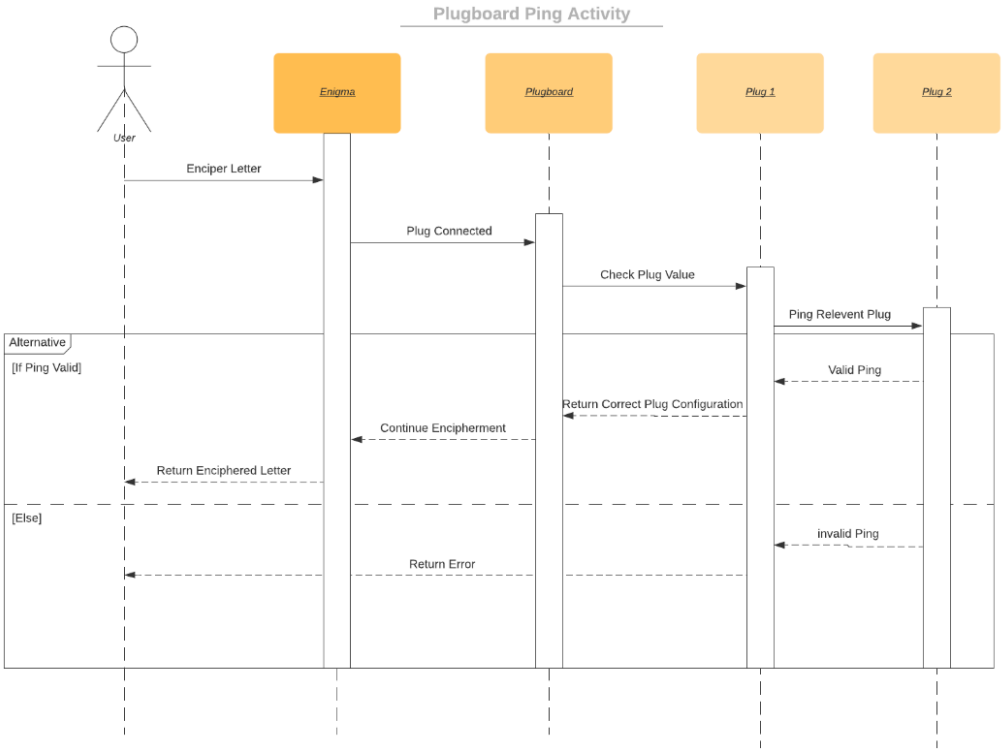


Figure 26: Plugboard Activity Diagram

To use all these methods small additional parts of the program have to be designed, similar to the method used to input the base plaintext letter. This can be done by a simple key press on a keyboard, however, if the user was to input an invalid character (space, etc.) then you would have to filter this out as the Enigma cannot encipher these letters. The best way to do this is to have a programmed keyboard that when pressed returns the letter it has stored. This allows the letter to pass directly through an Enigma method which includes the rotors array and the plugboard pings. There is also ample opportunity to make this a different keyboard layout to remain historically accurate, in the Enigma's case this would be the "QWERTZ" layout.

In conclusion, the user types on the keyboard the letter they want to encipher, then the plugboard will run a check looking for connected plugs and if it finds them it will complete a test on each connected plug. From there, the letter is swapped through a plug if the plaintext letter is connected in the plugboard and put though the letter arrays. The first letter array shifts before the encipherment is made (acting as the rotation of the rotor) and if it encounters a notch position it will do the same to the middle and end rotors. These rotors may also be shifted and the notch shifted by the key positions supplied beforehand. Once it has reached the end of the rotors it will hit the reflector array and return back through the rotor array in the opposite direction, the final enciphered letter is then checked for a plug connection and switched if required, before returning the final enciphered letter.

The UI for this must remain clean as it is important that the user knows exactly what to input and where for this reason a UI (User Interface) sketch for how this Enigma should look can be found below in Figure 27.

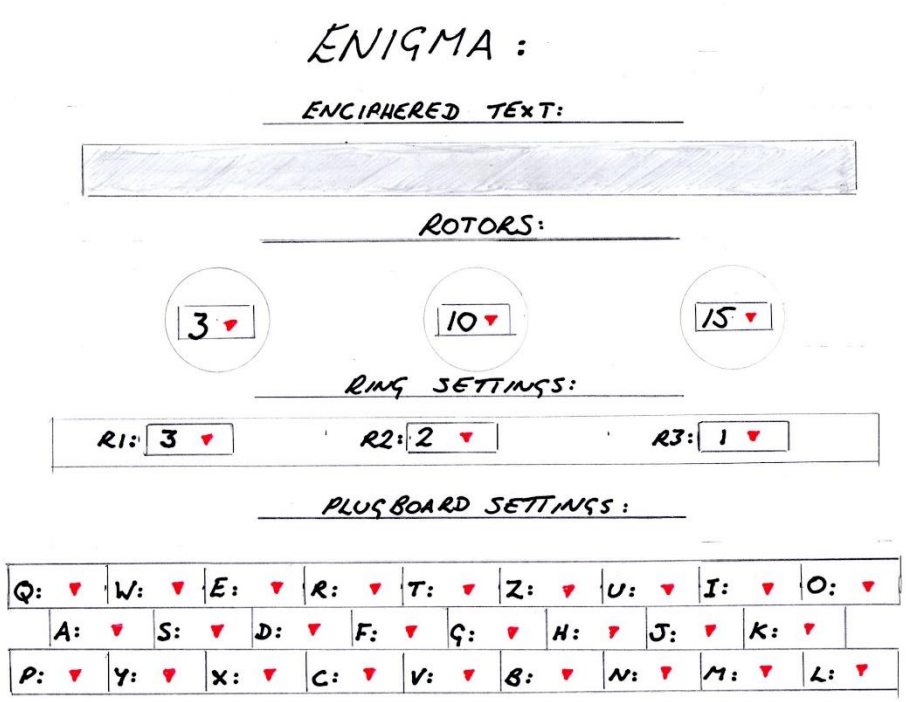


Figure 27: Enigma Wireframe

4.1.3 Improved Enigma

The improved Enigma is evolved from the Enigma in the previous section, so it has all the same inputs and functions, however, the improved Enigma also has the switchboard function which takes the original input and checks against an array of different binary inputs. For example, if a user was to set the letter “A” to on and then enciphered “A” it would return “A” then switch the “A” switch off. The entire encipherment process still steps forward with this, meaning the fast rotor moves 1 place and if it hits its notch position it will push the next rotor etc. An example of this process can be found in Figure 28.

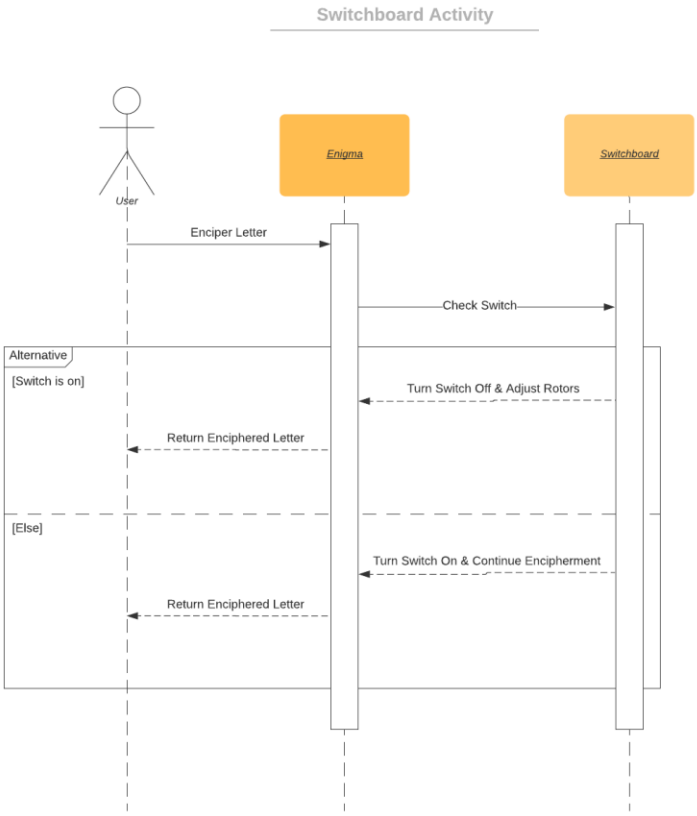


Figure 28: Switchboard

Decipherment may be an issue when enciphering a message using the improved Enigma, for that reason it would be advantageous to keep to hand a note of where the positions of the letters that self-encipher are. An example could be that “Hello” is enciphered to “TEELY” if the decipherment of this could result in a lot of different possibilities so sending a binary message along with it or asymmetrically using Modern encryption would allow full decryption using an Enigma without switchboard. For example, the binary representation of “TEELY” would be “01010” where the 1’s represent the same letter as the plaintext. Of course, these numbers need to be protected themselves which could result in a significant weakness in this improved Enigma. An example of how the switchboard may look can be found in Figure 29, it should be located below the plugboard on the full Enigma.

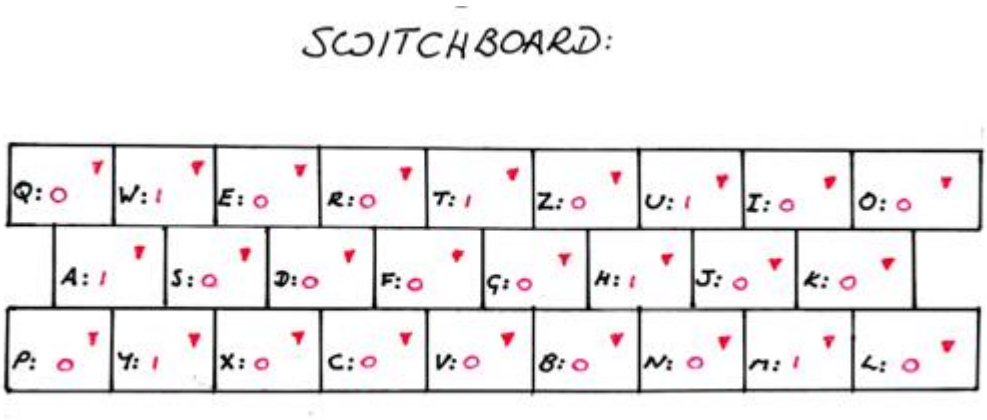


Figure 29: Improved Enigma Switchboard Wireframe

4.1.4 Cribbing

Cribbing works by taking the values of the plaintext and ciphertext and converting them both to character arrays. Then you loop around the current state of the arrays and see if any characters match; if they do it's an invalid crib, if not it's a possible valid crib. You do this for the entire length of the input and if the ciphertext inputted is longer than the plaintext, the ciphertext array can be shifted to see if any of the possible combinations contain a crib. A simple UI Implementation of this can be found in Figure 30.

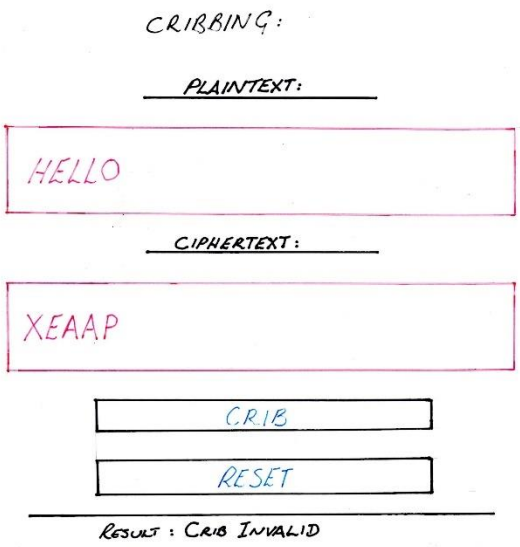


Figure 30: Cribbing Wireframe

4.1.5 Modern encryption

Modern encryption must be implemented to the highest possible amount of accuracy as a lot of the attacks that are done using AES are a result of poor implementations of the algorithm. For this reason, there are two options when implementing AES. Firstly, you could self-develop the system by going through the same steps as mentioned in the literature review (Substitute Bytes, mix columns, mix rows, add round key). This entire process is standardised so what the developer could do is to take a string of data, split down the data into different blocks then transform that data with a supplied key. The second, and debatably better way to implant is to use pre-existing AES libraries. The library

handles all the data itself and is a lot less likely to get the transformations incorrect. The same UI layout can be used to take all the inputs (this can be found in Figure 31).

Hashing follows the same process as there is a single way to implement this code which makes all the code standardised, so it is worth implementing pre-existing code or a library to ensure accuracy within the methods used. In regard to the methods developed for this project MD5 and SHA-256 were chosen, this is as MD5 is considered broken and SHA2-256 isn't yet. This is a valid comparison point to make later in the report, however, any Hashing implementation could be used. The SHA-256 transformation is explained in the literature review above (see 2.2.5). A UI of Hashing can be found in Figure 32.

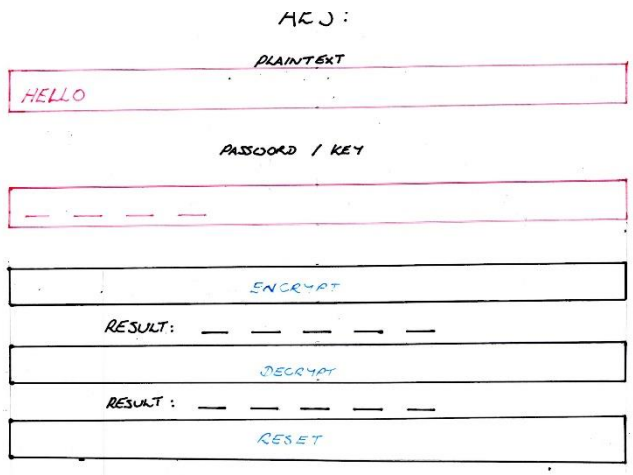


Figure 31: AES Wireframe

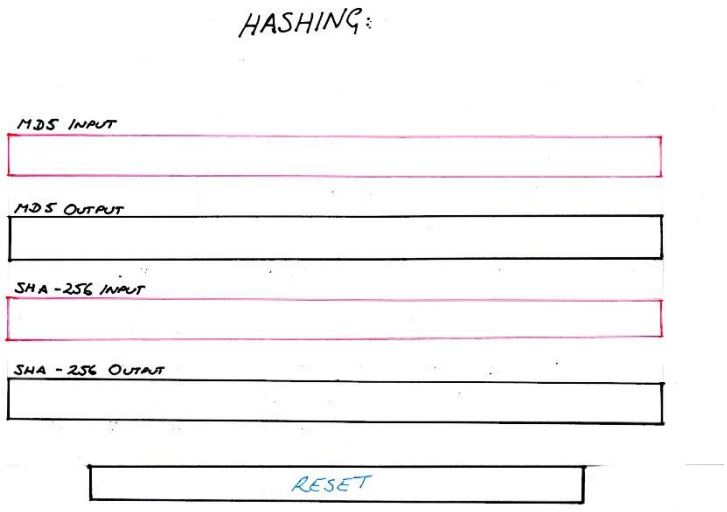


Figure 32: Hashing Wireframe

4.1.6 Morse code

Morse code should take a plaintext input, convert it to a character array, then perform a look up for each character in a separate array. From this it will construct a new string of Morse characters and return the result to the user.

4.1.7 Evaluation

To evaluate the strength of the improved Enigma the developer should encipher a number of different words/passwords or phrases to test against the cribbing machine. If the cribbing machine finds a valid crib then this method becomes less secure as it is still plausibly vulnerable to the Bombe machine; if not then it is secure. However, additional tests and statistics can be pulled from this data such as the self-encipher frequency and any patterns that emerge from the new encipherment. This could be provided in an Excel spreadsheet or as a set of graphs. The comparisons to Modern encryption will be done mostly theoretically although if the developer has time they could test multiple cryptographic attacks on Modern encryption methods such as the aforementioned rainbow table for hashing. Now that the reader is accustomed to how the project is designed the next section of the report covers the project from the planning phase to full development.

4.2 Conclusion

This section goes into depth on how each piece of software may be developed and underpins how that development may function. This is especially relevant in the design of the plugboard where the developer spent significant time discussing how to validate the plug. This is explored in detail in the next section when implemented into code. Also laid out is the UI for the encryption methods considering all inputs and key ways the design could be implemented into code. The following section discusses full development of each of the encryption methods described in this design section including how the project was managed in an efficient timely manner.

5 Implementation & Testing

This section aims to cover the implementation of the software from start to finish including any project management that the developer used to keep the project on time. This section could be used to guide a developer attempting to reconstruct this code including the improved Enigma's switchboard and any other form of encryption such as AES. By the end of this section conclusions may start to be drawn about how the improved Enigma performed against modern encryption by using the cribbing machine.

5.1 Implementation

5.1.1 Project Management

To implement this project the developer had to consider different methodologies to complete the system. The developer chose an incremental approach with work split down into features and implemented on top of the main codebase. This methodology is suitable for this project as a proportion of the main code base is already split up into different sections. Although, the developer opted to use incremental methodology, tools were adopted from the agile methodology to help keep the project managed throughout. One of the main tools used was a Kanban board, the developer could rearrange tasks depending on the state of progress. This was helpful with such a large project. The developer opted to use the Trello web app for management of the Kanban board and had prior experience with this software (although many others exist). The reader can find a screenshot of the Kanban board used below in Figure 33 and the full evolution of the board in the appendix A.

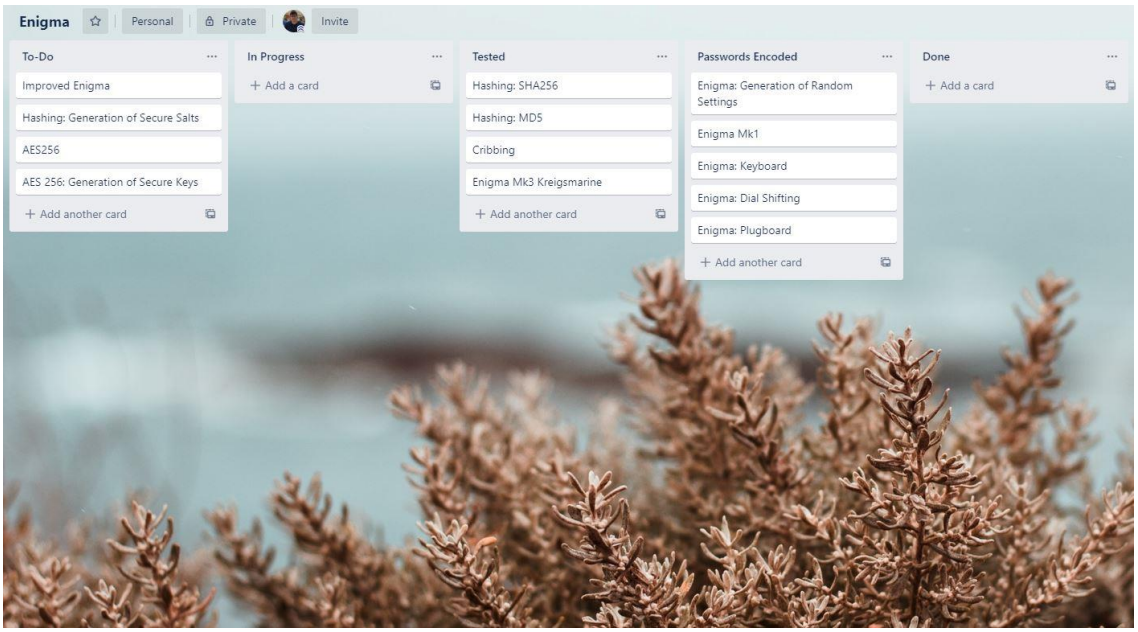


Figure 33: Kanban Board

The developer also decided to use version control; this allowed reversion and maintenance of multiple backups of the code in case anything of issue with the code or an implementation of a feature was functional. There are many different methods used in industry and the developer has had personal experience with Subversion (SVN) and GitHub. GitHub is the primary version control software used in industry. The developer chose GitHub to gain some industrial experience of managing a large project through the software and for its additional bug tracking features. A sample of some developmental commit history can be found in Figure 34 and the full commit history can be found in the appendix A.

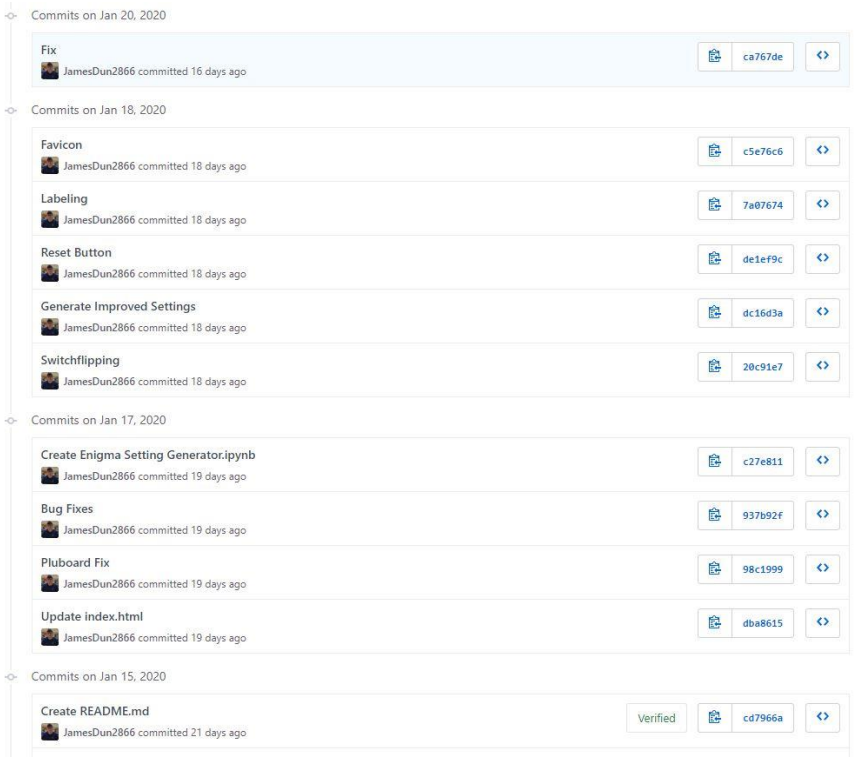


Figure 34: GitHub commits

GitHub allows bug tracking which was of assistance when testing the various encryption implementations throughout as the developer was able to note details of what the bug was and how it was fixed. An example of this can be found in Figure 35.

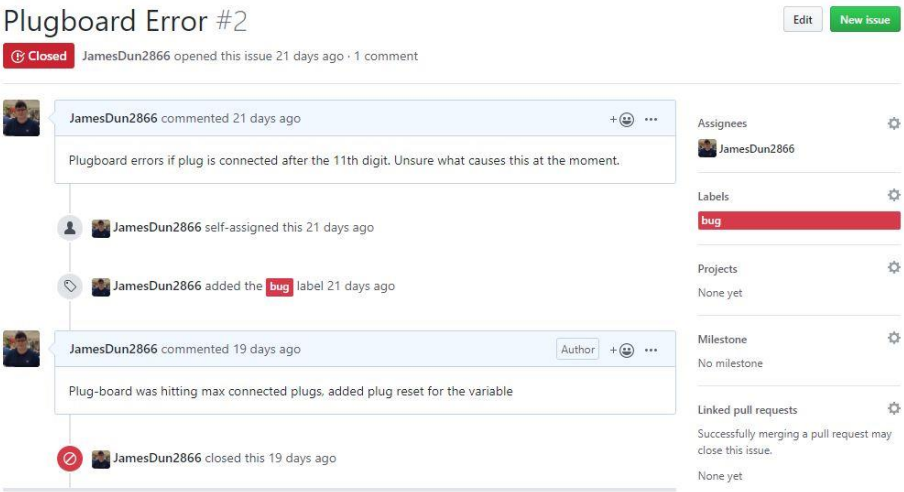


Figure 35: GitHub bug tracking

To keep track of time the developer kept the Gantt chart from the PID updated. Although this went through adjustments throughout due to the developer's time shifting, this happened on multiple occasions. Firstly, the developer was actively seeking a Graduate Scheme post university meaning more time was spent at the beginning looking for roles than conducting the literature review. This stopped after the developer accepted a graduate position and was able to refocus on the project. The secondary major shift in time was due to the coronavirus pandemic that closed the working

environment for this project, although the disruption was managed as the developer had concluded development at that point.

The final project management procedure in place was the fortnightly project meetings the developer had with the project supervisor who ensured the developer was keeping on track for the project and offering useful suggestions based on progress. The developer's first supervisor left just after the project development, and as a result there was a small transition period to the new supervisor, which barely impacted on progress as it was anticipated and factored in months beforehand.

Before any programming could commence the developer had to choose a development language. There are no required platforms for development, so the developer had free rein on what language to use. A list of considered languages can be found in Table 3.

Platform	*Language*	*Experience*	*Online Resources*
Windows	C#	High	High
Multiplatform	Python	High	High
Web Application	JavaScript	Medium	High
Android	Java	Low	Medium

Table 3: Considered Programming Languages

The developer had experience in most of these languages but liked the idea of developing a web app due to the portability and high compatibility with all platforms allowing for the Enigma to be used from anywhere. This allows the developer's Enigma to be portable, as in the war. For development of a web technology JavaScript, HTML and CSS can be used.

5.1.2 User Interfaces

The first stage of development was to replicate the UI's from the wireframes and make the website appear visually appealing. The main style took inspiration from the Developers personal website (https://jamesjduncan.dev/) although the entire UI was overhauled to fit with the encryption methods. The Enigma was by far the hardest UI to develop as there are multiple different elements requiring design, in particular the keyboard caused the developer some difficulty. Due to the amount of spacing and requirements for the keyboard, significant CSS tweaking was undertaken to get it right, as a result the developer can now call the "keyboard" class from the CSS and nest "Keyboard__key" to create a row of keys then call "keyboard__key" inside it to create a keyboard row with a key. The position of the keyboard is fixed so all the keys correspond to the website whilst scrolling; each one of these keys can be coded to activate a different function which is required to encipher a single letter. Most of the CSS is styling of these buttons but the key difference is each one has a different height, width and max width. An example use of this can be found in Figure 36.

```
<div class="keyboard">
  <!-- Create Keyboard DIV-->
  <div class="keyboard__keys">
    <!-- Create Keyboard key rows and stlye them -->
    <button type="button" class="keyboard__key" id="Q" onclick="Q()">Q</button>
```

Figure 36: Creating keyboard

With the keyboard developed it is possible to complete the UI quite simply by adding HTML dropdowns for the rotor number and plugboard settings, making the output box a read only text area and similar dropdowns for the ring settings and plugboard. The styling of these elements is different to ensure that the UI is user friendly. The colour is also distinct allowing for the colour-blind to see all

of the settings in use. The developed UI for the Enigma can be found in Figure 37 (Enigma Mk1) and Figure 38 (Enigma Kreigsmarine).

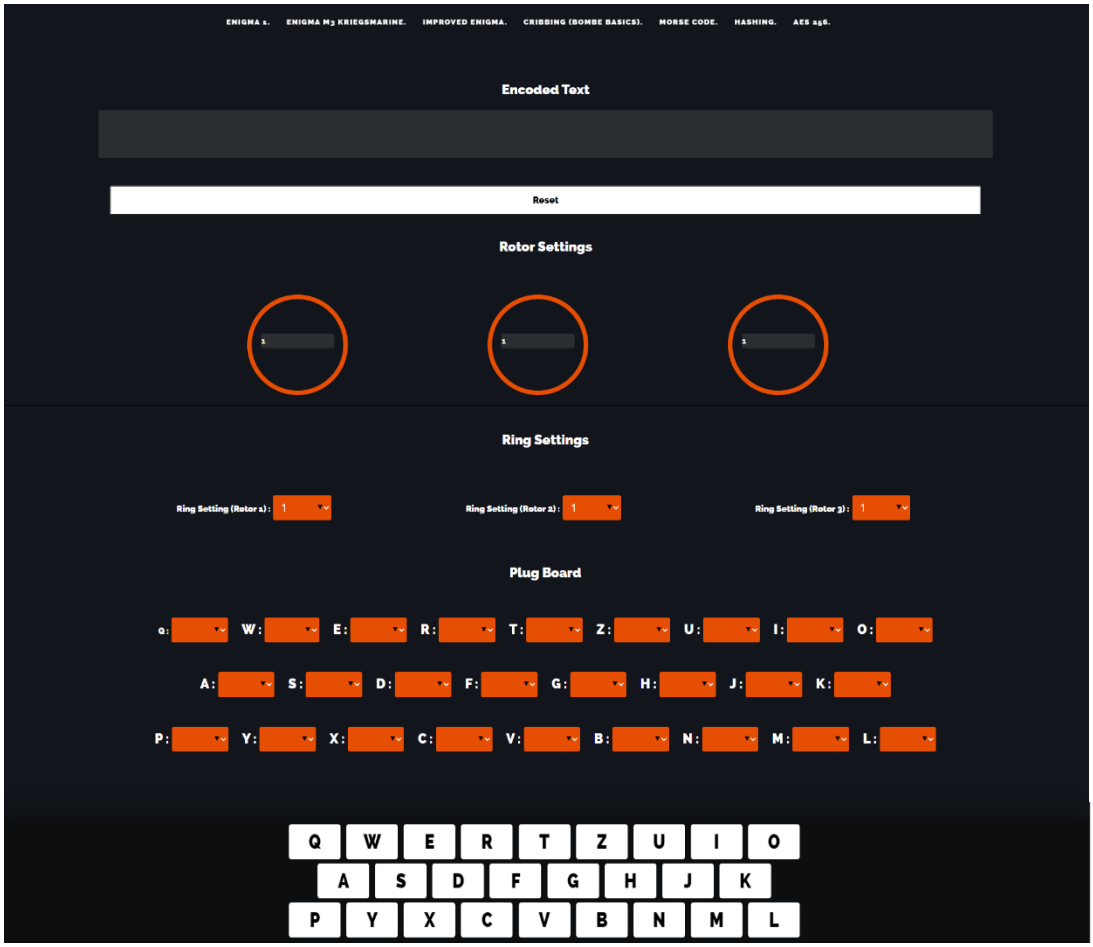


Figure 37: Enigma Machine UI

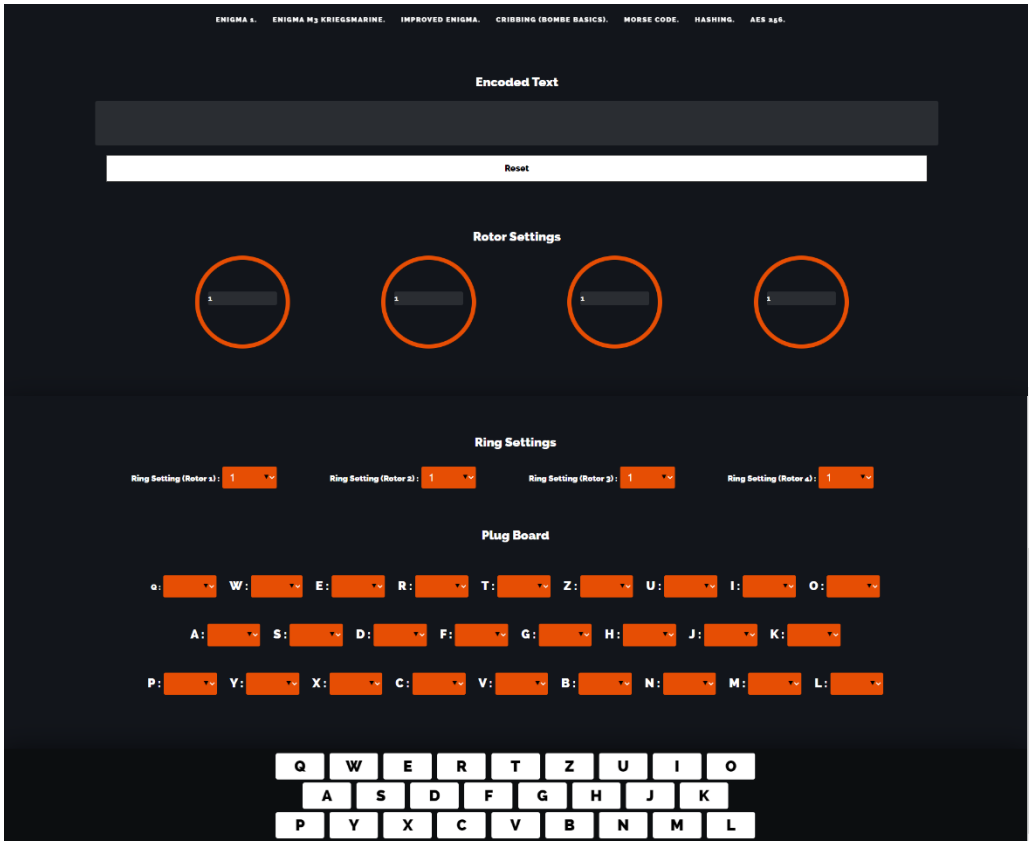


Figure 38: Enigma Kreigsmarine

The switchboard for the improved Enigma was based on the plugboard's HTML and CSS so only required minor tweaking to implement into the UI. Additionally, the other UI's were simple to implement as they mostly consisted of HTML text areas for input and output. The UI for the switchboard can be found in Figure 39, UI for the cribbing machine can be found in Figure 40 and UIs for Hashing, Morse Code and AES can be found in Figures 41, 42 and 43.

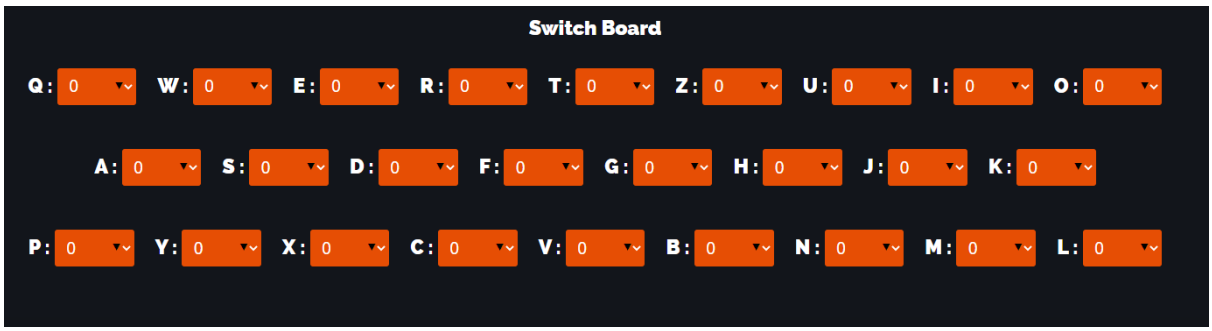


Figure 39: Switchboard UI

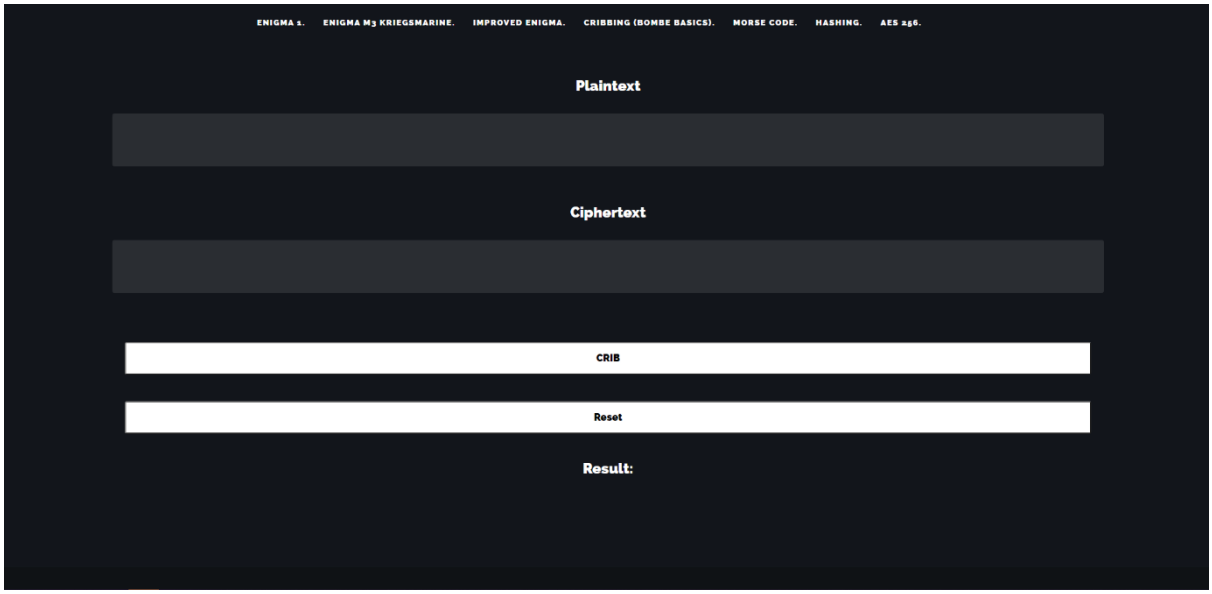


Figure 40: Cribbing Machine

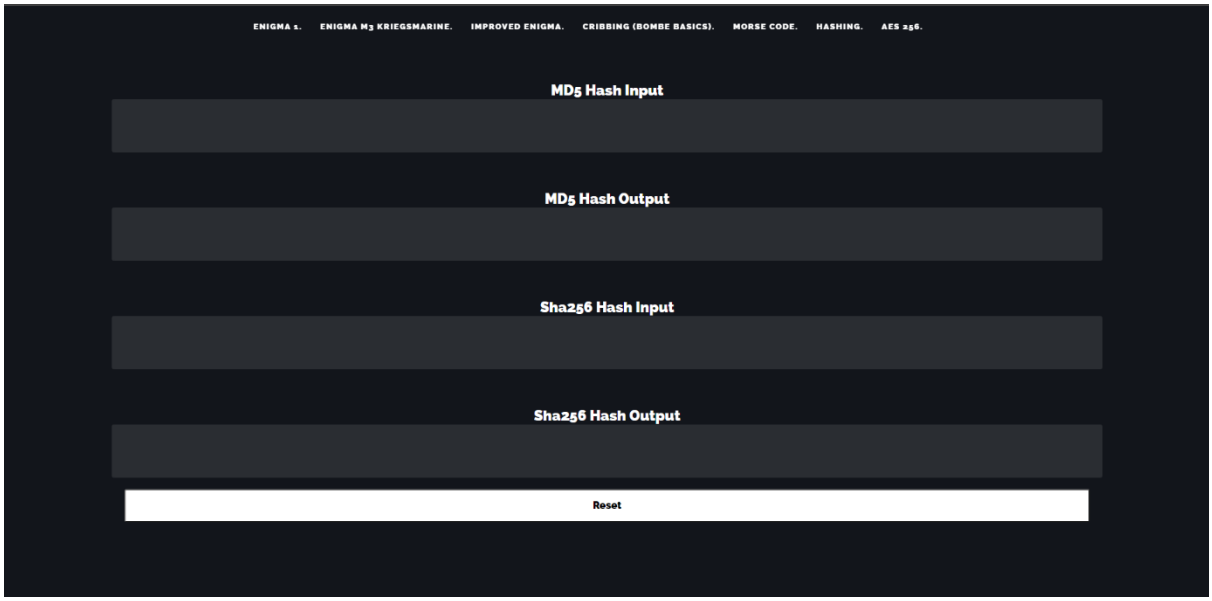


Figure 41: Hashing

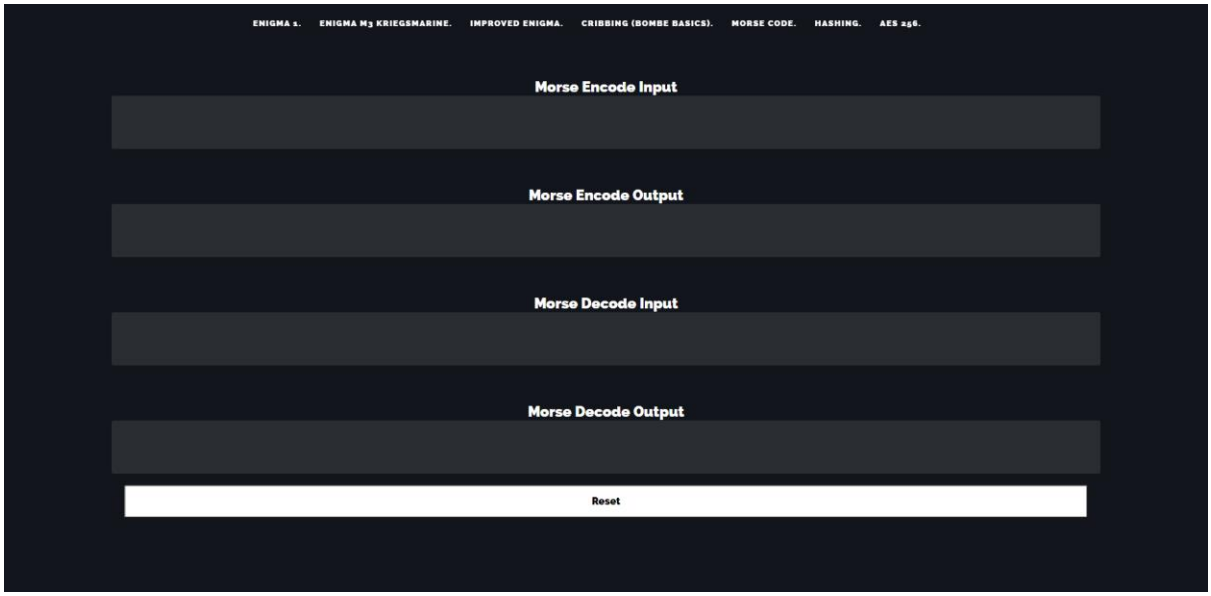


Figure 42: Morse Code

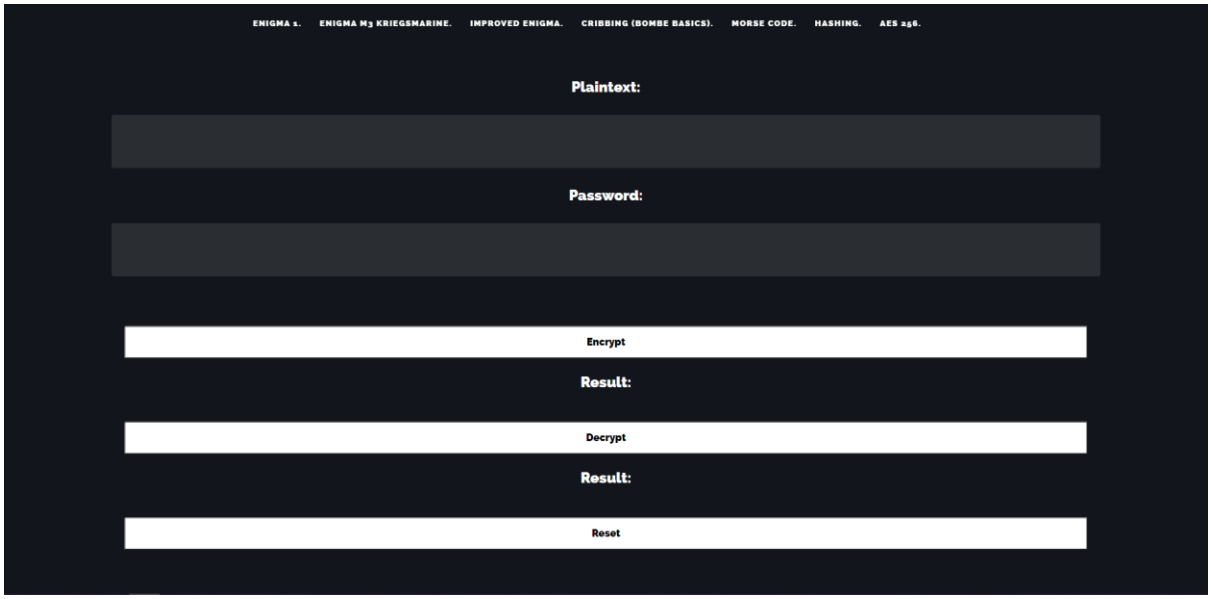


Figure 43: AES

5.1.3 Enigma

Now that the UI for the Enigma is complete it is possible to start looking at how the Enigma functions. The developer spent some time plotting paths on how an Enigma may function in Excel and found that the reflector transformation was not well explained in any existing literature. The developer needed to comprehend exactly how the reflector functioned before programming it. Tests were undertaken to get the pattern correct before a solution was found as the reflector acts as a rotor itself that doesn't move; this solution was more illustrative than simply stating that the reflector returned the current to the rotors. If a future developer is required to develop this Enigma it would be worth considering some of these Excel tests; the correct result of this experiment can be found in Figure 44.

Lang	Base	Rotor IIIC		Rotor IIC		Rotor I C		Reflector B		Rotor I C		Rotor IIC		Rotor IIIC	
Key	Output	Key	Output	Key	Output	Key	Output	Key	Output	Key	Output	Key	Output	Key	Output
A	A	A	U	A	H	A	D	A	Y	A	D	A	H	A	U
B	B	B	Q	B	Q	B	M	B	R	B	M	B	Q	B	Q
C	C	C	N	C	Z	C	T	C	U	C	T	C	Z	C	N
D	D	D	T	D	G	D	W	D	H	D	W	D	G	D	T
E	E	E	L	E	P	E	S	E	Q	E	S	E	P	E	L
F	F	F	S	F	J	F	I	F	S	F	I	F	J	F	S
G	G	G	Z	G	T	G	L	G	L	G	L	G	T	G	Z
H	H	H	F	H	M	H	R	H	D	H	R	H	M	H	F
I	I	I	M	I	O	I	U	I	P	I	U	I	O	I	M
J	J	J	R	J	B	J	Y	J	X	J	Y	J	B	J	R
K	K	K	E	K	L	K	Q	K	N	K	Q	K	L	K	E
L	L	L	H	L	N	L	N	L	G	L	N	L	N	L	H
M	M	M	D	M	C	M	K	M	O	M	K	M	C	M	D
N	N	N	P	N	I	N	F	N	K	N	F	N	I	N	P
O	O	O	X	O	F	O	E	O	M	O	E	O	F	O	X
P	P	P	K	P	D	P	J	P	I	P	J	P	D	P	K
Q	Q	Q	I	Q	Y	Q	C	Q	E	Q	C	Q	Y	Q	I
R	R	R	B	R	A	R	A	R	B	R	A	R	A	R	B
S	S	S	V	S	W	S	Z	S	F	S	Z	S	W	S	V
T	T	T	Y	T	V	T	B	T	Z	T	B	T	V	T	Y
U	U	U	G	U	E	U	P	U	C	U	P	U	E	U	G
V	V	V	J	V	U	V	G	V	W	V	G	V	U	V	J
W	W	W	C	W	S	W	X	W	V	W	X	W	S	W	C
X	X	X	W	X	R	X	O	X	J	X	O	X	R	X	W
Y	Y	Y	O	Y	K	Y	H	Y	A	Y	H	Y	K	Y	O
Z	Z	Z	A	Z	X	Z	V	Z	T	Z	V	Z	X	Z	A

Figure 44: Enigma Encipherment (Reflector)

Now that the Developer understood how the letters may swap it became possible to code the Enigma method to swap the letters. The development process for this Enigma is the same for the Kreigsmarine Enigma which has the additional rotor. The developer first has to declare the arrays of the rotor alphabets then swap this letter by finding the index, then make a method that takes the plaintext letter (from the keyboard) and compare the indexes to swap the letter. JavaScript has a useful method that can be used here called “indexOf()” which takes the input and searches the array for that input; if it finds it, it will return the index integer of that letter. This can be utilised to find the letter in the rotor array and make the swap. The declaration of arrays can be found in Figure 45 and the code for the swaps in 46. It is notable that to change the rotor you need to change the values in the arrays. This will allow reshuffling, reorder and creation of your own Enigma alphabets.

```
//The Alphabet, what the rotors compare against
Orignal = ["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"];
//Rotor 1 the last dail the letter enters
Dial1 = ['E', 'K', 'M', 'F', 'L', 'G', 'D', 'Q', 'V', 'Z', 'N', 'T', 'O', 'W', 'Y', 'H', 'X', 'U', 'S', 'P', 'A', 'I', 'B', 'R', 'C', 'J'];
// Rotor 2 the second Dail the letter enters
Dial2 = ['A', 'J', 'D', 'K', 'S', 'I', 'R', 'U', 'X', 'B', 'L', 'H', 'W', 'T', 'M', 'C', 'Q', 'G', 'Z', 'N', 'P', 'Y', 'F', 'V', 'O', 'E'];
//Rotor 3 the first Dial the letter enters
Dial3 = ['B', 'D', 'F', 'H', 'J', 'L', 'C', 'P', 'R', 'T', 'X', 'V', 'Z', 'N', 'Y', 'E', 'I', 'W', 'G', 'A', 'K', 'M', 'U', 'S', 'Q', 'O'];
// ReflectorB Send the letter through all the Rotors again
ReflectorB = ['Y', 'R', 'U', 'H', 'Q', 'S', 'L', 'D', 'P', 'X', 'N', 'G', 'O', 'K', 'M', 'I', 'E', 'B', 'F', 'Z', 'C', 'W', 'V', 'J', 'A', 'T'];
```

Figure 45: Enigma Arrays

```
//ENIGMA LETTER SWAPPING
BasePos = Dial3.indexOf(letter); // find the base position of the letter in the dail
letter = Orignal[BasePos]; //swap it with the letter in the orignal alphabet
BasePos = Dial2.indexOf(letter); // find the base position of the letter in the dail
letter = Orignal[BasePos]; //swap it with the letter in the orignal alphabet
BasePos = Dial1.indexOf(letter); // find the base position of the letter in the dail
letter = Orignal[BasePos]; //swap it with the letter in the orignal alphabet
BasePos = ReflectorB.indexOf(letter); // find the base position of the letter in the reflector
letter = Orignal[BasePos]; //swap it with the letter in the orignal alphabet
BasePos = Orignal.indexOf(letter); // find the base position of the letter in the original alphabet
letter = Dial1[BasePos]; //swap it with the letter in the Dial
BasePos = Orignal.indexOf(letter); // find the base position of the letter in the original alphabet
letter = Dial2[BasePos]; //swap it with the letter in the Dial
BasePos = Orignal.indexOf(letter); // find the base position of the letter in the original alphabet
letter = Dial3[BasePos]; //swap it with the letter in the Dial
```

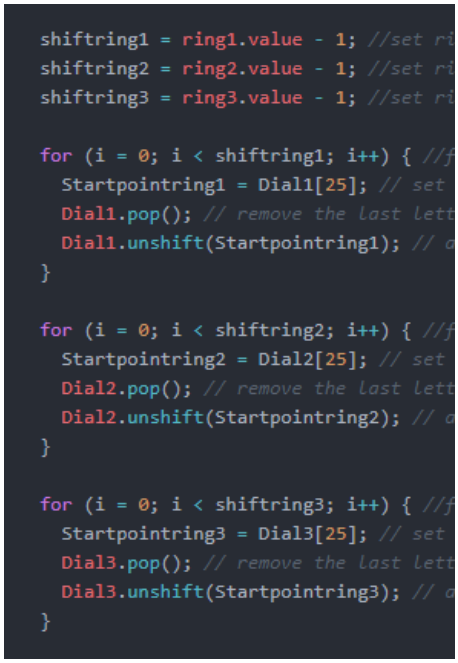
Figure 46: Enigma Letter Swaps (Where letter is the plaintext letter)

Now swapping is possible the arrays must shift depending on the settings. For the standard rotor shifts that shift on a keypress the following code in Figure 47 can be used. By default - the value of rotor 1 will increase once as this must increase every keypress (rotor 1 denoted as R1) and for each number in the rotation it will set a variable to the last letter of the array then use the pop() method to take that letter out of the array. This letter can then use the unshift() method to add it back to the front of the array. In essence this letter has shifted a singular position and it is possible to do this for each number in the shifts.

```
Newvalue = R1.value++; //increase
for (i = 0; i < R1.value; i++) {
  Startpoint = Dial3[25]; // set
  Dial3.pop(); // remove the last
  Dial3.unshift(Startpoint); // ad
}
```

Figure 47: Shifting the rotor letters

Once the code for shifting the letters has been established you can then reset the number to 1 when it hits 26, in turn resetting the rotor. You can then have" if statements" where you want the index of the notch to kick the next rotor forward. These values can then use the same code in Figure 47 to shift the relevant arrays before entering the code in Figure 46. The values of the ring settings can actually follow the same code as it shifts the notch as the rotor rotates and this can then be pinned in place by an "if statement" much like the aforementioned "if statements" for the notch index. The ring shifts can be found in Figure 48.



```
shiftring1 = ring1.value - 1; //set ri
shiftring2 = ring2.value - 1; //set ri
shiftring3 = ring3.value - 1; //set ri

for (i = 0; i < shiftring1; i++) { //f
  Startpointring1 = Dial1[25]; // set
  Dial1.pop(); // remove the last lett
  Dial1.unshift(Startpointring1); // a
}

for (i = 0; i < shiftring2; i++) { //f
  Startpointring2 = Dial2[25]; // set
  Dial2.pop(); // remove the last lett
  Dial2.unshift(Startpointring2); // a
}

for (i = 0; i < shiftring3; i++) { //f
  Startpointring3 = Dial3[25]; // set
  Dial3.pop(); // remove the last lett
  Dial3.unshift(Startpointring3); // a
}
```

Figure 48: Ring shifting

Finally, the most complex part of the code is the plugboard check, as mentioned in the design section. It's main function is to check the value of the opposite plug in order to check a valid connection. The following code uses references to the IDs assigned to the plugboard plug itself so in Figure 49 you can find how a plug was declared in the code base to help the reader understand how the plugboard checks each plug.

```
<label class="label2">Q :</label>
<!-- Each plug has a name of "Plug"
<select id="plugQ" name="Plug">
  <!-- Plug default is blank or not
  <option selected></option>
  <option>W</option>
  <option>E</option>
  <option>R</option>
  <option>T</option>
  <option>Z</option>
  <option>U</option>
  <option>I</option>
  <option>O</option>
  <option>A</option>
  <option>S</option>
  <option>D</option>
  <option>F</option>
  <option>G</option>
  <option>H</option>
  <option>J</option>
  <option>K</option>
  <option>P</option>
  <option>Y</option>
  <option>X</option>
  <option>C</option>
  <option>V</option>
  <option>B</option>
  <option>N</option>
  <option>M</option>
  <option>L</option>
</select>
</span>
```

Figure 49: Plug Declaration

Each plug is part of the name “Plug” so the full set of 26 plugs is a list with a separate ID for each plug. The first task, the “plugcheck” is to establish if the plug is set or not; if not set, it will not check if the plug has a connection on the other side. If it does it will set a variable “plugmatch” to equal the string of “plug” and the set letter. An example of this is that if Q is connected to E then the output of “plugmatch” will be “PlugE”. The Code can then loop around 26 times (letters in the alphabet) and search for the recipient plug. A new variable is declared “x” which takes each plug in the loop and runs the following check on it. If x’s id (if X is on plug E then it would be PlugE) is the same as the plug match variable then it will check the connection by checking if the value of x is the same as the base plug. If it is then the connection is valid and if not then it’s invalid. Depending on the Enigma used it is necessary to increment the total plug value and check at the end of the Plug Check method if the plugs exceed the plug connection limit. The code for this can be found in Figure 50.

```
if (plugQ.value !== "") { // if plug is set (not empt
  plugmatch = "plug" + plugQ.value; //create a varibl
  for (i = 0; i < 26; i++) { //loop around all 26 let
    var x = document.getElementsByName('Plug')[i]; //
    if (x.id === plugmatch) { //if the plug in the it
      if (x.value !== "Q") { //if the opposing plug a
        plugerror = true; // return an error
        break; //break the loop
      } else {
        plugcount = plugcount + 1; // if the opposing
        break; //break the loop
      }
    }
  }
}
```

Figure 50: Plug Checking

These plugboard checks are then implemented into a method which checks all 26 plugs and returns either true (where all plugs are valid, and the count does not exceed the connection limit) or false (where plugs are invalid, or connection limit is exceeded). This ensures that all plugs have a valid connection but does not swap a letter. To do this a similar code can be implemented in the main Enigma method where it finds the plug using the same x value checker but instead of checking for validity it takes the value of x and swaps the letter directly. This can be found in Figure 51. The full Enigma method contains all of this code and takes a singular plaintext letter and returns a single ciphertext letter. The full code for this method can be seen in Figure 52.

```
    // The below code swaps the letter to a connected
    plugfind = "plug" + letter; // Find the unique ID
    for (i = 0; i < 26; i++) { //loop round the alphab
      var x = document.getElementsByName('Plug')[i];
      if (x.id == plugfind) { // if the plug matches t
        if (x.value !== "") { //if the plug is set cha
          //(This is possible as all plugs are valid b
          letter = x.value; //swap the letter
          break; //break the loop
        }
      }
    }
```

Figure 51: Plugboard Swapping

```
function Enigma(letter) {

  if (PlugTest() == true) { //if the above method (plugboard) returned no errors then the enigma can encrypt a letter

    for (i = 0; i < R2.value; i++) { //for each value in the Rotor, shift the array to new positions
      Startpoint0 = Dial2[25]; // set the value of Startpoint to the last letter in the rotor array
      Dial2.pop(); // remove the last letter of the array
      Dial2.unshift(Startpoint0); // add the last letter of the array back at the front of the array
    }
    for (i = 0; i < R3.value; i++) { //for each value in the Rotor, shift the array to new positions
      Startpoint23 = Dial3[25]; // set the value of Startpoint to the last letter in the rotor array
      Dial3.pop(); // remove the last letter of the array
      Dial3.unshift(Startpoint23); // add the last letter of the array back at the front of the array
    }

    shiftring1 = ring1.value - 1; //set ring shifts to the ring value -1(default is 1 and I do not want to shift an addtional letter)
    shiftring2 = ring2.value - 1; //set ring shifts to the ring value -1(default is 1 and I do not want to shift an addtional letter)
    shiftring3 = ring3.value - 1; //set ring shifts to the ring value -1(default is 1 and I do not want to shift an addtional letter)

    for (i = 0; i < shiftring1; i++) { //for each value in the shift, shift the array to new positions
      Startpointring1 = Dial1[25]; // set the value of Startpointring1 to the last letter in the rotor array
      Dial1.pop(); // remove the last letter of the array
      Dial1.unshift(Startpointring1); // add the last letter of the array back at the front of the array
    }

    for (i = 0; i < shiftring2; i++) { //for each value in the shift, shift the array to new positions
      Startpointring2 = Dial2[25]; // set the value of Startpointring1 to the last letter in the rotor array
      Dial2.pop(); // remove the last letter of the array
      Dial2.unshift(Startpointring2); // add the last letter of the array back at the front of the array
    }

    for (i = 0; i < shiftring3; i++) { //for each value in the shift, shift the array to new positions
      Startpointring3 = Dial3[25]; // set the value of Startpointring1 to the last letter in the rotor array
      Dial3.pop(); // remove the last letter of the array
      Dial3.unshift(Startpointring3); // add the last letter of the array back at the front of the array
    }

  // The below code swaps the letter to a connected plug if one is connected
  plugfind = "plug" + letter; // Find the unique ID of the current letters plug
  for (i = 0; i < 26; i++) { //loop round the alphabet
    var x = document.getElementsByName('Plug')[i]; // loop round all the plugs
    if (x.id == plugfind) { // if the plug matches the Unique ID of the letter set above then the plug is found
      if (x.value !== "") { //if the plug is set change the letter to the value of the plug
        //(This is possible as all plugs are valid before entering the enigma method ensuring all combinations dont have to be checked against
        letter = x.value; //swap the letter
        break; //break the loop
      }
    }
  }

  Newvalue = R1.value++; //increase the first rotors value on every call of the function
  for (i = 0; i < R1.value; i++) { //for each value in the Rotor, shift the array to new positions
    Startpoint = Dial3[25]; // set the value of Startpoint to the last letter in the rotor array
    Dial3.pop(); // remove the last letter of the array
    Dial3.unshift(Startpoint); // add the last letter of the array back at the front of the array
  }



  incremental = false; //set a bool value to be false (used to ensure correct incrementation between Rotors)
  if (R1.value <= 27) { // if the value of Rotor 1 is less than 27 (26 displayed on rotor)
    if (R1.value == 27) { // if the value of rotor 1 is 27 (26 displayed on rotor)
      R1.value = 1; //reset the value of rotor 1 to 1
    }
    if (R1.value == 26) { // this is the set kickpoint of the rotor, if you add a rotor that has addittional kickpoints add an additional "OR"
      R2.value++; //increment the next rotor
      for (i = 0; i < R2.value; i++) { //for each value in the Rotor, shift the array to new positions
        Startpoint2 = Dial2[25]; // set the value of Startpoint to the last letter in the rotor array
        Dial2.pop(); // remove the last letter of the array
        Dial2.unshift(Startpoint2); // add the last letter of the array back at the front of the array
      }
      incremental = true; // set incremental to be true
    }
  }
  if (R2.value <= 27) { // if the value of Rotor 2 is less than 27 (26 displayed on rotor)
    if (R2.value == 27) { // if the value of rotor 2 is 27 (26 displayed on rotor)
      R2.value = 1; //reset the value of rotor 1 to 1
    }
    if (R2.value == 26 && incremental == true) { // this is the set kickpoint of the rotor, if you add a rotor that has addittional kickpoints
      R3.value++; //increment the next rotor
```

```
        for (i = 0; i < R3.value; i++) { //for each value in the Rotor, shift the array to new positions
          Startpoint3 = Dial1[25]; // set the value of Startpoint to the last letter in the rotor array
          Dial1.pop(); // remove the last letter of the array
          Dial1.unshift(Startpoint3); // add the last letter of the array back at the front of the array
        }
        incremental = false; // set incremental to be False
      }
    }
    if (R3.value >= 27) { // if the value of rotor 3 is 27 (26 displayed on rotor)
      R3.value = 1; //reset the value of rotor 1 to 1
    }

    //ENIGMA LETTER SWAPPING
    BasePos = Dial3.indexOf(letter); // find the base position of the letter in the dail
    letter = Orignal[BasePos]; //swap it with the letter in the orignal alphabet
    BasePos = Dial2.indexOf(letter); // find the base position of the letter in the dail
    letter = Orignal[BasePos]; //swap it with the letter in the orignal alphabet
    BasePos = Dial1.indexOf(letter); // find the base position of the letter in the dail
    letter = Orignal[BasePos]; //swap it with the letter in the orignal alphabet
    BasePos = ReflectorB.indexOf(letter); // find the base position of the letter in the reflector
    letter = Orignal[BasePos]; //swap it with the letter in the orignal alphabet
    BasePos = Orignal.indexOf(letter); // find the base position of the letter in the original alphabet
    letter = Dial1[BasePos]; //swap it with the letter in the Dial
    BasePos = Orignal.indexOf(letter); // find the base position of the letter in the original alphabet
    letter = Dial2[BasePos]; //swap it with the letter in the Dial
    BasePos = Orignal.indexOf(letter); // find the base position of the letter in the original alphabet
    letter = Dial3[BasePos]; //swap it with the letter in the Dial



    plugfind2 = "plug" + letter; // Find the unique ID of the current letters plug
    for (i = 0; i < 26; i++) { //loop round the alphabet
      var x2 = document.getElementsByName('Plug')[i]; // loop round all the plugs
      if (x2.id == plugfind2) { // if the plug matches the Unique ID of the letter set above then the plug is found
        if (x2.value !== "") { //if the plug is set change the letter to the value of the plug
          //(This is possible as all plugs are valid before entering the enigma method ensuring all combinations dont have to be checked against their counterpart plug)
          letter = x2.value; //swap the value
          break; //break the loop
        }
      }
    }
    return letter; // return the encrypted letter
  } else { // if the Above method (Pluboard) Returned an error print error
    return "PlugBoard Error: Please check all connections and ensure you only have 10 pairs connected";
  }
}
```

Figure 52: Full Enigma Method

5.1.4 Improved Enigma

The improved Enigma follows the same method as the base Enigma above, except this Enigma has the switchboard code. The switchboard code works off the same style code that the plugboard does for finding the relevant switch then taking the value of that switch. However, it only does this to find the state of the switch and unlike the plugboard it cannot have an invalid connection. When the code finds the plug, it checks if its state is in 0 or 1, if it's in 1 it will return the original letter instead of the enciphered one then turn the switch off. On the other hand, if it's off it will return the enciphered letter and turn the switch on. The code for this can be found in Figure 52.

```
switchfind = "Switch" + orignalLetter; // find the un
for (i = 0; i < 26; i++) { //for each letter in the a
  var sw = document.getElementsByName('Switch')[i]; /
  if (sw.id == switchfind) { //if the switch ID is th
    if (sw.value == 1) { // if the value is true
      sw.value = 0;
      return orignalLetter; //return the original let
      break; //break out the loop
    } else { //if switch is false return the encrypte
      sw.value = 1;
      return letter;
    }
  }
```

Figure 52: Switchboard code

5.1.5 Cribbing

Cribbing was developed as per the specification in the design; the inputs were both made into character arrays and analysed to see if the characters matched or not. If there was no match, a variable will be incremented and if there's is a match it won't increment the variable. That variable can then be used to compare against the plaintext length. If the plaintext is the same size as the incremented number it's a valid setting, otherwise it is not a valid setting. The code for this can be found in Figure 53.

```
function Crib() {
  cipher = cipherText.value; //get the value of the cipherText
  cipherarr = new Array(...cipher); //turn the cipherText into a char array

  plain = plainText.value; // get the value of the plaintext
  plainarr = new Array(...plain); //turn the plaintext into an array
  f = 0; // set f to 0
  for (i = 0; i < cipherarr.length; i++) { //for each character in the cipherText
    if (cipherarr[i] !== plainarr[i]) { //if ciphetext does not  match the plaintext increment f
      f++;
    }
  }
  if (f === plainarr.length) { // if f is the same length as the plaintext then all characters are differ
    document.getElementById('result').innerHTML = 'Result: No Letters Match (Vaible Settings)';
  } else { //if f is not the same then its an invalid crib
    document.getElementById('result').innerHTML = 'Result: Letters Match (Invalid Settings)';
  }
}
```

Figure 53: Cribbing

5.1.6 Morse Code

The developer opted to use a library for Morse code as it wasn't a primary objective and the developer was running short of development time when Morse code was to be implemented. It is worth noting that even though a library was used for the implementation, the developer's design from the design section of this report was an accurate method of implementation as the chosen library uses this exact method. Syncthetics Morse Code JavaScript implementation *(Syncthetic, 2017)* was adopted which takes the input and encodes or decodes the message dependent on the method used; the code for this can be found below in Figure 54.

```
function Encode(f) {
  morseEncode = Object.create(MorseCode); //Create the object
  message = morseEncode.encode(f); //encode the message
  return message; //return the message
}

function Decode(f) {
  morseEncode = Object.create(MorseCode); //create the object
  dcode = morseEncode.morse(f, true); //decode the message
  return dcode; //return the message
}
```

Figure 54: Morse Code

5.1.7 Modern encryption

Modern encryption was primarily implemented using libraries, AES was implemented using CryptoJS *('CryptoJS', no date)* which has full JavaScript implementations of AES 128, 192 and 256 and can adapt depending on the key you provide. This was chosen as it was an effective way to implement the AES algorithm into JavaScript without implementation errors that may cause a lapse in security. The library supports a wide range of modes, but the default is cipher block chaining (CBC) and the code for this can be found in Figure 55.

```
function encrypt() {
  var encrypted = CryptoJS.AES.encrypt(document.getElementById("text").value, document.getElementById("pass").value); //encrypts using the value of the pla
  document.getElementById("result").innerHTML = encrypted; //sets value of label to the encrypted string
}

function decrypt() {
  var decrypted = CryptoJS.AES.decrypt(document.getElementById("result").innerHTML, document.getElementById("pass").value).toString(CryptoJS.enc.Utf8); //D
  document.getElementById("decrypted").innerHTML = decrypted; //sets value of label to the Decrypted string
}
```

Figure 55: AES Encryption

This code takes the text inputted, and the password/key value to encipher the block and return the encrypted message. SHA-256 was implemented with the use of JsSHA *(Caligatio, no date)* which takes the input and returns the SHA-256 Hash of the input supplied. This was used for much the same reason as the AES Library in that using it minimised the risk of implementation errors. The code for this can be found in Figure 56.

```
<script>
  //Sha256 is implmented with help from the jsSHA library which can be found here
  //https://github.com/Caligatio/jsSHA
  function Sha256hash() {
    var sha256 = new jsSHA('SHA-256', 'TEXT'); //Create a new hash of sha256 and text
    sha256.update(b.value); //hash the value of b
    var hash = sha256.getHash("HEX"); //output the hash as a HEX value
    return hash; //return the hash
  }
```

Figure 56: SHA-256 Hashing

MD5 hashing was harder to acquire as it is deemed insecure, however, the developer discovered a JavaScript implementation on stack overflow *(Powtac, 2009)* that could be used to implement MD5 hashing. This was adopted, however the developer did note that CryptoJs could also have performed this hashing function. MD5 works similarly to the SHA family but manipulates the input differently and returns a much smaller string.

5.1.8 Evaluation

For the evaluation of the Enigma it is required to test the strength using the cribbing machine; to do this a large sample was required so it was necessary to encipher a set of test words. The developer thought that using the most common passwords would give a suitably wide variety of test entries and if the encipherment was strong enough it might be able to protect some of these. The list used originated from Wikipedia *(10,000 most common passwords, no date)* as the developer did not mind if any of the content changed *(accessed January 2020)* as the content was not being used to back up a fact but to give an adequate test set. The developer also built a random generator for different Enigma settings which can be found in Figure 57.

```
from random import seed
from random import randint
import random
import string

for x in range (15):
    Rotor1 = randint(1, 26)
    Rotor2 = randint(1, 26)
    Rotor3 = randint(1, 26)
    RingSetting1 = randint(1, 26)
    RingSetting2 = randint(1, 26)
    RingSetting3 = randint(1, 26)
    Plugs = randint(0, 1)

    if (Plugs == 1):
        plugpairs = randint(1, 10)

    pairs = []
    for x in range (plugpairs):
        first = random.choice(string.ascii_uppercase)
        second = random.choice(string.ascii_uppercase)
        if (first != second):
            plugpair = first + second
            pairs.append(plugpair)
    pairstring = ""
    for x in pairs:
        pairstring += x
        pairstring += ' '

    if (Plugs == 0):
        print('R1:', Rotor1, 'R2:', Rotor2, 'R3:', Rotor3,'Ring1:', RingSetting1, 'Ring2:',RingSetting2, 'Ring3:', RingSetting3,
    else:
        print('R1:', Rotor1, 'R2:', Rotor2, 'R3:', Rotor3,'Ring1:', RingSetting1, 'Ring2:',RingSetting2, 'Ring3:', RingSetting3,
```

Figure 57: Enigma Setting Generator

Example output (R1: 8 R2: 25 R3: 14 Ring1: 5 Ring2: 20 Ring3: 22 Plugs Connected: XZ NW VL)

These passwords and random settings could then be used to create a large assortment of samples to be analysed in the evaluation. Finally, the Modern encryption methods can also be used to encrypt these passwords and evaluated later.

5.2 Testing

Testing is a major part of making sure that the program is functioning the way the developer wants and that there are no bugs affecting the program invalidating the output. Testing on this project took two forms: manual testing that the developer undertook during development and unit testing. An example of a unit test can be found in Figure 58 and a test plan in appendix C.

```
original = letter
BasePos = Dial3.indexOf(letter); // find the base position of the letter in the dail
letter = Orignal[BasePos]; //swap it with the letter in the orignal alphabet
if (letter != orignal)
{
  alert("Test Passed: Enigma Rotor 3 Enciphers to different letter")
}
else {
  alert("Test Failed: Rotor 3 does not encipher to differet letter")
}
```

Figure 58: Unit Test

Of course, testing has its limitations as the developer found at the mid project review with his supervisor. A bug was discovered that the developer had never considered would be an issue, the plugboard with valid connections errored the user after typing limited characters. This puzzled the developer and the issue was noted in a GitHub bug report to fix as soon as possible. It turns out that the plugboard count that was responsible for checking if the amount of connections exceeded the limit was not being reset and as a result would error after many encipherments. This bug was quickly

remedied although the developer noted that mass user testing would be crucial if the project was to be deployed - to allow for such bugs to be found and fixed.

5.3 Conclusion

The software developed met all of the specified requirements and follows the design section to result in a finished software solution.

The Enigma M1 and Kreigsmarine both function as intended and take all the required inputs for encipherment and decipherment of the message. The improved Enigma was implemented with full switchboard functionality, although, as referred to in previous sections there are some decryption issues which are discussed in the evaluation.

Modern encryption methods (AES, SHA-256 and MD5) were all implemented via libraries.

This section also discussed how the developer kept the project managed despite external factors.

Finally, the next section evaluates the results collected from the developed software per the evaluation section above.

6 Evaluation

This section critically explores how the improved Enigma performed against other methods of encryption. Also explained is how the improved Enigma can decrypt despite its decryption issues caused by the switchboard, additionally this section evaluates the software developed as a whole before coming to a conclusion.

6.1 Improved Enigma Decryption Issues

As predicted previously the improved Enigma does have many decryption issues that make the system harder to decipher and read the message. Encryption with the improved Enigma is easy as you just set up the machine (plugboard, switchboard, rotors, etc…) and type your message. So the encryption for "Hello" (using R1: 10, R2: 11, R3: 12, Ring1: 1, Ring2: 2, Ring3: 3, Plugs Connected: TG, Switch Connected: H) would be "HHSLY", the trouble is if you try and decrypt this with the same settings you get "HELFO" which is close but not correct. This is because the switchboard is still in operation even in decryption and for longer messages would make decryption by guessing near impossible. If you use the standard Enigma and try to decipher this (using R1: 10, R2: 11, R3: 12, Ring1: 1, Ring2: 2, Ring3: 3, Plugs Connected: TG) you will get "CELFO" which is actually better as the switchboard is not changing any characters and won't change any through the decryption process. The only way to get "Hello" back would be if you knew where the original plain text was and what it was. The issue with sending the plaintext would be that the integrity of the message would be compromised and it's plausible that an attacker could reconstruct the message from the remaining data. Another way to plausibly get the message through would be to send a binary string of all the places a letter self-enciphered, for our example this would be "10010", you can then take this and apply it to our output on the Improved Enigma and get the following decrypted message "HELLO".

The problem with having a binary representation of the encipherment is that it changes with each message and you have to send that binary representation to the receiver with the message, this could compromise some message integrity as it will start to show patterns in the ciphertext and break down the problem if both are received. Therefore, this method of decryption although it works, is slightly flawed and could cause integrity issues if both the ciphertext and binary are received by an attacker.

6.2 Improved Enigma Encipherment patterns

The improved Enigma suffers from patterns throughout the ciphertext, this was mostly found due to the encipherment of the password set. The developer manually enciphered 500 passwords in the improved Enigma (passwords with numbers not enciphered) to look for patterns, a sample of the excel sheet can be found in figure 59 although the whole sheet can be found in appendix D.

Password	Enigma Setting (Improved)	Enigma Encrypted	Frequency (%)
123456			
Password	R1: 12 R2: 15 R3: 5 Ring1: 1 Ring2: 5 Ring3: 22 Plugs Connected: None Switches R	BLVSFWRS	25.00%
12345678			
qwerty	R1: 25 R2: 1 R3: 26 Ring1: 26 Ring2: 21 Ring3: 14 Plugs Connected: None Switches H Q M	QKVJKP	16.67%
123456789			
12345			
1234			
111111			
1234567			
dragon	R1: 4 R2: 4 R3: 15 Ring1: 22 Ring2: 2 Ring3: 16 Plugs Connected: None Switches N S	CPRTSN	16.67%
123123			
baseball	R1: 15 R2: 15 R3: 9 Ring1: 20 Ring2: 13 Ring3: 24 Plugs Connected: GR VH IF KN Switches R G E S Q T B O	BUSEJAKL	62.50%
abc123			
football	R1: 7 R2: 3 R3: 23 Ring1: 1 Ring2: 2 Ring3: 8 Plugs Connected: RH Switches F N H T C A M D O K S X E	FOTTVANL	62.50%
monkey	R1: 14 R2: 5 R3: 10 Ring1: 17 Ring2: 26 Ring3: 4 Plugs Connected: MJ EV SG Switches X K J N P D W I H V C	OGNKSX	33.33%
letmein	R1: 9 R2: 4 R3: 3 Ring1: 10 Ring2: 7 Ring3: 11 Plugs Connected: KP BH Switches C R N I P W	THKHEIN	42.86%
696969			
shadow	R1: 10 R2: 2 R3: 3 Ring1: 2 Ring2: 8 Ring3: 10 Plugs Connected: None Switches J Q Y G	LXHKNA	0.00%
master	R1: 14 R2: 16 R3: 26 Ring1: 21 Ring2: 18 Ring3: 26 Plugs Connected: SK CP Switches U A B S B M R V W Q Y H E N D	MASTER	100%
666666			
qwertyuiop	R1: 7 R2: 6 R3: 19 Ring1: 23 Ring2: 2 Ring3: 24 Plugs Connected: None Switches Z H T B A N J K Y G L C D O	QPKQTYWSOH	40.00%
123321			
mustang	R1: 18 R2: 7 R3: 25 Ring1: 10 Ring2: 8 Ring3: 4 Plugs Connected: VN LW Switches T F K R G D E B M Q U L S A	MUSTAPG	85.71%
1234567890			
micheal	R1: 15 R2: 4 R3: 25 Ring1: 17 Ring2: 11 Ring3: 20 Plugs Connected: IZ TS UO AC DX Switches S X Z W G O N A P L C F R Q E	UDCLEAL	57.14%
654321			

Figure 59: Improved Enigma Encipherment

This revealed that the improved Enigma was self-enciphering 38.45% of the time, this seemed significant, so the developer ran a further test on this by encrypting a large weather report from the met office during storm Ciara. This revealed the biggest weakness of the improved Enigma; the switchboard was self-enciphering at such a high rate that it would sometimes encipher whole words almost exactly the same as the plaintext. This is a major defect for data integrity as if something important was revealed in the encipherment then the entire purpose of the message could be exposed. A key example of this is the first two words of the encipherment "This morning" enciphered as "THYSJMRBING", the whole encipherment can also be found in appendix D.

The developer also noted a pattern that emerged with double letters due to the switchboard's binary nature, a word like "Hello" has two "L"s in it meaning that the switchboard will encipher one of those "L"s as an "L" regardless. Finally, the developer found that the self-encipherments appeared to happen at the start or end of the message. A message was 33.85% likely to start with a self-encipherment, 44% likely to end with a self-encipherment and there was a 63.54% chance it would either start or end with a self-encipherment. These figures are massive and give an attacker a start when deciphering the ciphertext. These patterns are a major flaw to implementing the switchboard as it weakens the data's integrity.

6.3 Improved Enigma Crib Resistance & Modern Encryption

Unlike other areas of the improved Enigma the crib resistance is an incredible improvement on the standard and Kreigsmarine Enigma. The standard Enigma had a 100% chance of a crib being found in a message whereas the improved Enigma has just a 5% chance of a crib being found. This proves that adding the switchboard solved the cribbing issue and that a Bombe wouldn't be able to use the same cribbing method as with other Enigmas. However, it does this at a cost due to the decryption method and all the patterns introduced to the improved Enigma, thus didn't improve security but improved resistance against the Bombe machine.

Unfortunately, this means that the improved Enigma does not stand up to a modern encryption method at all - as AES has none of these weaknesses and could easily foil the Bombe machine. Perhaps if the block cipher Enigma was developed it might stand up to the modern methods we have today. If the improved Enigma was used in the modern landscape it would almost certainly be broken even if the users were using asymmetric encryption to transport the binary representation securely to each other. A modern computer could easily brute force this Enigma much like the historical Enigma that was used in the war. The key possibilities are just too small when compared against AES and Triple DES. Without the binary representation the improved Enigma behaves much more like a hash function with a lot of key customisation as the results cannot be fully decrypted without knowing where the binary bits are. However, much like MD5 I'm sure collisions could be found breaking the function entirely. With hashing such as SHA-256 (and others) modern encryption is far stronger than anything the developer could have made in this project and I invite the reader to try to break AES although I am confident this is currently out of reach for even the best current supercomputers.

6.4 Software Developed

The overall software that was developed meets all the needs stated in the requirements section. It does not store data, and it keeps a clean UI to help users and ensures all encipherments were accurate for the above evaluation. All methods developed played a key part in the developer's understanding and use of encryption. The software allows a user to use the developed software without the need for adjustment. Furthermore, the software meets objectives 1, 2 and 3 whilst the discussion in this evaluation section discusses how the developed software proves and disproves the research question.

7 Conclusion

7.1 Objectives & Research Question

The software developed meets all the objectives defined in the introduction. For objective 1 the Enigma and Enigma Kreigsmarine were developed to replicate the period Enigmas. For objective 2 the improved Enigma was developed to try and improve on the Enigma's key weakness of no letter being able to self-encipher and, for objective 3, modern encryption algorithms AES, MD5, and SHA-256 were developed.

All these objectives were then reflected upon at the evaluation stage showing that the improved Enigma has some severe weaknesses causing the effective security to be weaker but crib resistance to be much higher.

This both proves and disproves the research question as improving the Enigma did ensure security against the Bombe machine but in all other ways made the machine weaker. In comparison to the original machine the improved Enigma is much weaker due to the patterns implemented and so in that sense the machine failed. The machine cannot stand up to modern encryption at all as this machine has significantly less keys in comparison to AES and the transformation of the improved Enigma implemented weaknesses that could easily be exploited. The additional objective to create a Morse code translator was met although this added limited functionality and comparison points to the evaluation.

In conclusion the improved Enigma proved it was possible to make a machine without the core weakness of the Enigma, however, the method used made the machine much weaker than initially thought. If another developer wanted to review, the developer suggests they read the further work section found below.

7.2 Project Management

The developer ensured the project was well managed throughout having spent a lengthy period mapping out manageable sections both in the PID and throughout. The developer kept the Gantt chart from the PID up to date and used a select variety of online management tools such as Trello to keep track and increment individual tasks from start to finish. The methodology used worked very well for this project due to segmentation of the Encryption methods, and as a result development proceeded smoothly. Despite the delays to the project plan the changes were dealt with by minor rescheduling and restructuring of time and tasks where required, inclusive of scope change. The clearly defined planning as outlined in the implementation section allowed the developer to complete the entire project and additional objectives within time.

7.3 Developers Reflection

On reflection the developer learnt a lot from this project. During the literature review the developer was so immersed in Alan Turing's biography which although used as a reference book, was finished even though not all of the content was directly relevant. The developer further learnt about historical encryption in the war, how it was broken and how the Enigma and machines created to crack it paved the way for the modern computer.

Knowledge of Modern encryption was enhanced as the developer already knew that AES was the current standard, but not how it worked or what were the standards beforehand.

In undertaking this project, the developer's coding skills were improved learning a new language. This included the developer undertaking and passing the LinkedIn test on JavaScript after project development.

7.4 Further Work

Further work from this project would include, making the switch on the switchboard change between 1 and 0 at a different rate to fix the frequency. If this were changed then it is possible that the encipherments may protect the integrity of the message. Although due to the decryption it is advised that any subsequent developer should not make the switchboard and instead create an Enigma block cipher that would have been constructed if capacity allowed, as this could significantly increase security and plausibly protect it from the Bombe machine.

References

10,000 most common passwords (no date) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Wikipedia:10,000_most_common_passwords (Accessed: 15 January 2020).

Aleisa, N. (2015) 'A comparison of the 3DES and AES encryption standards', *International Journal of Security and its Applications*, 9(7), pp. 241–246. doi: 10.14257/ijsia.2015.9.7.21.

Anand, A. (2017) *Breaking Down : SHA-256 Algorithm, Medium*. Available at: https://medium.com/bugbountywriteup/breaking-down-sha-256-algorithm-2ce61d86f7a3 (Accessed: 16 December 2019).

Arias, D. (2018) *Adding Salt to Hashing: A Better Way to Store Passwords, Auth0.com*. Available at: https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/ (Accessed: 15 December 2019).

Bellare, M. *et al.* (2015) 'Introduction to Modern Cryptography', pp. 1–283. Available at: http://www-cse.ucsd.edu/users/mihir%0Ahttp://www.cs.ucdavis.edu/~rogaway%0Ahttp://www-cse.ucsd.edu/users/mihir%0Ahttp://www.cs.ucdavis.edu/~rogaway.

Caligatio (no date) 'JsSHA'. Available at: https://github.com/Caligatio/jsSHA.

Callahan, K. (2013) 'The Impact of the Allied Cryptographers on World War II: Cryptanalysis of the Japanese and German Cipher Machines', pp. 1–14. Available at: https://pdfs.semanticscholar.org/c7cf/0c41932d61457dd943dc4dffca2c8bb92e95.pdf.

Carter, F. (2008) 'The Turing Bombe', *The Rutherford Journal*. Available at: http://www.rutherfordjournal.org/article030108.html.

Computerphile (2012) *Feistel Cipher, SpringerReference*. doi: 10.1007/springerreference_262.

Computerphile (2019) *AES Explained (Advanced Encryption Standard) - Computerphile*. Available at: https://www.youtube.com/watch?v=O4xNJsjtN6E (Accessed: 22 September 2020).

Computing, 101 (2019) *Enigma Crib Analysis*. Available at: https://www.101computing.net/enigma-crib-analysis/ (Accessed: 1 February 2020).

Crypto Museum (2009) *Enigma Wiring*. Available at: https://www.cryptomuseum.com/crypto/enigma/wiring.htm (Accessed: 19 December 2019).

'CryptoJS' (no date). Available at: https://cryptojs.gitbook.io/docs/.

Daemen, J. and Rijmen, V. (2002) 'The Design of Rijndael', *New York*, p. 255. doi: 10.1007/978-3-662-04722-4.

Daniel (2018) *Cryptographic Hash Functions Explained: A Beginner's Guide, komodo*. Available at: https://komodoplatform.com/cryptographic-hash-function/ (Accessed: 9 December 2019).

Ellis, C. (2009) 'Exploring the Enigma The birth of an enigma How the Enigma machine worked', (March 2005), pp. 1–9.

Gabbasov, D. (2015) 'Breaking the Enigma', pp. 1–9. Available at: https://courses.cs.ut.ee/MTAT.07.022/2015_spring/uploads/Main/dmitri-report-s15.pdf.

Hallenberg, F. (2015) 'A Few Words on the Enigma Machine and the Bombe More details about rotors and drums', pp. 1–13.

HODGES, A. (2019) *Alan Turing: The Enigma, Alan Turing: The Enigma*. doi: 10.2307/j.ctvc77913.

J, G. (2017) *How does RSA work?, Hackernoon*. Available at: https://hackernoon.com/how-does-rsa-work-f44918df914b (Accessed: 19 February 2020).

Korpal, G. (no date) ‘Enigma Cryptanalysis’.

Marc, S. E. B. (2017) *Shattered*. Available at: https://shattered.io/ (Accessed: 15 February 2020).

Meyers, M. and Jernigan, S. (2018) *CompTIA Security + Certification Guide Second Edition*.

Morkel, T. (no date) ‘Encryption Techniques : a Timeline’, (2054024), pp. 1–14.

Powtac (2009) *fastest MD5 Implementation in JavaScript*, *StackOverflow*. Available at: https://stackoverflow.com/questions/1655769/fastest-md5-implementation-in-javascript (Accessed: 20 December 2019).

Ranger, S. (2019) *Facebook: We stored hundreds of millions of your passwords in plain text*, *ZDNet*. Available at: https://www.zdnet.com/article/facebook-we-stored-hundreds-of-millions-of-passwords-in-plain-text/ (Accessed: 10 October 2019).

Rouse, M. (no date) *asymmetric cryptography (public key cryptography)*, *Search Security*. Available at: https://searchsecurity.techtarget.com/definition/asymmetric-cryptography (Accessed: 21 January 2020).

Singh, G. and Supriya, S. (2013) ‘A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security’, *International Journal of Computer Applications*, 67(19), pp. 33–38. doi: 10.5120/11507-7224.

Sobti, R. and Geetha, G. (2012) ‘Cryptographic Hash functions - a review’, *International Journal of Computer Science Issues*, (March 2012).

SSL2BUY (2015) *Symmetric vs. Asymmetric Encryption – What are differences?* Available at: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences (Accessed: 25 January 2020).

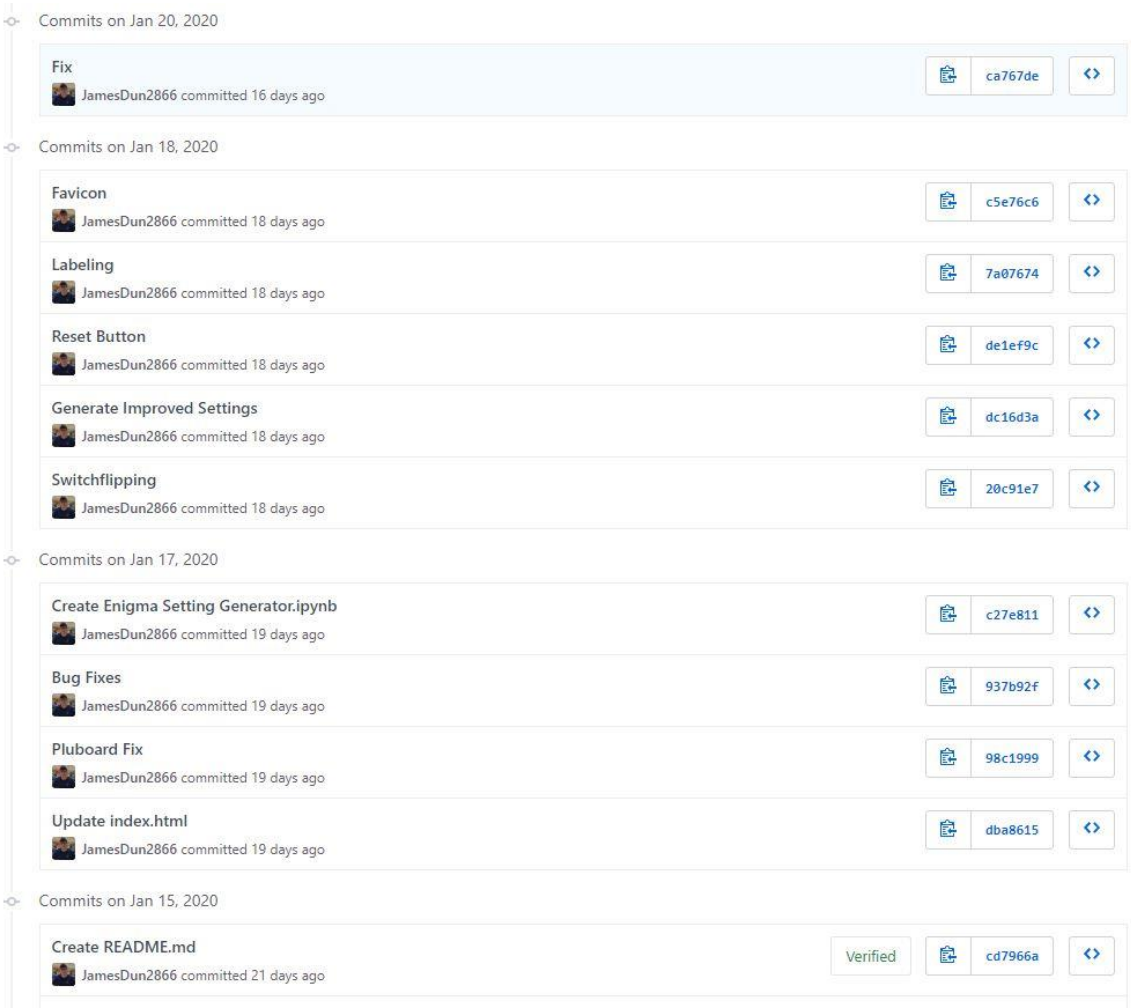
Syncthetic (2017) ‘MorseCode’. Available at: https://github.com/Syncthetic/MorseCode.

Turing, D. (2018) *The Bombe Breakthrough*. Pavilion Books Company.

Ward, M. and Singh, S. (2001) ‘The Code Book: The Secret History of Codes and Codebreaking’, *The Mathematical Gazette*, p. 351. doi: 10.2307/3622055.

Appendix A – Project Management Screenshots

Commit History



Create Salt Generator.ipynb
JamesDun2866 committed 21 days ago — 1931017

Commits on Dec 27, 2019

Create Mid Project.pptx
JamesDun2866 committed on 27 Dec 2019 — b39f4b0

Morse Code
JamesDun2866 committed on 27 Dec 2019 — f98fcbd

Improved Enigma Added
JamesDun2866 committed on 27 Dec 2019 — aec0c02

Removed usless files
JamesDun2866 committed on 27 Dec 2019 — c69699f

Code commenting
JamesDun2866 committed on 27 Dec 2019 — 6b3285b

Commenting
JamesDun2866 committed on 27 Dec 2019 — 45e3cc3

Commenting
JamesDun2866 committed on 27 Dec 2019 — 022d382

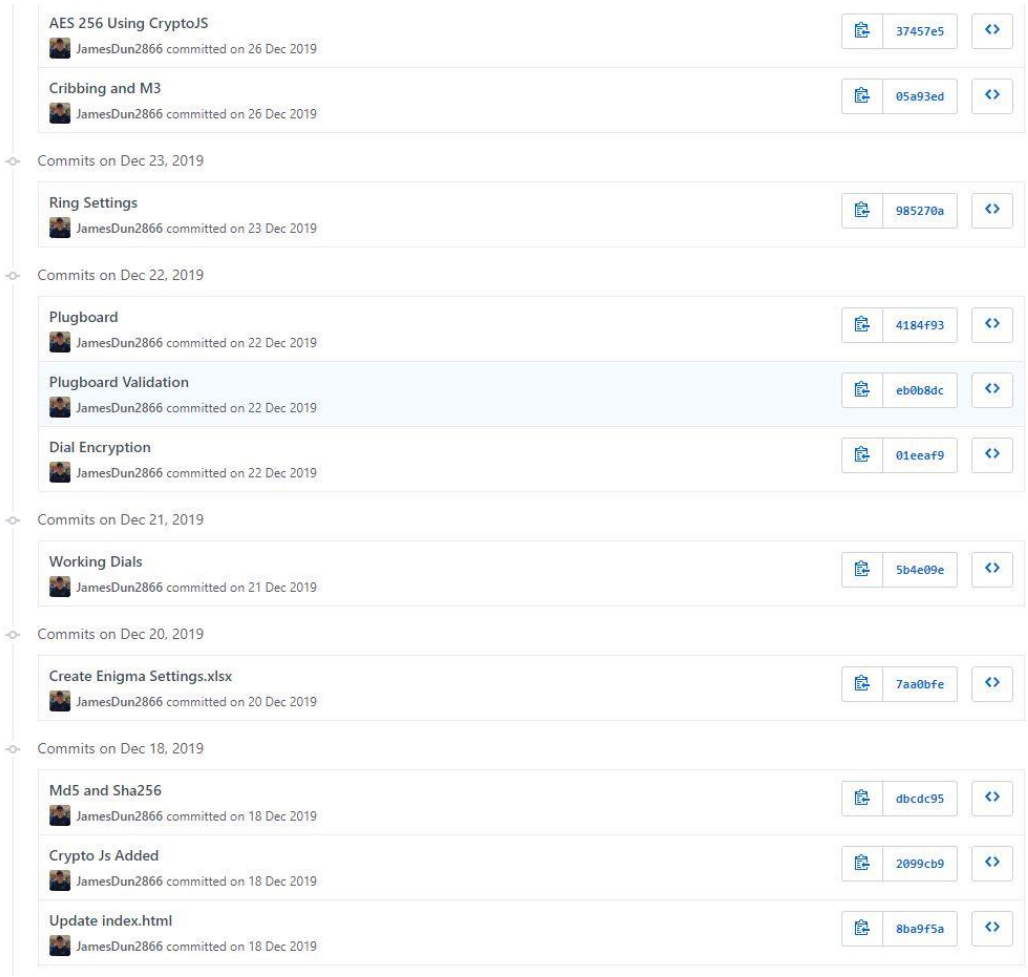
Code commenting
JamesDun2866 committed on 27 Dec 2019 — a65ba16

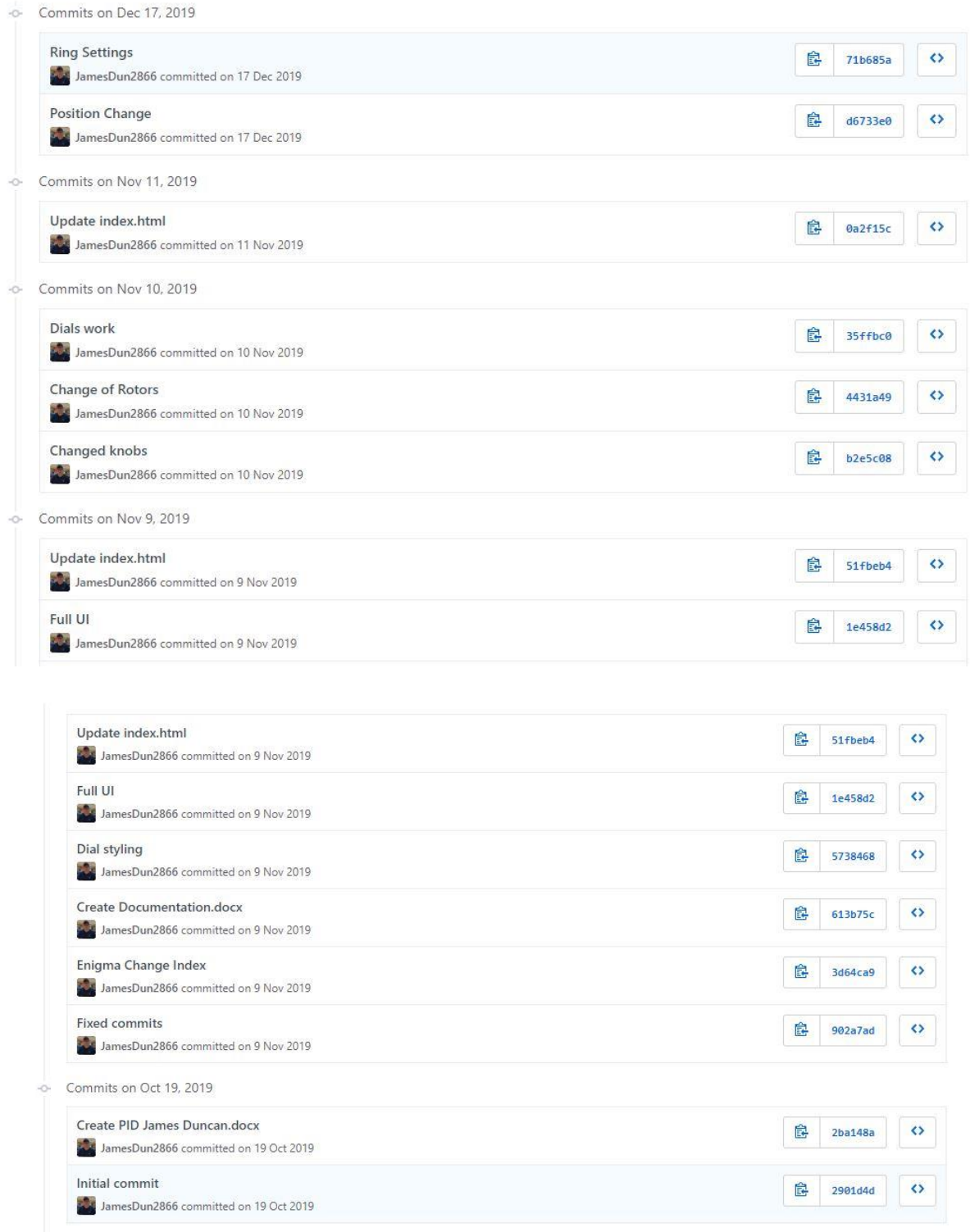
Code Commenting
JamesDun2866 committed on 27 Dec 2019 — 659608f

Code Commenting
JamesDun2866 committed on 27 Dec 2019 — 892ae1d

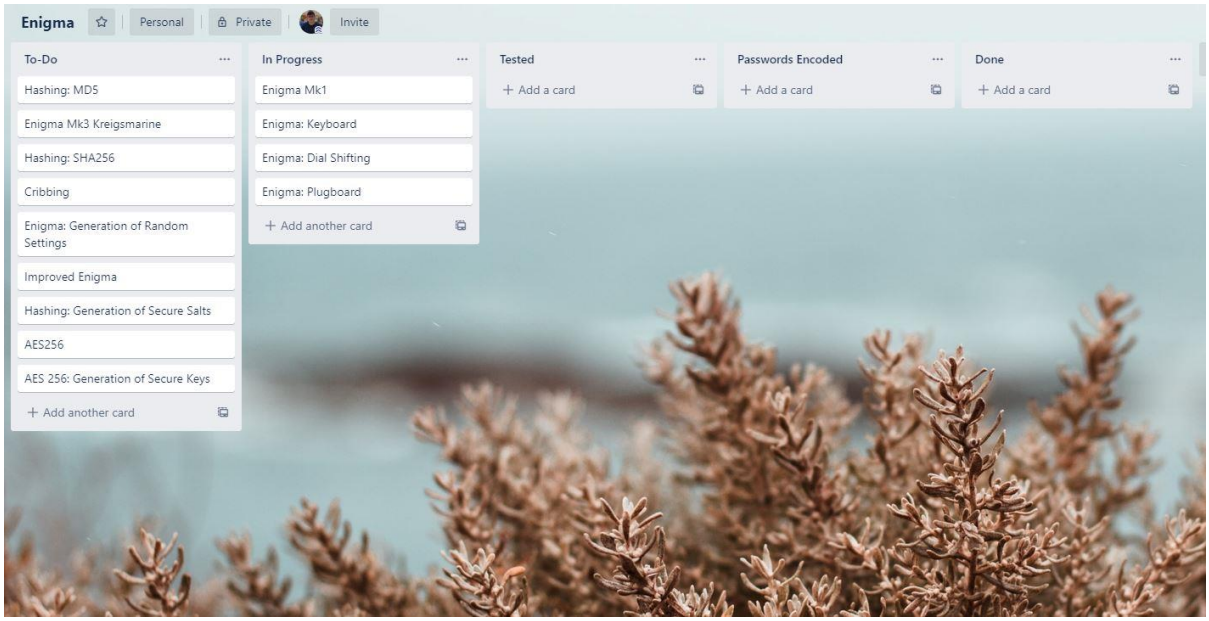
Commits on Dec 26, 2019

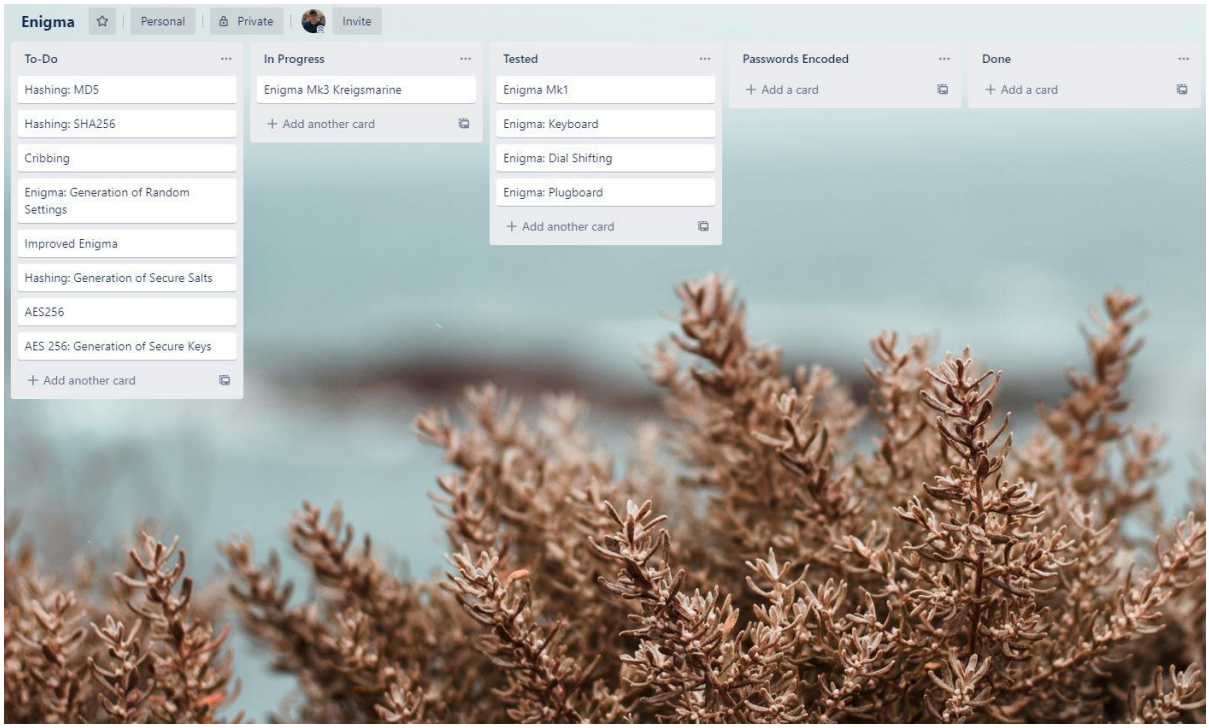
AES 256 Using CryptoJS
JamesDun2866 committed on 26 Dec 2019 — 37457e5

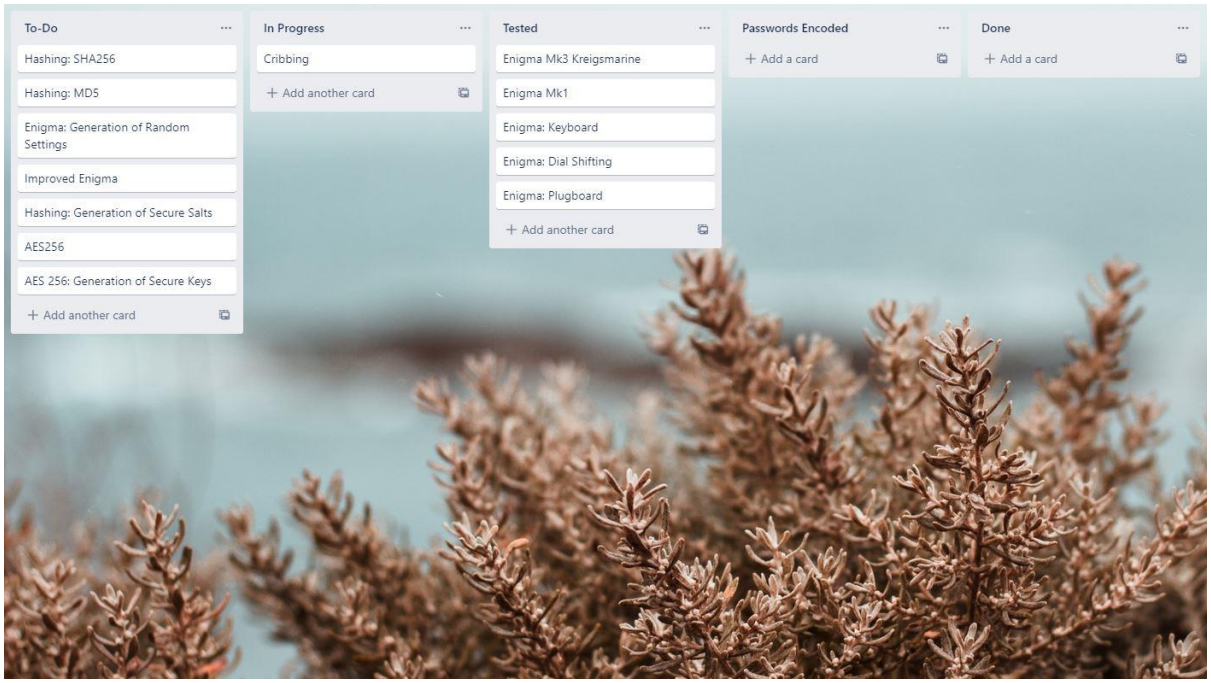

AES 256 Using CryptoJS
JamesDun2866 committed on 26 Dec 2019
37457e5
Cribbing and M3
JamesDun2866 committed on 26 Dec 2019
05a93ed
Commits on Dec 23, 2019
Ring Settings
JamesDun2866 committed on 23 Dec 2019
985270a
Commits on Dec 22, 2019
Plugboard
JamesDun2866 committed on 22 Dec 2019
4184f93
Plugboard Validation
JamesDun2866 committed on 22 Dec 2019
eb0b8dc
Dial Encryption
JamesDun2866 committed on 22 Dec 2019
01eeaf9
Commits on Dec 21, 2019
Working Dials
JamesDun2866 committed on 21 Dec 2019
5b4e09e
Commits on Dec 20, 2019
Create Enigma Settings.xlsx
JamesDun2866 committed on 20 Dec 2019
7aa0bfe
Commits on Dec 18, 2019
Md5 and Sha256
JamesDun2866 committed on 18 Dec 2019
dbcdc95
Crypto Js Added
JamesDun2866 committed on 18 Dec 2019
2099cb9
Update index.html
JamesDun2866 committed on 18 Dec 2019
8ba9f5a

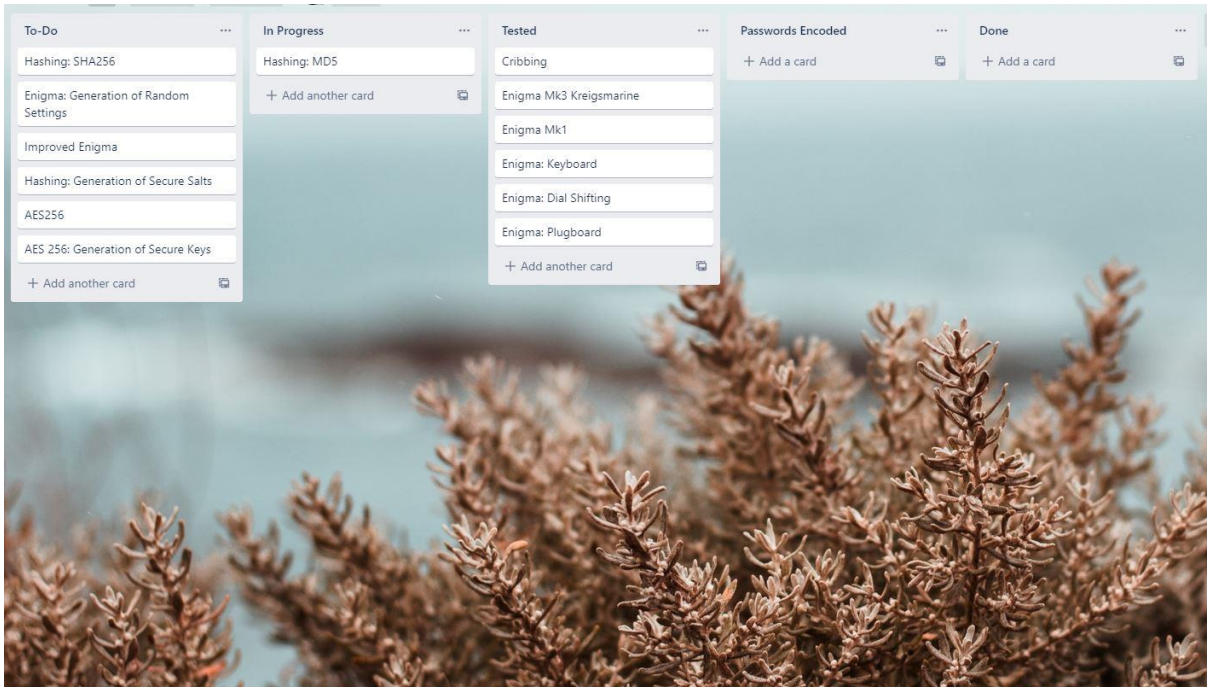


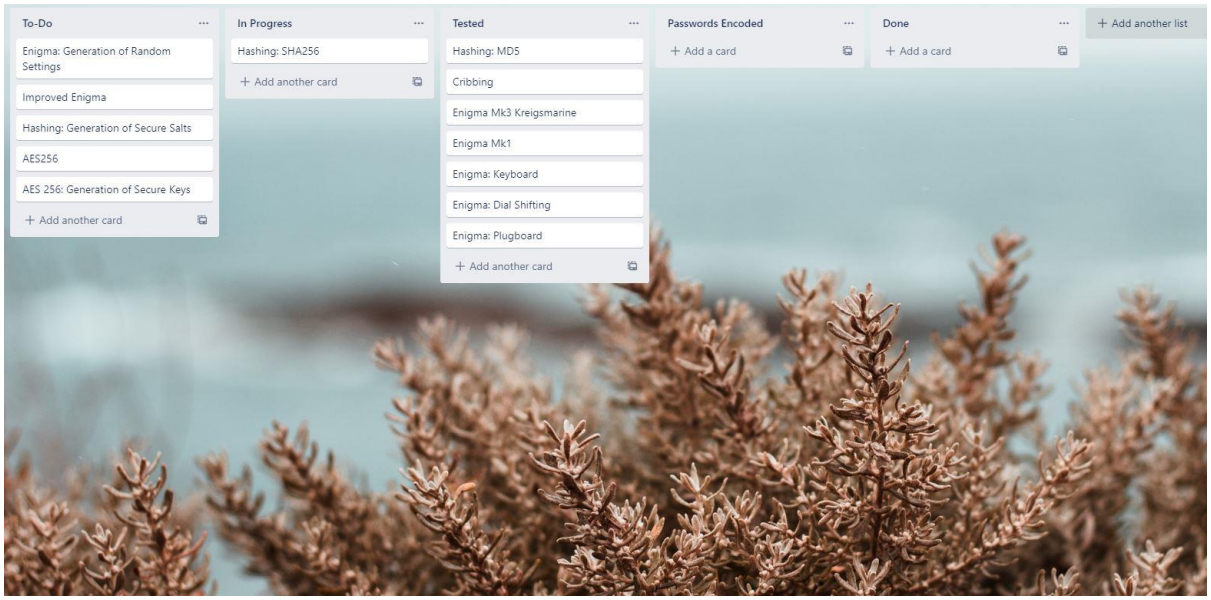
Trello Board

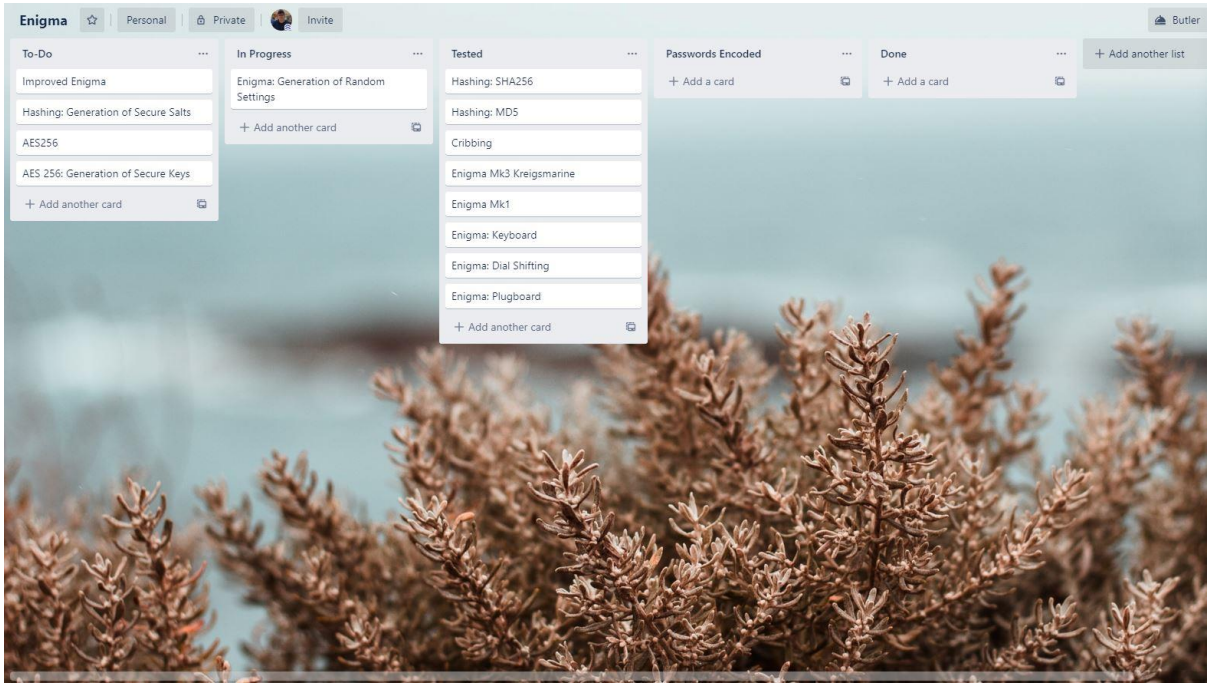

Enigma
Personal
Private
Invite
To-Do
Hashing: MD5
Enigma Mk3 Kreigsmarine
Hashing: SHA256
Cribbing
Enigma: Generation of Random Settings
Improved Enigma
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma: Plugboard
Add another card
Tested
Add a card
Passwords Encoded
Add a card
Done
Add a card

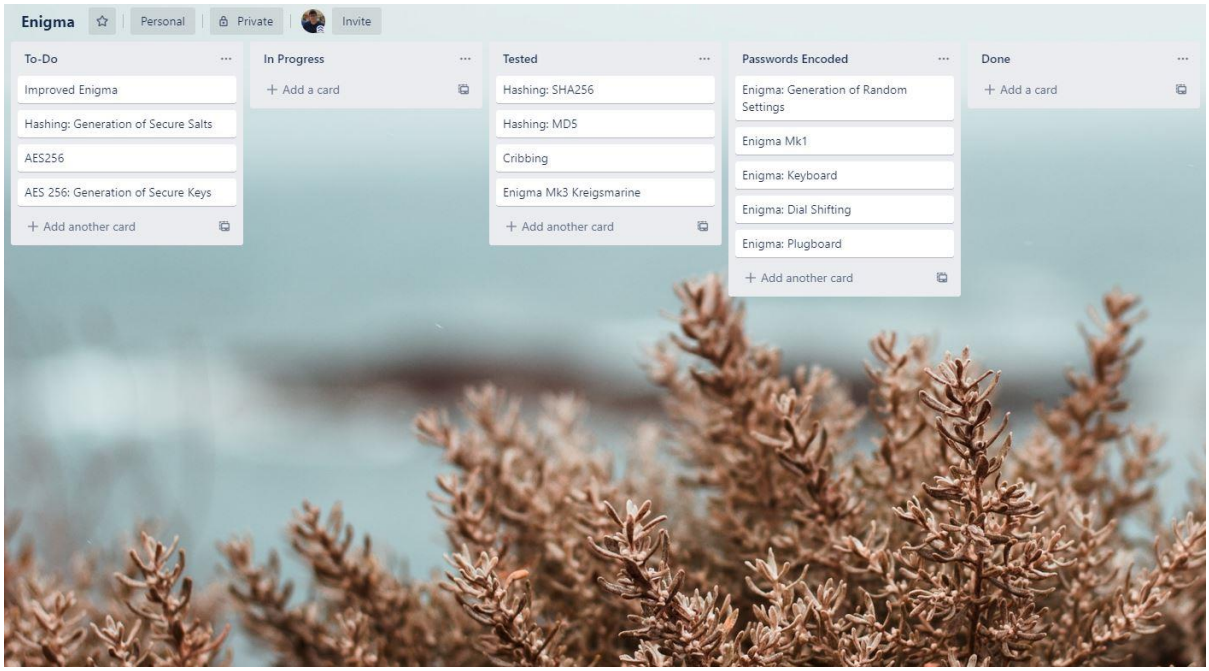

Enigma
Personal
Private
Invite
To-Do
Hashing: MD5
Hashing: SHA256
Cribbing
Enigma: Generation of Random Settings
Improved Enigma
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Enigma Mk3 Kreigsmarine
Add another card
Tested
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma: Plugboard
Add another card
Passwords Encoded
Add a card
Done
Add a card

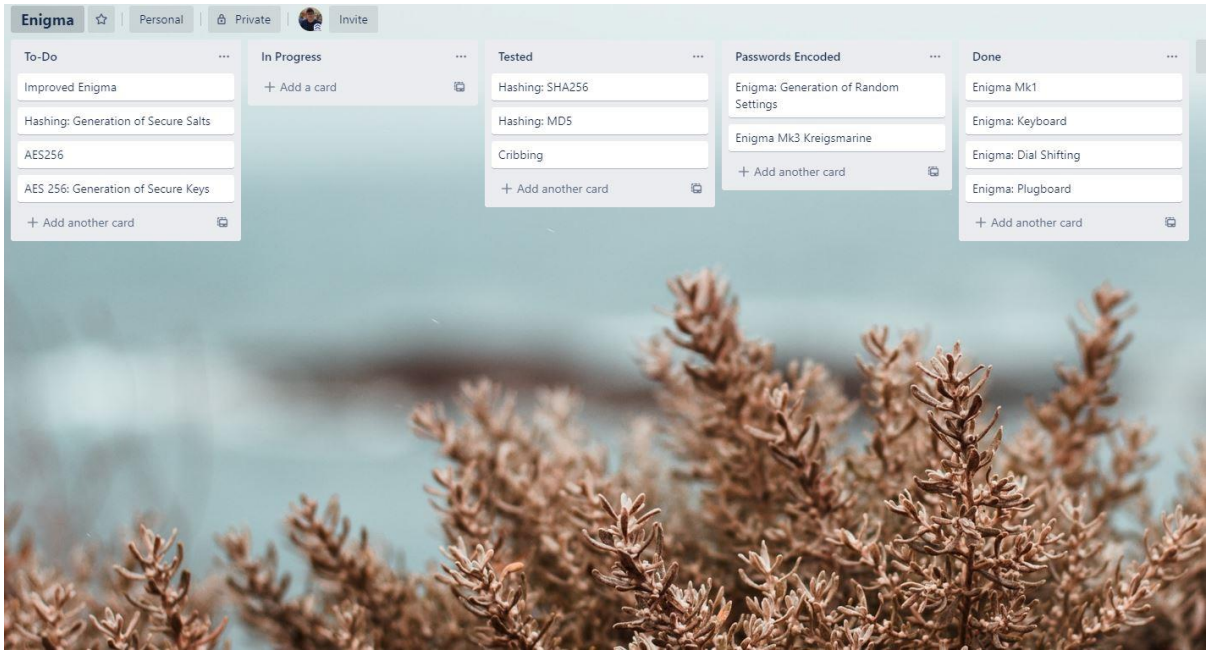

To-Do
Hashing: SHA256
Hashing: MD5
Enigma: Generation of Random Settings
Improved Enigma
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Cribbing
Add another card
Tested
Enigma Mk3 Kreigsmarine
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma: Plugboard
Add another card
Passwords Encoded
Add a card
Done
Add a card

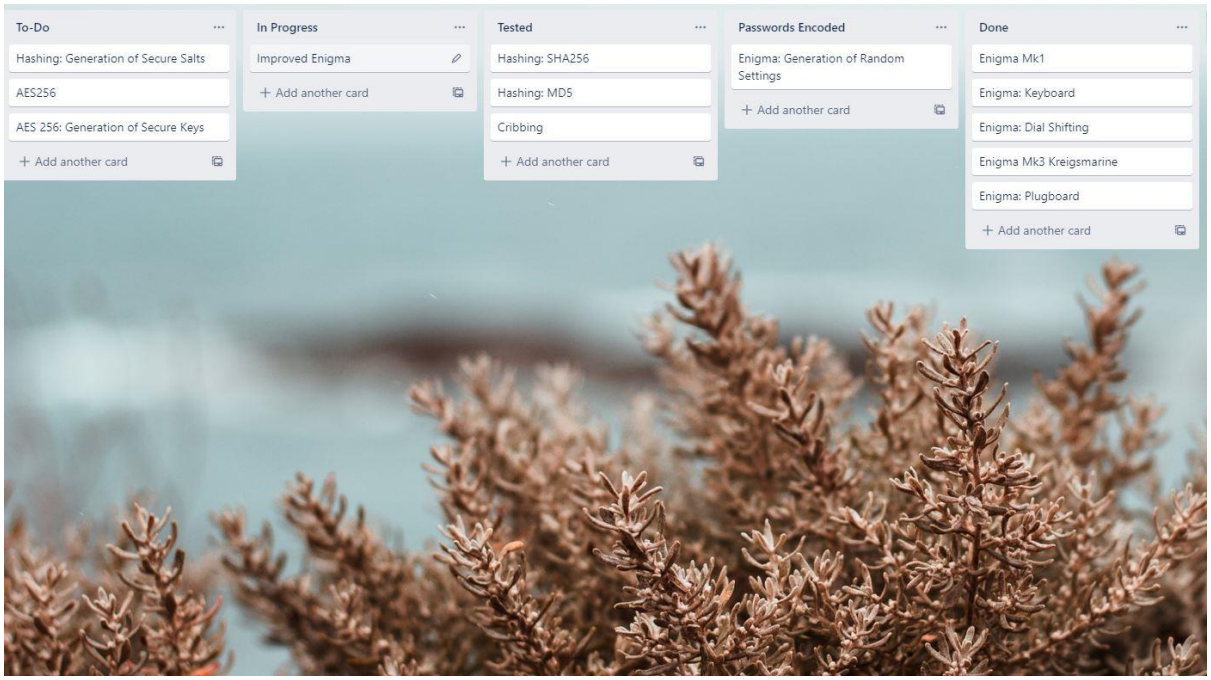

To-Do
Hashing: SHA256
Enigma: Generation of Random Settings
Improved Enigma
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Hashing: MD5
Add another card
Tested
Cribbing
Enigma Mk3 Kreigsmarine
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma: Plugboard
Add another card
Passwords Encoded
Add a card
Done
Add a card

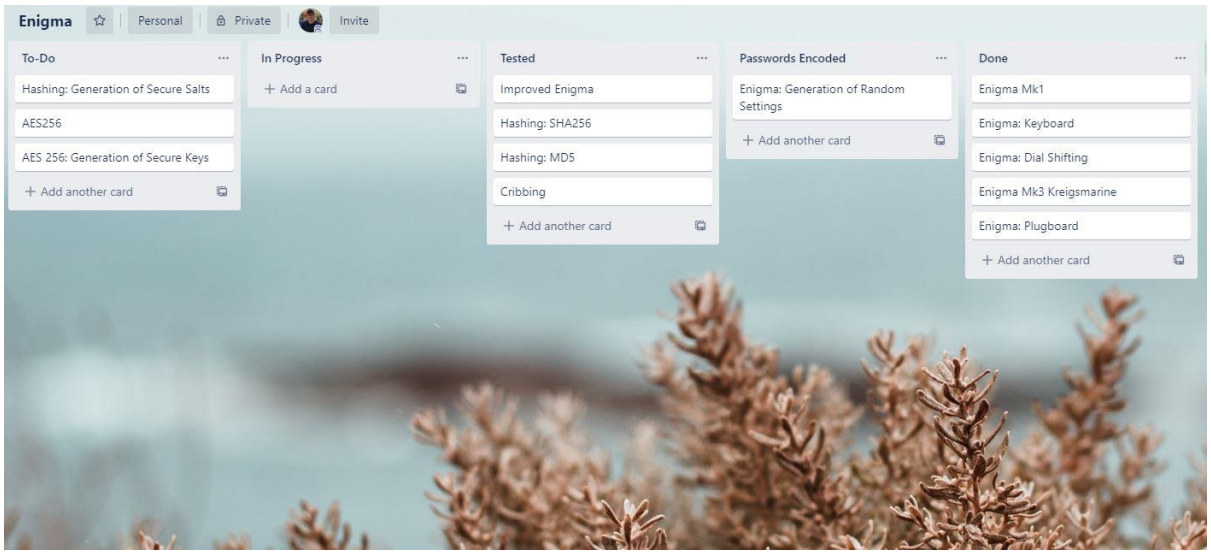

To-Do
Enigma: Generation of Random Settings
Improved Enigma
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Hashing: SHA256
Add another card
Tested
Hashing: MD5
Cribbing
Enigma Mk3 Kreigsmarine
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma: Plugboard
Add another card
Passwords Encoded
Add a card
Done
Add a card
Add another list

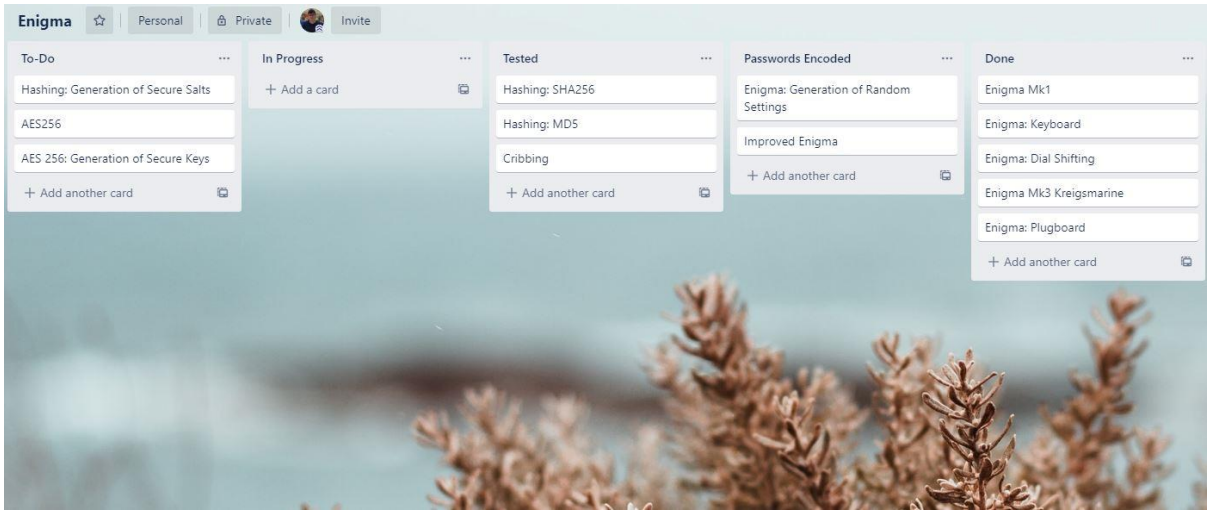

Enigma
Personal
Private
Invite
Butler
To-Do
Improved Enigma
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Enigma: Generation of Random Settings
Add another card
Tested
Hashing: SHA256
Hashing: MD5
Cribbing
Enigma Mk3 Kreigsmarine
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma: Plugboard
Add another card
Passwords Encoded
Add a card
Done
Add a card
Add another list

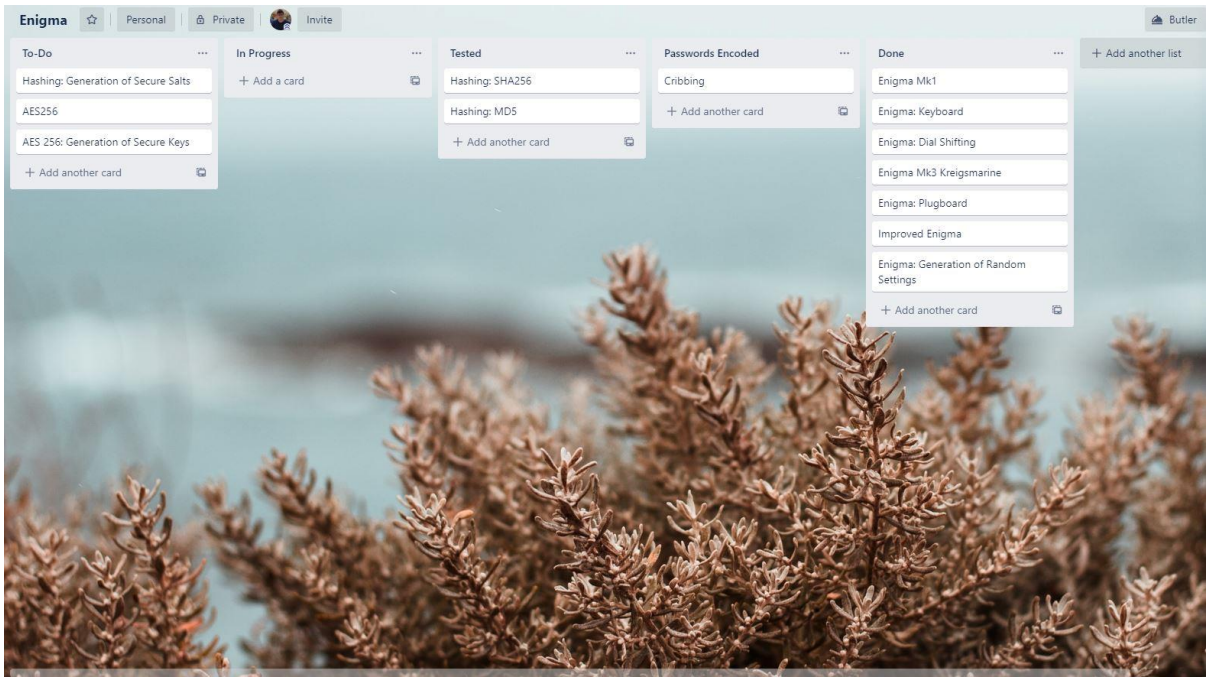

Enigma
Personal
Private
Invite
To-Do
Improved Enigma
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Add a card
Tested
Hashing: SHA256
Hashing: MD5
Cribbing
Enigma Mk3 Kreigsmarine
Add another card
Passwords Encoded
Enigma: Generation of Random Settings
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma: Plugboard
Add another card
Done
Add a card

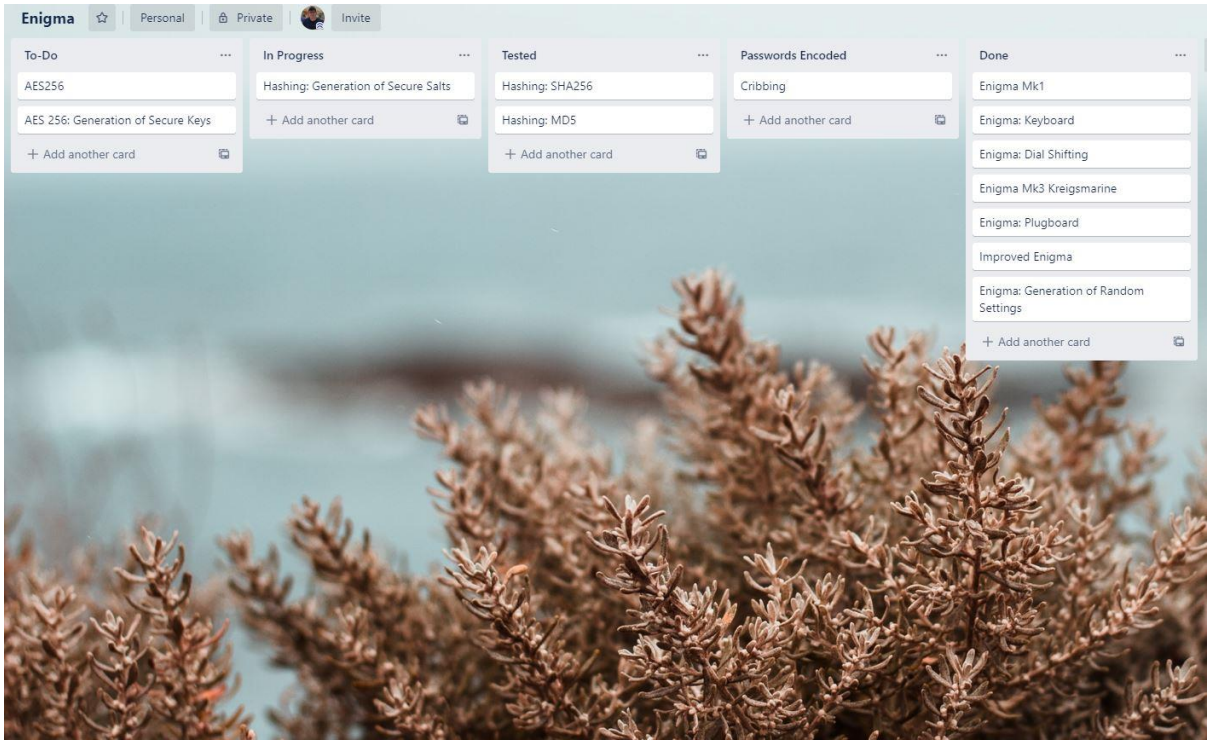

Enigma
Personal
Private
Invite
To-Do
Improved Enigma
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Add a card
Tested
Hashing: SHA256
Hashing: MD5
Cribbing
Add another card
Passwords Encoded
Enigma: Generation of Random Settings
Enigma Mk3 Kreigsmarine
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma: Plugboard
Add another card

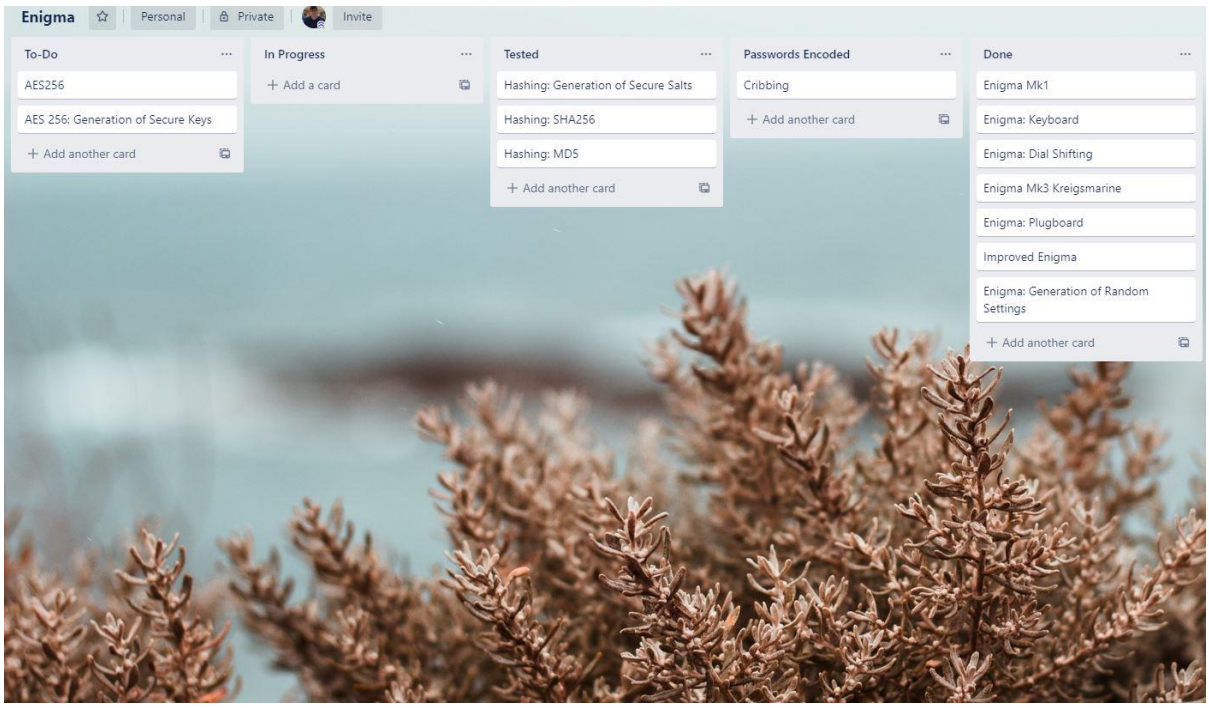

To-Do
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Improved Enigma
Add another card
Tested
Hashing: SHA256
Hashing: MD5
Cribbing
Add another card
Passwords Encoded
Enigma: Generation of Random Settings
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Add another card

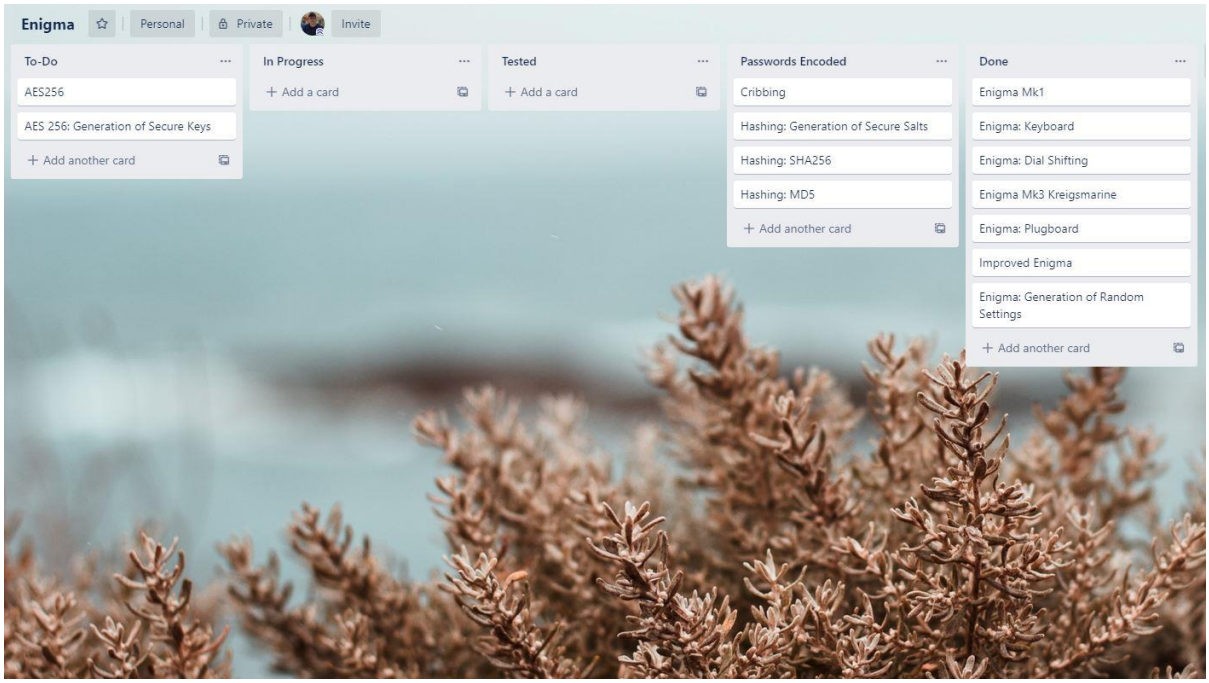

Enigma
Personal
Private
Invite
To-Do
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Add a card
Tested
Improved Enigma
Hashing: SHA256
Hashing: MD5
Cribbing
Add another card
Passwords Encoded
Enigma: Generation of Random Settings
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Add another card

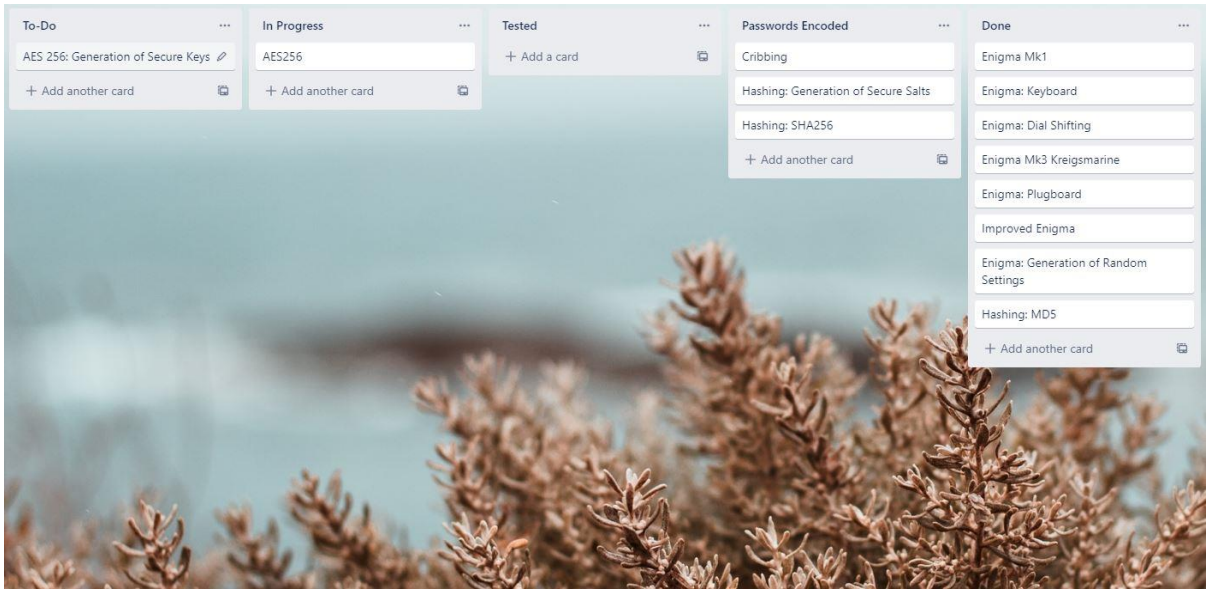

Enigma
Personal
Private
Invite
To-Do
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Add a card
Tested
Hashing: SHA256
Hashing: MD5
Cribbing
Add another card
Passwords Encoded
Enigma: Generation of Random Settings
Improved Enigma
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Add another card

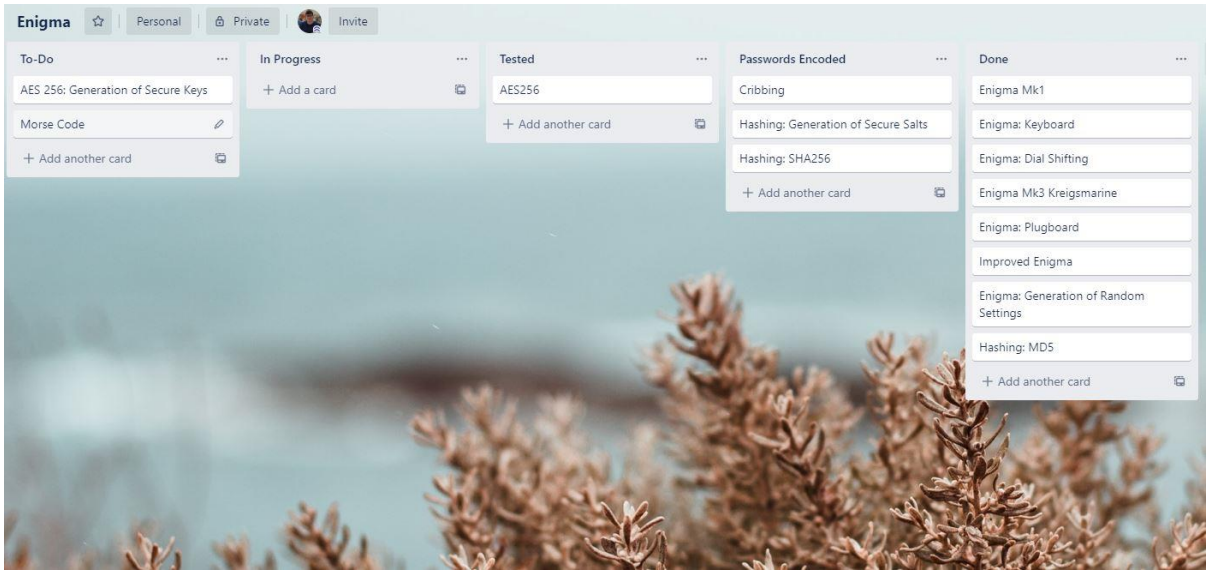

Enigma
Personal
Private
Invite
Butler
To-Do
Hashing: Generation of Secure Salts
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Add a card
Tested
Hashing: SHA256
Hashing: MD5
Add another card
Passwords Encoded
Cribbing
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Add another card
Add another list

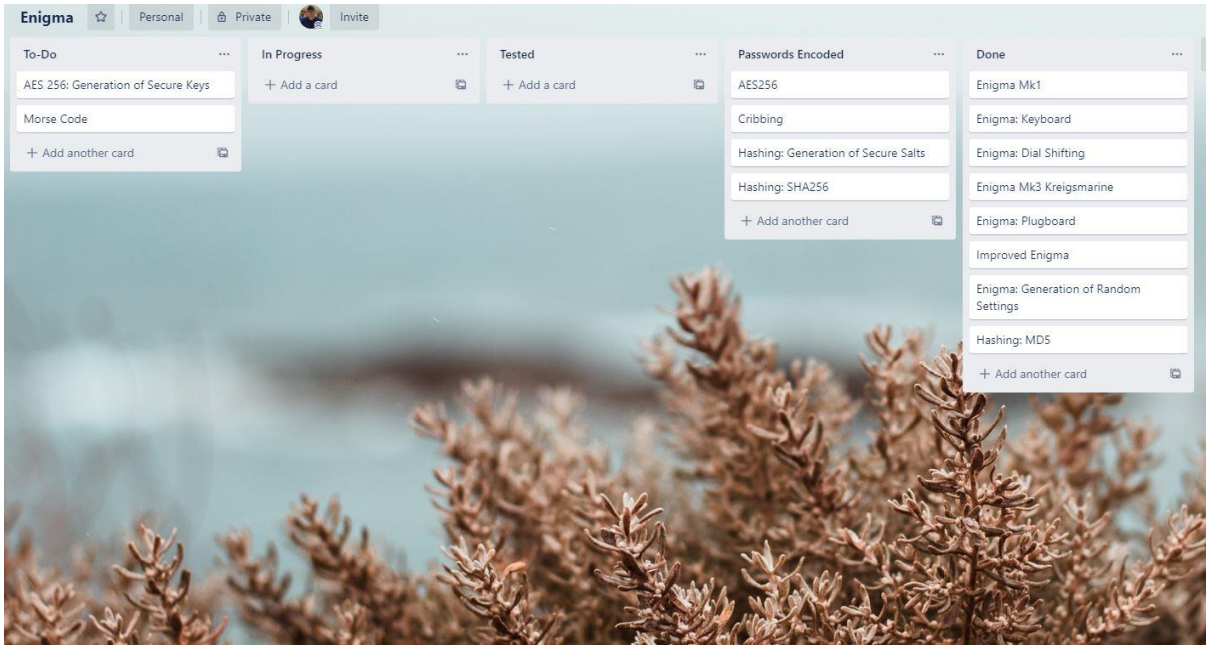

Enigma
Personal
Private
Invite
To-Do
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Hashing: Generation of Secure Salts
Add another card
Tested
Hashing: SHA256
Hashing: MD5
Add another card
Passwords Encoded
Cribbing
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Add another card

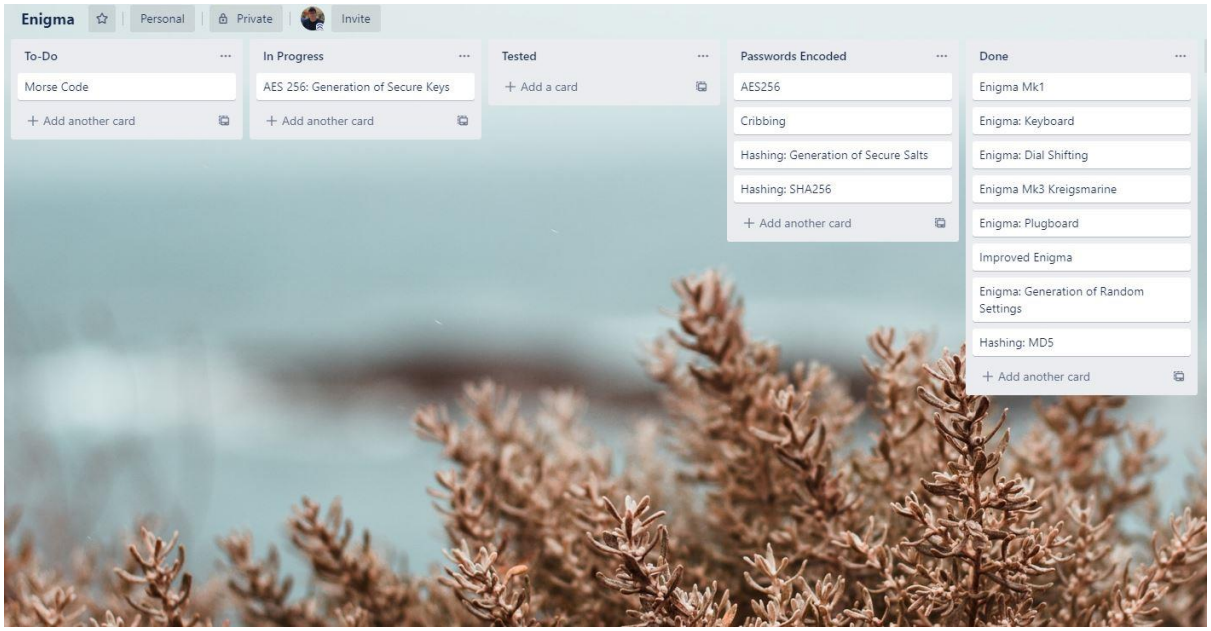

Enigma
Personal
Private
Invite
To-Do
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Add a card
Tested
Hashing: Generation of Secure Salts
Hashing: SHA256
Hashing: MD5
Add another card
Passwords Encoded
Cribbing
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Add another card

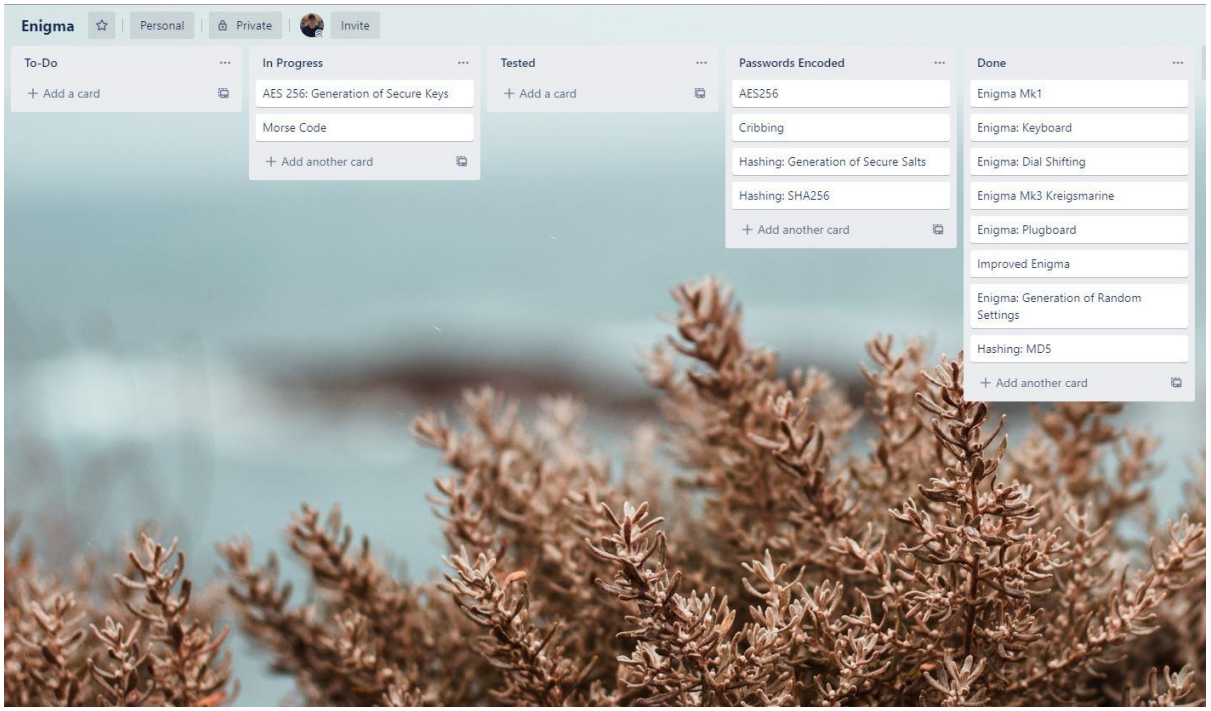

Enigma
Personal
Private
Invite
To-Do
AES256
AES 256: Generation of Secure Keys
Add another card
In Progress
Add a card
Tested
Add a card
Passwords Encoded
Cribbing
Hashing: Generation of Secure Salts
Hashing: SHA256
Hashing: MD5
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Add another card

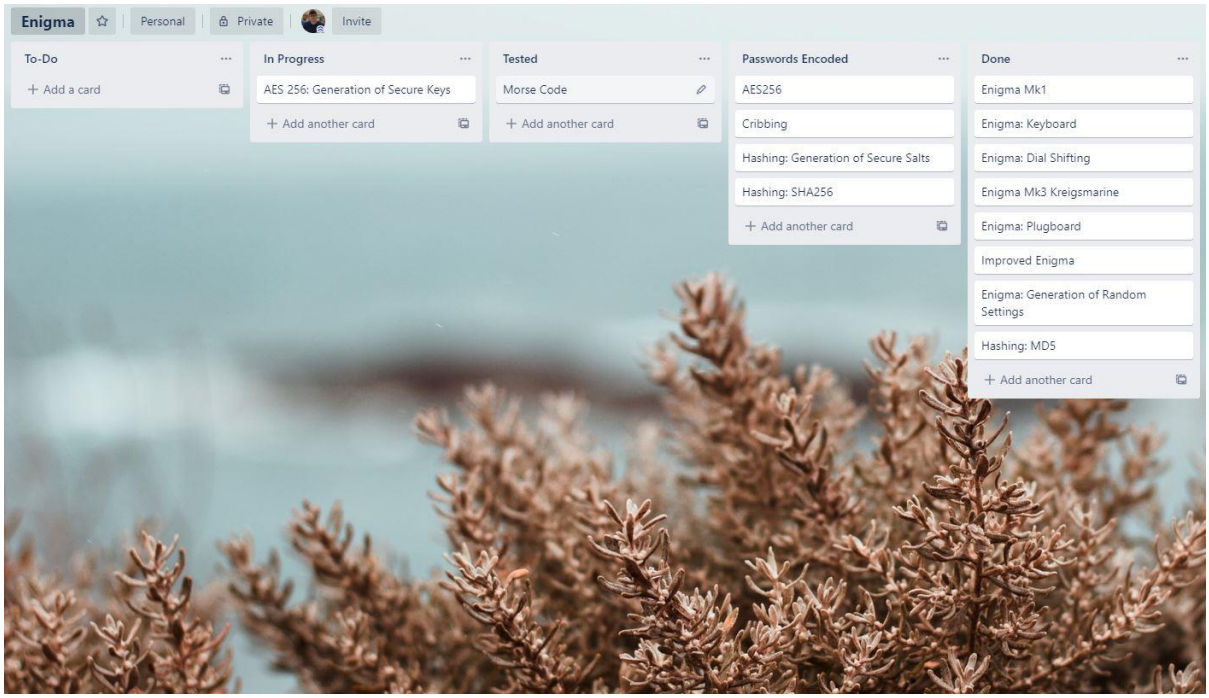

To-Do
AES 256: Generation of Secure Keys
Add another card
In Progress
AES256
Add another card
Tested
Add a card
Passwords Encoded
Cribbing
Hashing: Generation of Secure Salts
Hashing: SHA256
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Hashing: MD5
Add another card

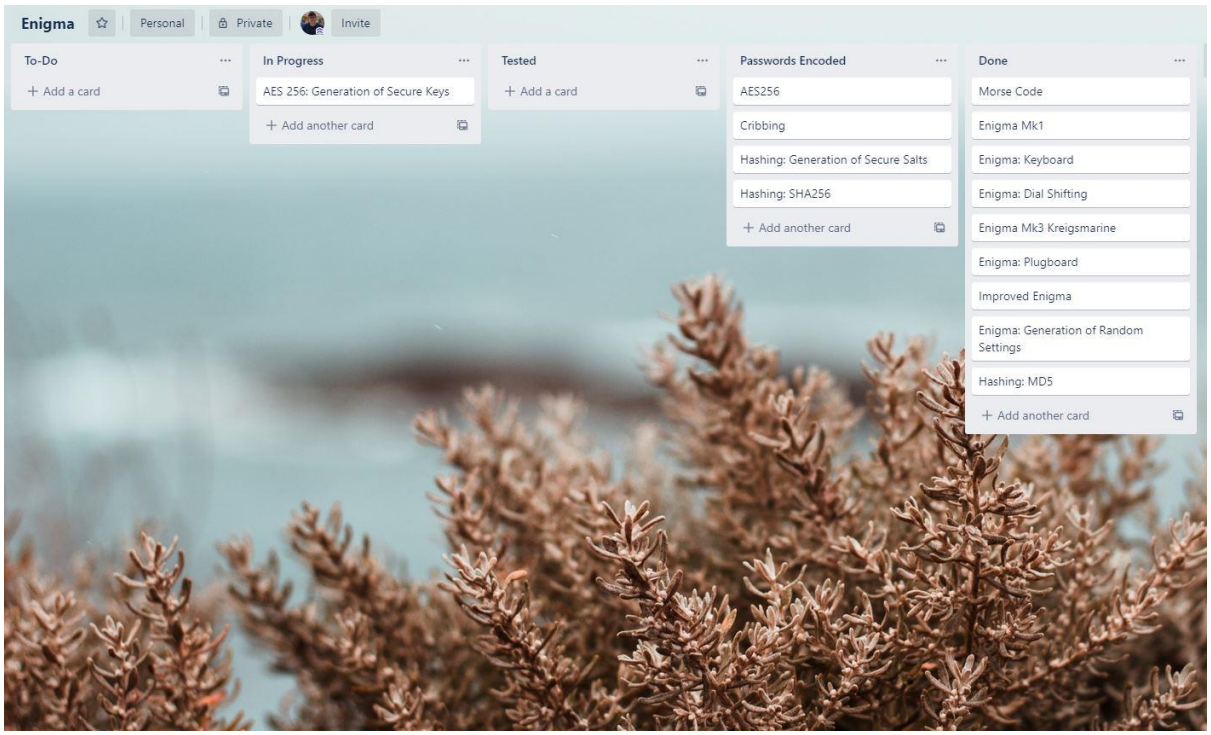

Enigma
Personal
Private
Invite
To-Do
AES 256: Generation of Secure Keys
Morse Code
Add another card
In Progress
Add a card
Tested
AES256
Add another card
Passwords Encoded
Cribbing
Hashing: Generation of Secure Salts
Hashing: SHA256
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Hashing: MD5
Add another card

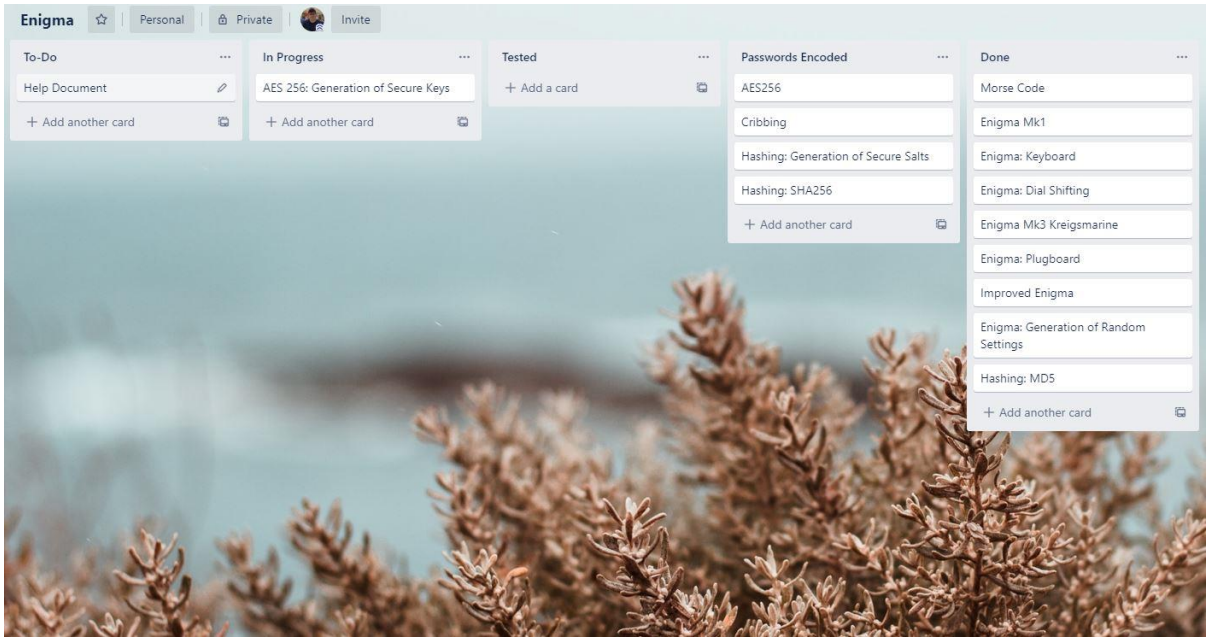

Enigma
Personal
Private
Invite
To-Do
AES 256: Generation of Secure Keys
Morse Code
Add another card
In Progress
Add a card
Tested
Add a card
Passwords Encoded
AES256
Cribbing
Hashing: Generation of Secure Salts
Hashing: SHA256
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Hashing: MD5
Add another card


Enigma
Personal
Private
Invite
To-Do
Morse Code
Add another card
In Progress
AES 256: Generation of Secure Keys
Add another card
Tested
Add a card
Passwords Encoded
AES256
Cribbing
Hashing: Generation of Secure Salts
Hashing: SHA256
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Hashing: MD5
Add another card

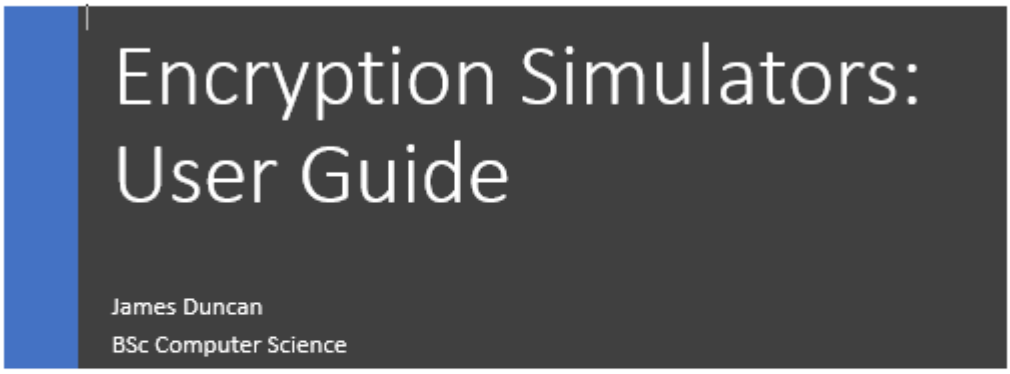

Enigma
Personal
Private
Invite
To-Do
Add a card
In Progress
AES 256: Generation of Secure Keys
Morse Code
Add another card
Tested
Add a card
Passwords Encoded
AES256
Cribbing
Hashing: Generation of Secure Salts
Hashing: SHA256
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Hashing: MD5
Add another card


Enigma
Personal
Private
Invite
To-Do
Add a card
In Progress
AES 256: Generation of Secure Keys
Add another card
Tested
Morse Code
Add another card
Passwords Encoded
AES256
Cribbing
Hashing: Generation of Secure Salts
Hashing: SHA256
Add another card
Done
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Hashing: MD5
Add another card


Enigma
Personal
Private
Invite
To-Do
Add a card
In Progress
AES 256: Generation of Secure Keys
Add another card
Tested
Add a card
Passwords Encoded
AES256
Cribbing
Hashing: Generation of Secure Salts
Hashing: SHA256
Add another card
Done
Morse Code
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Hashing: MD5
Add another card


Enigma
Personal
Private
Invite
To-Do
Help Document
Add another card
In Progress
AES 256: Generation of Secure Keys
Add another card
Tested
Add a card
Passwords Encoded
AES256
Cribbing
Hashing: Generation of Secure Salts
Hashing: SHA256
Add another card
Done
Morse Code
Enigma Mk1
Enigma: Keyboard
Enigma: Dial Shifting
Enigma Mk3 Kreigsmarine
Enigma: Plugboard
Improved Enigma
Enigma: Generation of Random Settings
Hashing: MD5
Add another card

Appendix B – User Manual



Contents

Introduction

What does the software do?

This software was developed to emulate multiple encryption methods, principally the Enigma machine M1 and Kriegsmarine variant, as they were a core element of the research of the full project. Other methods developed include AES (Advanced Encryption Standard) which is the modern symmetrical encryption standard and Hashing Methods (SHA-256 and MD5). An improved Enigma was also developed to improve on the main weakness of the Enigma machines to ensure security against the Bombe machine. As a result, a cribbing device was created to test the Improved Enigma's strength.

Why the software was developed

This software was primarily developed to prove or disprove the research question stated below, this is discussed in the full report in detail.

> *"Does making improvements to serious security flaws in the Enigma Machine using modern computers ensure security against the bombe machine, and how does this Enigma stand up to Modern Encryption methods? "*

It is worth noting that there is also an educational perspective as it helps users understand the steps encryption takes and how modern encryption is used. It further may help businesses understand how to implement modern encryption techniques.

How to use this guide

This guide should be used to supplement the software developed, however it is by no means the full documentation of the software for that I encourage the user to read the full report alongside this software. This guide should be used to help the user understand key areas of the software so the user can use it and understand how to set up each encryption method.

Prerequisites

To run this program the user must use an up to date web browser to get the best experience, the developer suggests using a browser such as Google Chrome, Mozilla Firefox or Microsoft Edge (Chromium).

Security

It is worth noting that these methods should not be used for personal data as multiple methods specifically the Enigma M1, Kreigsmarine and MD5 Hashing are considered broken. Other methods such as the improved Enigma are also on this list as even through the developer attempted to improve the Enigma as a result the machine suffered integrity issues. I also ask the user to not use AES or SHA-256 as primary methods as there may be implementation errors that the developer was not aware of. No data is stored on the system therefore the system abides by all data protection laws such as the GDPR.

Enigma

What Is an Enigma Machine?

The first thing the user will see when they launch the software is the Enigma Machine, the German World War 2 Encryption device. This is shown in Figure 1 below.

Figure 1: Enigma Machine

The Enigma was adopted by the German Military after World War 1 due to their previous Cipher being broken by the British. It works by having an electrical current flow through the three or four rotors which all have interconnected wires which transform the inputted letter into other letters. The first rotor rotates one position every time a letter is pressed, the second rotates when the first rotor hits its notch position (i.e. Trigger in the full rotation that pushes the next rotor forward). The first rotor rotates when the second hits the notch and so on. The current then hits the reflector which returns the current back through all of the rotors to light up a lamp on the lamp board. Many settings affect the rotors such as the ring settings that shift the notch position of that rotor forward and the starting positions that pre-shift the rotor forward. The final element of the setting is the plugboard that swaps a letter before the current flows through the rotor, for example if "A" is connected to "B" then when the user types "A" it will swap to "B" and go through the rotors.

Rotor Order

To change the rotor order of the Developed Enigma the user must enter the code to do so, this is because the developer did not program this into the UI. Without changing the order, the default is

rotors 1,2 and 3 and in the case of the Kreigsmarine including Rotor 4 with reflector designator B. If the user wishes to change these rotors to the historically accurate rotors used these can be found on the crypto museums website (https://www.cryptomuseum.com/crypto/enigma/wiring.htm), and the code requiring to be changed can be found in Figure 2. The user may also make up their own rotor combinations.

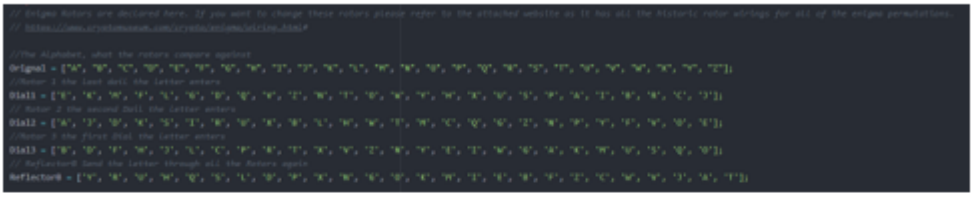


Figure 2: Rotor Orders

Rotor Orientation

The rotors can have pre-set settings which shift their own individual alphabets, this is the rotor orientation which can be set by inputting a number between 1 and 26 (where 26 is the last letter in the alphabet), this will shift the alphabet of a singular rotor forward 1 position. This can be shown in Figure 3.

Figure 3: Rotor Orientation

Ring Settings

The Ring setting changes the position of the notch by shifting the rotor's notch position forward for each number set. The user can change this by changing the drop down related to the rotor, each ring setting is situated directly below its origin rotor. This can be shown in Figure 4.

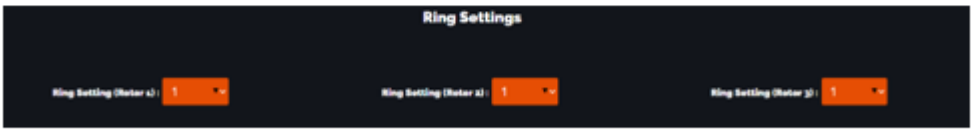


Figure 4: Ring Settings

Plugboard

The Final component of the Enigma is the plugboard which swaps the letter before that letter enters the rotors. An example of this is if "A" and "B" are connected to each other then when the user types "A" it will swap to "B". The implemented plugboard works by a user finding the plug they wish to connect say "E" and setting that plug letter to another say "A" after that the user must connect

plug “A” to “E” this creates a valid connection where the plugs are connected to each other. An invalid connection can be created if a user sets a plug that is not reciprocated, for example if plug “E” is connected to “A” but “A” is connected to “C”. An example of a valid connection can be found in Figure 5 and an invalid connection in Figure 6. On both the M1 and Kreigsmarine Enigmas the maximum number of plug connections is 10 due to the restrictions in wartime, it is also fine for the user to not connect plugs as this does not interfere with the encipherment.

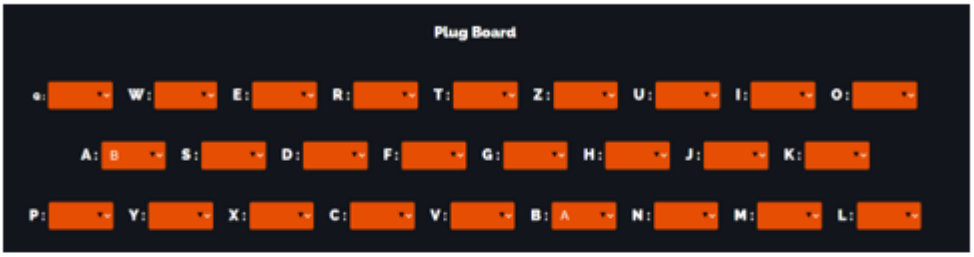


Figure 5: Valid Connection

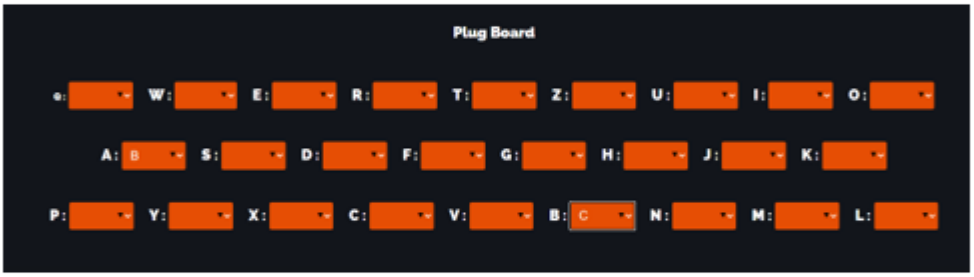


Figure 6: Invalid Connection

Encipherment

Now the user knows how to establish each setting it is possible for the user to encipher a message by typing on the supplied keyboard in Figure 7, and the results can be displayed on the read only text box seen in Figure 8. The decipherment process is exactly the same as the encipherment.

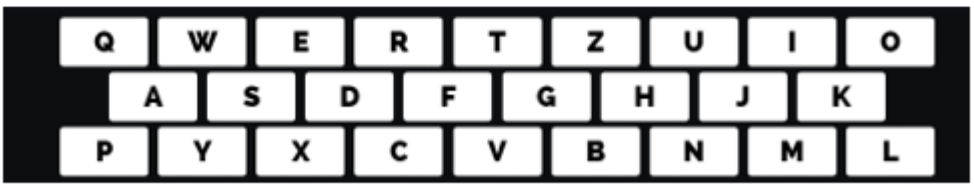


Figure 7: Keyboard

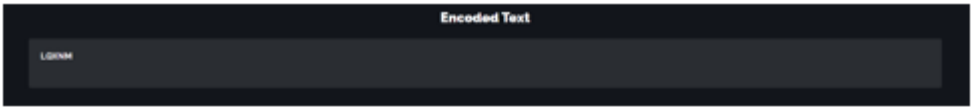


Figure 8: Output Box

Improved Enigma

What is the Improved Enigma?

The improved Enigma was designed to fix the main flaw in the Enigma where no letter enciphered could encipher exactly the same as the plaintext, this led to the Enigma being weak to a plaintext attack called cribbing which is discussed in the next section and the full report. Many designs were discussed for the improved Enigma and I encourage the user again to read the full report to understand all the options considered. For the purpose of this guide the switchboard was implemented and the Enigma functions nearly exactly as the basic Enigma above.

Switchboard

The switchboard turns a switch either on or off depending on its set state. If the switch is on the plaintext will self-encipher and turn off the switch and if not it will encipher normally. An example of this is if the user types "A" and the "A" switch is on then the letter will self-encipher as "A" then turn the switch off. The next time the user types "A" it will encipher to a different letter and turn on the switch. An example of this switchboard can be found in Figure 9.

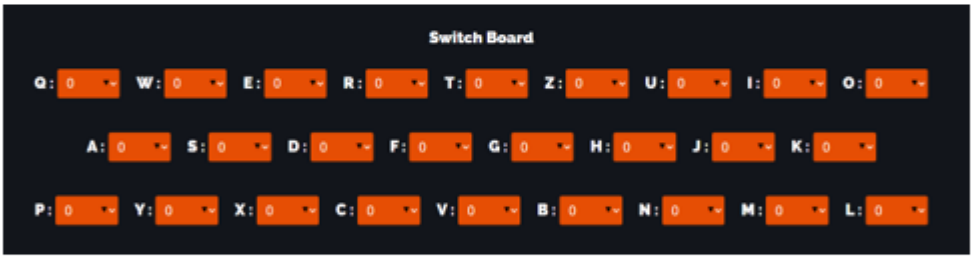


Figure 9: Switchboard

Cribbing

What is Cribbing

Cribbing is the main plaintext attack used to exploit the weakness of the Enigma. Cribbing works by matching a piece of plaintext to a piece of ciphertext accurately. This is done by checking each letter of the ciphertext to the plaintext if any letters match it's an invalid crib and if no letters match it's a valid crib. Valid cribs were utilised in the Bombe machine to break the Enigma.

Cribbing Machine

The implemented cribbing machine works by taking a user input in capitals and comparing it to a piece of ciphertext also in capitals. An example of this can be found in Figure 10.

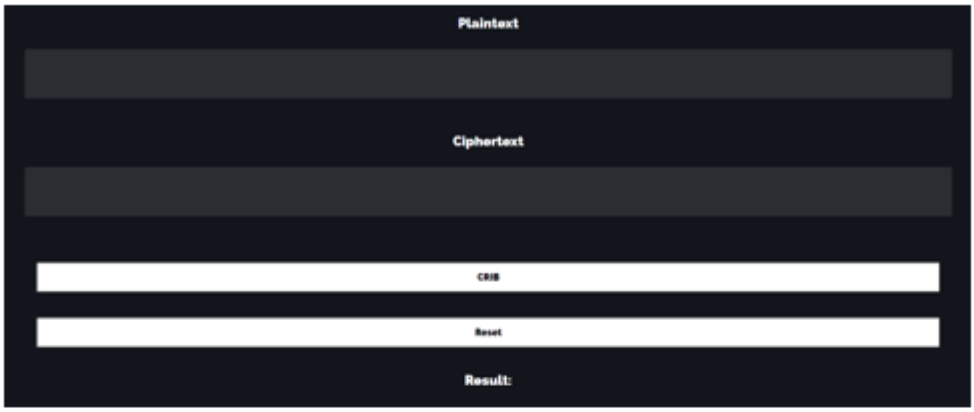


Figure 10: Cribbing Machine

Hashing

What is Hashing

Hashing is one-way encryption, it's designed to ensure message integrity which means that having duplicate versions of a hash may mean that someone could fake a document and render the integrity of the hash invalid. If the hash of plaintext "A" is the same as the hash of the fake plaintext then this is called a hash collision and is considered broken. There are many Hashing algorithms that could have been implemented, the developer chose MD5 which is broken and SHA-256 which is not.

How to Encipher

A user can encipher a message by inputting a message into the Hashing input box and copying the hash out of the output box. The input if the same will always result in the same output. This process can be shown in Figure 11.

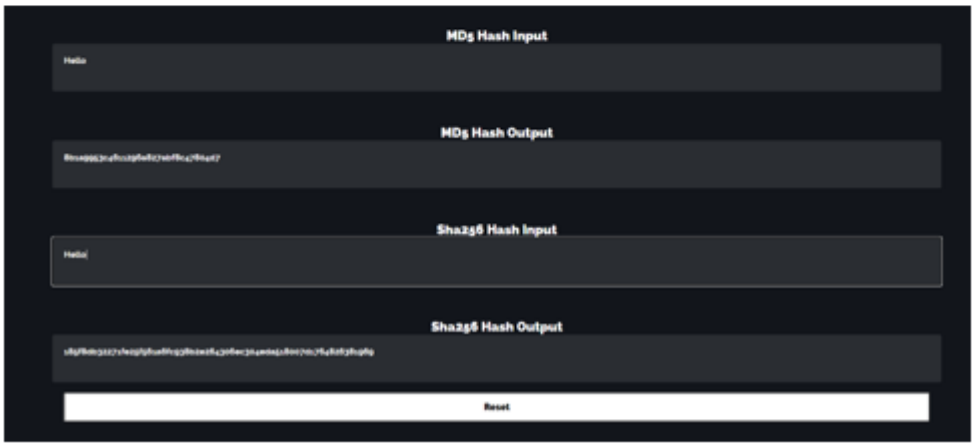


Figure 11: Hashing

AES

What is AES

AES or Advanced Encryption Standard is the current symmetrical encryption standard, it's a block cipher that goes though many different rounds and transformations. I implore the user to read the full report where the developer goes into full details about AES and its encipherment rounds. AES takes 3 different key sizes 128, 192- and 256-bit keys which due to the library that is implemented for AES the program can decide what key length to use based on the key input.

How to Encipher

If a user wishes to encipher using AES then they must input the message in the first input box then decide on what key length they wish to use. The user can then input the key into the key box and encrypt the message. Decryption can then happen using that same key. An example of this can be shown in Figure 12.

Figure 12: AES

Morse Code

What Is Morse code?

Morse code is the method of translating a word into a series of dots and dashes, it is not a form of encryption as the entire code is public, it's a form of alphabet that is convenient for the user to communicate over radio in a non-spoken format. It was used as the way to transfer encrypted Enigma messages,

How to use Morse Code

If a user wants to use Morse code all they have to do is type in the input box and the message will encode and decode as shown in Figure 13.

.

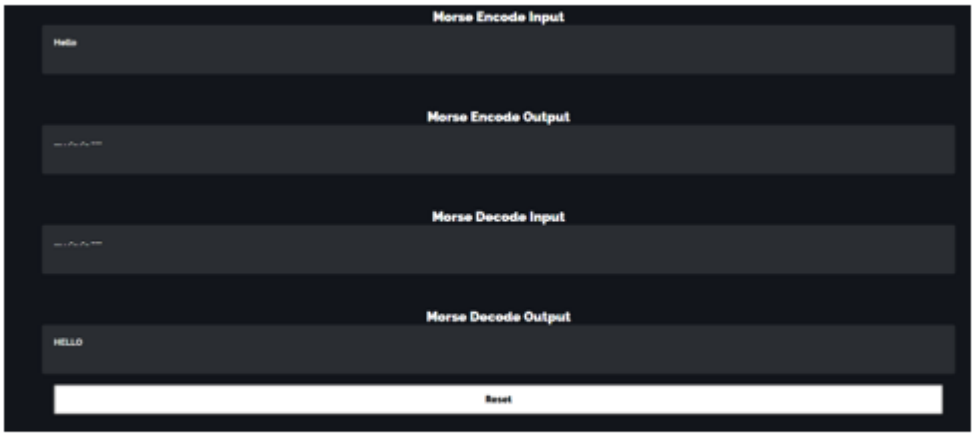


Figure 13: Morse Code

Glossary

Plaintext – The original message

Ciphertext – The encrypted message

AES (Advanced Encryption standard) – The modern symmetrical encryption standard

Hashing – one-way encoding, used for integrity and password storage

Enigma – the German World War 2 Encryption Device

Kriegsmarine – German Naval Enigma

Improved Enigma – The developers improved version of the Enigma machine

Ring Settings – The setting to change the notch position of the Enigma

Switchboard - Binary addition to the Enigma added by the developer

Plugboard – Set of inputs that swap a letter based on a valid connection

Rotor Order – the order each rotor in the enigma sits

Rotor Orientation – the setting each rotor is on

Key – Password/Phrase/Setting used to encrypt the data

Encrypt/Encipher – The act of making plaintext ciphertext

Decrypt/Decipher - The act of making ciphertext plaintext

MD5 (Message Digest 5) – Hashing method (Broken)

SHA-256 (Secure Hashing Algorithm 256) – Hashing method (Not Broken)

Cribbing – The act of matching plaintext to ciphertext accurately

Morse Code – Changing the letters into a dot and dash format that can be communicated over radio

Appendix C – Testing Table

Test	*Data*	*Expected Result*	*Actual Result*	*Pass/Fail*
Rotor 1 Changes plaintext	If key is pressed say "E"	Rotor 1 changes "E" to Different letter	Rotor 1 changes "E" to Different letter	Pass
Rotor 2 Changes Plaintext	If Rotor 1 changes "E" to Different letter	Rotor 2 Changes Different letter (1) to different letter (2)	Rotor 2 Changes Different letter (1) to different letter (2)	Pass
Rotor 3 Changes Plaintext	If Rotor 2 changes Different letter (1) to different letter (2)	Rotor 3 Changes Different letter (2) to different letter (3)	Rotor 3 Changes Different letter (2) to different letter (3)	Pass
Rotor 4 Changes Plaintext	If Rotor 2 changes Different letter (2) to different letter (3)	Rotor 4 Changes Different letter (4) to different letter (4)	Rotor 4 Changes Different letter (4) to different letter (4)	Pass
Plugboard Stops User when it hits limit	If more than 10 plugs are connected	Plugboard will error user	Plugboard will error user	Pass
Plugboard Changes letter	Plugboard connection is made from letter 1 to letter 2	When letter 1 is pressed it will switch to letter 2	When letter 1 is pressed it will switch to letter 2	Pass
Plugboard Doesn't allow invalid connections	Plugboard has an invalid connection	Plugboard will error user	Plugboard will error user	Pass
Reflector Switches Letter	Different letter (3) or (4) enters reflector and is changed to a reflected letter	Reflected Letter is returned	Reflected Letter is returned	Pass
Keyboard types	When letter pressed encrypted letter returned	Encrypted letter returned in output box	Encrypted letter returned in output box	Pass
Enigma Output observed is encrypted	Enigma Input results in ciphertext	Ciphertext is different from plaintext	Ciphertext is different from plaintext	Pass

Hashing Types return the same hashes with the same data	Plaintext inputted twice	Both Hashes match	Both Hashes match	Pass
Hashing Returns output consistent with other simulators	Plaintext Hash generated	Online Simulator matches this hash with same input	Online Simulator matches this hash with same input	Pass
Rotors push other rotors at notch points	Rotor reaches notch point	Rotor pushes next rotor	Rotor pushes next rotor	Pass
AES Enciphers and Deciphers	Plaintext	Ciphertext is different from plaintext and can be deciphered with the same key	Ciphertext is different from plaintext and can be deciphered with the same key	Pass
Morse Code Encodes and Decodes	Input supplied	Morse code translates input and decodes back to standard input	Morse code translates input and decodes back to standard input	Pass
Decryption of the Enigma is a mirror process to encryption	Ciphertext of plaintext supplied	Enigma can decrypt Enigma messages	Enigma can decrypt Enigma messages	Pass

Appendix D – Improved Enigma Evaluation

(Excel Sheet also included in code submission)

Password	Enigma Setting (Improved)	Enigma Encrypted	Cribbed	Start letter	End Letter	Start & End	Start Or End	Frequency
123456								
Password	R1: 12 R2: 15 R3: 5 Ring1: 1 Ring2: 5 Ring3: 22 Plugs Connected: None Switches R	BLVSFWRS						25.00%
12345678								
qwerty	R1: 25 R2: 1 R3: 26 Ring1: 26 Ring2: 21 Ring3: 14 Plugs Connected: None Switches H Q M	QKVJKP						16.67%
123456789								
12345								
1234								
111111								
1234567								
dragon	R1: 4 R2: 4 R3: 15 Ring1: 22 Ring2: 2 Ring3: 16 Plugs Connected: None Switches N S	CPRTSN						16.67%
123123								
baseball	R1: 15 R2: 15 R3: 9 Ring1: 20 Ring2: 13 Ring3: 24 Plugs Connected: GR VH IF KN Switches R G E S Q T B O	BUSEJAKL						62.50%
abc123								
football	R1: 7 R2: 3 R3: 23 Ring1: 1 Ring2: 2 Ring3: 8 Plugs Connected: RH Switches F N H T C A M D O K S X E	FOTTVANL						62.50%
monkey	R1: 14 R2: 5 R3: 10 Ring1: 17 Ring2: 26 Ring3: 4 Plugs Connected: MJ EV SG Switches X K J N P D W I H V C	OGNKSX						33.33%
letmein	R1: 9 R2: 4 R3: 3 Ring1: 10 Ring2: 7 Ring3: 11 Plugs Connected: KP BH Switches C R N I P W	THKHEIN						42.86%
696969								
shadow	R1: 10 R2: 2 R3: 3 Ring1: 2 Ring2: 8 Ring3: 10 Plugs Connected: None Switches J Q Y G	LXHKNA						0.00%
master	R1: 14 R2: 16 R3: 26 Ring1: 21 Ring2: 18 Ring3: 26 Plugs Connected: SK CP Switches U A B S B M R V W Q Y H E N D	MASTER						100%
666666								
qwertyuiop	R1: 7 R2: 6 R3: 19 Ring1: 23 Ring2: 2 Ring3: 24 Plugs Connected: None Switches Z H T B A N J K Y G L C D O	QPKQTYWSOH						40.00%
123321								
mustang	R1: 18 R2: 7 R3: 25 Ring1: 10 Ring2: 8 Ring3: 4 Plugs Connected: VN LW Switches T F K R G D E B M Q U L S A	MUSTAPG						85.71%
1234567890								
micheal	R1: 15 R2: 4 R3: 25 Ring1: 17 Ring2: 11 Ring3: 20 Plugs Connected: IZ TS UO AC DX Switches S X Z W G O N A P L C F R Q E	UDCLEAL						57.14%
654321								
pussy	R1: 17 R2: 14 R3: 24 Ring1: 6 Ring2: 24 Ring3: 8 Plugs Connected: OB RP LI DU Switches D H P V T N I R O L S C	PQSGU						40.00%
superman	R1: 1 R2: 22 R3: 6 Ring1: 5 Ring2: 9 Ring3: 23 Plugs Connected: RV XB EZ YQ MH Switches P Z C K H J I L E T A N Q	FQPEUQAN						50.00%
1qaz2wsx								
7777777								
fuckyou	R1: 7 R2: 10 R3: 14 Ring1: 11 Ring2: 9 Ring3: 9 Plugs Connected: AL OT ID Switches A N C M I B L D	YMCMRNU						28.57%
121212								
0000000								
qazwsx	R1: 22 R2: 21 R3: 17 Ring1: 6 Ring2: 8 Ring3: 4 Plugs Connected: None Switches I D J P Y B N O U V M H L Y S C X T	ZFOZSX						33.33%
123qwe								
killer	R1: 11 R2: 11 R3: 11 Ring1: 3 Ring2: 7 Ring3: 26 Plugs Connected: HC LT XK YQ WU GJ Switches E K Z F W T L D G J C X	KFLSES						50.00%
trustno1								
jordan	R1: 6 R2: 22 R3: 21 Ring1: 5 Ring2: 14 Ring3: 23 Plugs Connected: JL WX MC Switches Q P F I K N W J G L D A E X M	JXGDAN						66.67%
jennifer	R1: 8 R2: 16 R3: 22 Ring1: 11 Ring2: 19 Ring3: 12 Plugs Connected: AI NQ XL Switches H F J B T I D G O Q C X N	JHNYIFEC						62.50%
zxcvbnm	R1: 16 R2: 14 R3: 4 Ring1: 13 Ring2: 9 Ring3: 1 Plugs Connected: None Switches H E X J Z C R B K W U Y N Q P O F	ZXCPBNJ						62.50%
asdfgh	R1: 3 R2: 23 R3: 5 Ring1: 14 Ring2: 8 Ring3: 17 Plugs Connected: None Switches V P W Y X J U	QJAVAK						0.00%
hunter	R1: 6 R2: 22 R3: 1 Ring1: 25 Ring2: 18 Ring3: 3 Plugs Connected: UF QL OC IM Switches X A G Z T N P W L R Q Y U C	ZUNTGR						66.67%
buster	R1: 24 R2: 26 R3: 26 Ring1: 16 Ring2: 23 Ring3: 15 Plugs Connected: KL XM VG UB Switches R L X	YBHGZR						16.67%
soccer	R1: 9 R2: 17 R3: 16 Ring1: 23 Ring2: 26 Ring3: 5 Plugs Connected: None Switches V A Y B F D X U E Z P O K S I J C	SOCGEX						66.67%
harley	R1: 21 R2: 16 R3: 11 Ring1: 8 Ring2: 7 Ring3: 11 Plugs Connected: SA DL ZE HW FB UG RC JQ Switches L J M A	RANLKT						33.33%
batman	R1: 11 R2: 4 R3: 20 Ring1: 17 Ring2: 20 Ring3: 15 Plugs Connected: XF TB QS AY OW OB Switches T F Q I W D Y R	HBTEAC						33.33%
andrew	R1: 10 R2: 16 R3: 1 Ring1: 7 Ring2: 8 Ring3: 7 Plugs Connected: None Switches C U V O F X P W E L Q N B R G	RNJREW						66.67%
tigger	R1: 23 R2: 9 R3: 6 Ring1: 22 Ring2: 18 Ring3: 5 Plugs Connected: AP BH SV GW XL Switches B A P U Y	HMUGAT						16.67%
sunshine	R1: 22 R2: 13 R3: 26 Ring1: 1 Ring2: 7 Ring3: 7 Plugs Connected: None Switches O V W	RELSXQND						25.00%
iloveyou	R1: 8 R2: 3 R3: 5 Ring1: 11 Ring2: 12 Ring3: 11 Plugs Connected: None Switches O V W G A E P X R J C	SEOVESBL						37.50%
fuckme	R1: 14 R2: 16 R3: 7 Ring1: 12 Ring2: 13 Ring3: 22 Plugs Connected: None Switches D Y P	DSRQRR						0.00%
2000								
charlie	R1: 19 R2: 4 R3: 4 Ring1: 12 Ring2: 17 Ring3: 16 Plugs Connected: None Switches K N R P Z L F V S W H G Q E U C O	CHKRLLE						71.43%
robert	R1: 4 R2: 3 R3: 18 Ring1: 19 Ring2: 15 Ring3: 18 Plugs Connected: QR IL KP TS UN FG YX Switches O K H G M P I Z R	ROVUGS						33.33%
thomas	R1: 11 R2: 3 R3: 19 Ring1: 22 Ring2: 20 Ring3: 19 Plugs Connected: FV KB MG AO RZ QD PS Switches T B Z C O J M F X W M R I F P	TEOMLD						50.00%
hockey	R1: 23 R2: 26 R3: 7 Ring1: 17 Ring2: 2 Ring3: 15 Plugs Connected: None Switches A X C L F K I	UYCKVV						33.33%
ranger	R1: 10 R2: 25 R3: 21 Ring1: 10 Ring2: 19 Ring3: 20 Plugs Connected: None Switches B W Q L P S E R X I T N U C M J G	RTNGEG						66.67%
daniel	R1: 13 R2: 8 R3: 6 Ring1: 6 Ring2: 12 Ring3: 19 Plugs Connected: None Switches J F	EQZUNZ						0.00%
starwars	R1: 22 R2: 9 R3: 26 Ring1: 21 Ring2: 17 Ring3: 13 Plugs Connected: CB WS OK AJ ML ZP Switches A Z Q Y W N F G L B H E S J V T U D	STAQWCRN						62.50%
klaster	R1: 13 R2: 23 R3: 14 Ring1: 22 Ring2: 25 Ring3: 23 Plugs Connected: AO CN Switches W I U H Z O X E L Y G Q M S V	MLFSNEM						28.57%
112233								
george	R1: 13 R2: 12 R3: 5 Ring1: 20 Ring2: 3 Ring3: 14 Plugs Connected: TL NI YM UW GH Switches B R O K N E C V G D S L F J T U C K	GEORVT						66.67%
asshole	R1: 17 R2: 1 R3: 13 Ring1: 23 Ring2: 3 Ring3: 8 Plugs Connected: XG LM QP FW KU TZ EJ Switches N V X E H O F J G U	WESHOUE						57.14%
computer	R1: 6 R2: 22 R3: 12 Ring1: 9 Ring2: 15 Ring3: 14 Plugs Connected: UH GY LZ SV PI QX Switches L I G M A	PVMXJZYJ						12.50%
michelle	R1: 11 R2: 24 R3: 13 Ring1: 20 Ring2: 1 Ring3: 18 Plugs Connected: None Switches D O C L J P	ITCDYLFE						37.50%
jessica	R1: 8 R2: 22 R3: 5 Ring1: 10 Ring2: 11 Ring3: 20 Plugs Connected: None Switches N S L	UQSWJER						14.29%
pepper	R1: 20 R2: 11 R3: 10 Ring1: 3 Ring2: 8 Ring3: 5 Plugs Connected: JP OZ NF RU Switches R T G A K D Q C H X	VEPAKR						28.57%
111								
zxcvbn	R1: 26 R2: 9 R3: 25 Ring1: 17 Ring2: 3 Ring3: 22 Plugs Connected: VR AI FO GS UE Switches Z W E C D O R J B M G U Q	ZDCSBR						33.33%
555555								
11111111								
131313								
freedom	R1: 13 R2: 23 R3: 7 Ring1: 2 Ring2: 3 Ring3: 6 Plugs Connected: None Switches T	BGSEEFJ						14.29%
7777777								
pass	R1: 9 R2: 6 R3: 1 Ring1: 11 Ring2: 16 Ring3: 8 Plugs Connected: None Switches E Q W U H C V	HTVS						25.00%
fuck	R1: 15 R2: 26 R3: 13 Ring1: 5 Ring2: 22 Ring3: 18 Plugs Connected: LP GI VA HE Switches J Y R E	OOZZ						0.00%
maggie	R1: 3 R2: 14 R3: 23 Ring1: 10 Ring2: 15 Ring3: 20 Plugs Connected: RI JY DP VF ZK Switches G D N U B W Z	VTGUMA						16.67%
159753								
aaaaaaa	R1: 6 R2: 3 R3: 21 Ring1: 4 Ring2: 5 Ring3: 7 Plugs Connected: None Switches M W S I C Y P O G D W X	TAPAQAH						42.86%
ginger	R1: 8 R2: 8 R3: 20 Ring1: 9 Ring2: 17 Ring3: 19 Plugs Connected: None Switches T U M B F P E O U V G Q D Z	GOBNEA						16.67%
princess	R1: 21 R2: 8 R3: 26 Ring1: 18 Ring2: 12 Ring3: 4 Plugs Connected: WX NR TK VT GB ZL Switches Y R P M T G W V Z U A Q C K N J X	PRXNCOKS						50.00%
joshua	R1: 8 R2: 23 R3: 2 Ring1: 5 Ring2: 15 Ring3: 14 Plugs Connected: None Switches B V H U E G Z R X C P K L	GUTHUB						33.33%
cheese	R1: 20 R2: 1 R3: 19 Ring1: 18 Ring2: 13 Ring3: 1 Plugs Connected: ZS GD FX AC YO Switches P N R X M H U	PHVEHJ						33.33%
amanda	R1: 4 R2: 6 R3: 24 Ring1: 9 Ring2: 19 Ring3: 4 Plugs Connected: RL YQ XG AP DB Switches O K F Y L P U A B H C V G Z E N R W	AFRNUA						50.00%
summer	R1: 9 R2: 26 R3: 23 Ring1: 1 Ring2: 9 Ring3: 17 Plugs Connected: None Switches X Q A L C H V	ZBQMGU						16.67%
love	R1: 20 R2: 9 R3: 10 Ring1: 21 Ring2: 7 Ring3: 20 Plugs Connected: CV XK FR HS NY DE Switches D M L U	LTBU						25.00%
ashley	R1: 15 R2: 9 R3: 20 Ring1: 23 Ring2: 12 Ring3: 13 Plugs Connected: GZ JC Switches Y G T L Q V M L D S	FSNLFY						50.00%

6969			
nicole	R1: 24 R2: 15 R3: 8 Ring1: 23 Ring2: 22 Ring3: 11 Plugs Connected: None Switches Q	QJAGUR	0.00%
chelsea	R1: 10 R2: 24 R3: 25 Ring1: 7 Ring2: 20 Ring3: 24 Plugs Connected: DH FJ Switches F	PJUANEB	14.29%
biteme	R1: 20 R2: 12 R3: 12 Ring1: 19 Ring2: 11 Ring3: 2 Plugs Connected: XE DU TC PW Switches S N F V Z G E R L	ETKECQ	16.67%
matthew	R1: 5 R2: 9 R3: 8 Ring1: 3 Ring2: 3 Ring3: 20 Plugs Connected: ZQ FX VD KB OL GP WR Switches Q H S W A X V D N R F E O J Z B	DAETHEW	71.43%
access	R1: 4 R2: 15 R3: 1 Ring1: 7 Ring2: 19 Ring3: 21 Plugs Connected: EC KQ XW FH AY VL SO Switches I A B W Z K S H A N Y L C E P	ACTESM	66.67%
yankees	R1: 18 R2: 9 R3: 18 Ring1: 24 Ring2: 24 Ring3: 15 Plugs Connected: EO VU NA Switches N L E C S M W	BGNDENS	42.86%
987654321			
dallas	R1: 26 R2: 23 R3: 21 Ring1: 22 Ring2: 1 Ring3: 19 Plugs Connected: None Switches W R N H V	FQCLAN	33.33%
austin	R1: 13 R2: 25 R3: 7 Ring1: 3 Ring2: 24 Ring3: 6 Plugs Connected: EB SA XF ZJ ND Switches N J P S	LESVMN	33.33%
thunder	R1: 10 R2: 13 R3: 24 Ring1: 12 Ring2: 2 Ring3: 11 Plugs Connected: DU NR JY LT IM Switches F W Q N C J K I X E A R M O H B Z T	THONHER	71.43%
taylor	R1: 22 R2: 1 R3: 11 Ring1: 4 Ring2: 24 Ring3: 3 Plugs Connected: KA VG Switches B I A V K Z C P G Y W E X S O	PAYWOC	50.00%
matrix	R1: 14 R2: 22 R3: 2 Ring1: 20 Ring2: 19 Ring3: 18 Plugs Connected: None Switches H W	WNZTUJ	0.00%
william	R1: 17 R2: 18 R3: 21 Ring1: 21 Ring2: 19 Ring3: 19 Plugs Connected: GB WN CA IE SQ FX PK Switches P G S D Y	AMVLIUX	14.29%
corvette	R1: 2 R2: 8 R3: 19 Ring1: 3 Ring2: 1 Ring3: 6 Plugs Connected: YW Switches O B S	GOBQHWTE	25.00%
hello	R1: 19 R2: 8 R3: 7 Ring1: 20 Ring2: 5 Ring3: 20 Plugs Connected: ZY MK CF LV WG Switches S Z L V Y N U D C I B F G	WFLNC	20.00%
martin	R1: 6 R2: 7 R3: 18 Ring1: 2 Ring2: 1 Ring3: 26 Plugs Connected: OI JG BL Switches J O Q L K F H D	KMUERT	0.00%
heather	R1: 14 R2: 17 R3: 15 Ring1: 6 Ring2: 26 Ring3: 15 Plugs Connected: None Switches A C I V T W R Z U S L G X Y	XKATHER	71.43%
secret	R1: 16 R2: 16 R3: 26 Ring1: 20 Ring2: 10 Ring3: 23 Plugs Connected: Switches J G X B S U T I W	SOEBET	42.86%
Amazing	R1: 9 R2: 26 R3: 2 Ring1: 21 Ring2: 8 Ring3: 20 Plugs Connected: ZL VX WM IF Switches Y E B	MBAXXLB	14.29%
Merlin	R1: 24 R2: 4 R3: 14 Ring1: 17 Ring2: 12 Ring3: 7 Plugs Connected: SI NH TX YV WL Switches D F W V Y	QWOXAH	0.00%
diamond	R1: 24 R2: 15 R3: 4 Ring1: 12 Ring2: 22 Ring3: 23 Plugs Connected: None Switches B L U K F M W T G R C J P A E X Y	OOAMLDD	42.86%
1234qwer			
gfhjkm	R1: 5 R2: 26 R3: 10 Ring1: 12 Ring2: 6 Ring3: 6 Plugs Connected: LM WJ VF CR XO HU Switches F R O M L I E T K V N J Z D A Y	LFIJKM	50.00%
hammer	R1: 11 R2: 4 R3: 19 Ring1: 22 Ring2: 15 Ring3: 22 Plugs Connected: None Switches C A T M Y D U W H R E O Z K P L	HAMHER	83.33%
silver	R1: 5 R2: 21 R3: 15 Ring1: 16 Ring2: 18 Ring3: 16 Plugs Connected: JK UP AW FX Switches K N D R Y I X U T E B Z G Q	HIHEER	50.00%
222222			
88888888			
anthony	R1: 12 R2: 20 R3: 21 Ring1: 12 Ring2: 10 Ring3: 15 Plugs Connected: None Switches Z C N L V A X J M U Y Q F	ANZLATY	42.86%
poop	R1: 3 R2: 7 R3: 12 Ring1: 22 Ring2: 16 Ring3: 14 Plugs Connected: None Switches V P Y O W Q R U S L E C J M N	PORG	50.00%
test	R1: 26 R2: 10 R3: 26 Ring1: 21 Ring2: 1 Ring3: 22 Plugs Connected: None Switches K Q W R	RAGT	25.00%
bailey	R1: 20 R2: 12 R3: 9 Ring1: 2 Ring2: 18 Ring3: 15 Plugs Connected: None Switches L Z K Y Q X	VMHLNY	28.57%
q1w2e3r4t5			
patrik	R1: 19 R2: 21 R3: 9 Ring1: 11 Ring2: 11 Ring3: 12 Plugs Connected: SC ZJ BH GA MD FK Switches N O E S X L A Q	DAVWVR	16.67%
internet	R1: 8 R2: 3 R3: 17 Ring1: 15 Ring2: 15 Ring3: 15 Plugs Connected: VJ TR NZ KS WU AL DI Switches T C F Z S J	WXTWYNEO	37.50%
scooter	R1: 10 R2: 25 R3: 4 Ring1: 25 Ring2: 12 Ring3: 25 Plugs Connected: ZW LG OT Switches F W I X	QKRODGM	14.29%
orange	R1: 24 R2: 8 R3: 17 Ring1: 17 Ring2: 20 Ring3: 19 Plugs Connected: YG IN DW JZ EP Switches D M C H L U R Y J B T	TRJEYB	16.67%
11111			
golfer	R1: 20 R2: 24 R3: 4 Ring1: 13 Ring2: 19 Ring3: 11 Plugs Connected: EU HC SF KM OX JP Switches Z A O U J C K X R B I S Y N Q W	ZOXGCR	33.33%
cookie	R1: 16 R2: 12 R3: 15 Ring1: 3 Ring2: 18 Ring3: 13 Plugs Connected: CB FV XK RN OD LY Switches I S L P M A X Y V	DFOAIW	28.57%
richard	R1: 22 R2: 5 R3: 18 Ring1: 12 Ring2: 10 Ring3: 8 Plugs Connected: None Switches A C S W U H	ZPCHARF	57.14%
samantha	R1: 7 R2: 24 R3: 22 Ring1: 12 Ring2: 16 Ring3: 18 Plugs Connected: None Switches Z G R T V S U W L X	SZKAPTZF	37.50%
bigdog	R1: 20 R2: 12 R3: 14 Ring1: 1 Ring2: 20 Ring3: 9 Plugs Connected: None Switches T W O F B Z X N L P A J H S A G Q K	BCGEOX	50.00%
guitar	R1: 4 R2: 3 R3: 11 Ring1: 23 Ring2: 17 Ring3: 16 Plugs Connected: UT MD BA OF ZK WC LI Switches B I D W T E C	KNITGY	33.33%
jackson	R1: 6 R2: 13 R3: 11 Ring1: 17 Ring2: 26 Ring3: 13 Plugs Connected: WG Switches R P Q A	RAMDWDC	14.29%
whatever	R1: 2 R2: 5 R3: 16 Ring1: 15 Ring2: 18 Ring3: 2 Plugs Connected: None Switches S U H	AHZDJIEI	25.00%
mickey	R1: 2 R2: 16 R3: 22 Ring1: 1 Ring2: 9 Ring3: 15 Plugs Connected: KR BU NC EF Switches X	VBSIMK	0.00%
chicken	R1: 9 R2: 13 R3: 14 Ring1: 23 Ring2: 14 Ring3: 5 Plugs Connected: None Switches B U R X	JFSCEUT	14.29%
sparky	R1: 5 R2: 5 R3: 25 Ring1: 12 Ring2: 11 Ring3: 17 Plugs Connected: WF ZO QN HE YR KT Switches E S Y G A V F Z H B J O C	SJALAY	42.86%
snoopy	R1: 16 R2: 17 R3: 22 Ring1: 18 Ring2: 25 Ring3: 6 Plugs Connected: WX MQ NJ GE FL UY HN Switches S R L X D Y Z C G A F Q U B	SUROEY	42.86%
maverick	R1: 8 R2: 24 R3: 24 Ring1: 6 Ring2: 23 Ring3: 20 Plugs Connected: None Switches X H Z L O F G N U A E I Q S Y J	AATEGITG	37.50%
pheonix	R1: 12 R2: 25 R3: 15 Ring1: 24 Ring2: 18 Ring3: 14 Plugs Connected: KL IQ FO EY CU JM TD Switches C E F A K H X D Y R T	IHEVMTX	42.86%
camaro	R1: 23 R2: 19 R3: 10 Ring1: 9 Ring2: 26 Ring3: 2 Plugs Connected: EL KU SW PR Switches T R H J Z I K Q S O	HKHARO	50.00%
sexy	R1: 6 R2: 9 R3: 1 Ring1: 18 Ring2: 1 Ring3: 9 Plugs Connected: None Switches G U H R P W X O K J	WPXP	25.00%
peanut	R1: 22 R2: 8 R3: 11 Ring1: 1 Ring2: 15 Ring3: 22 Plugs Connected: JY LP KQ Switches L N J X V E	AEPNQV	33.33%
morgan	R1: 19 R2: 1 R3: 19 Ring1: 17 Ring2: 19 Ring3: 2 Plugs Connected: None Switches L F X O N D E S V H Y Q	UOGOHN	33.33%
welcome	R1: 10 R2: 20 R3: 16 Ring1: 26 Ring2: 22 Ring3: 2 Plugs Connected: SM AE Switches Y O H S L T N Z C R U M	ZQLCOME	71.43%
falcon	R1: 4 R2: 25 R3: 13 Ring1: 7 Ring2: 15 Ring3: 25 Plugs Connected: None Switches G U Z E O J M L F Q	FKLNOS	50.00%
cowboy	R1: 11 R2: 6 R3: 5 Ring1: 25 Ring2: 5 Ring3: 9 Plugs Connected: None Switches I C W O F S I N L Q T M J X Z	COWGCX	50.00%
ferrari	R1: 23 R2: 13 R3: 16 Ring1: 11 Ring2: 16 Ring3: 7 Plugs Connected: KA BV Switches X R V Y U A M Q E D S Z L H J	ZERIARD	42.86%
samsung	R1: 24 R2: 18 R3: 4 Ring1: 16 Ring2: 7 Ring3: 16 Plugs Connected: LR KU ZS CM BA VW IO Switches U Y W C O M D A X B L Z S G H	SAMEUAG	71.43%
andrea	R1: 26 R2: 4 R3: 2 Ring1: 3 Ring2: 22 Ring3: 16 Plugs Connected: IF ZE SV MP Switches J Z V E D G O F K X	LRDJEA	50.00%
smokey	R1: 8 R2: 5 R3: 25 Ring1: 2 Ring2: 17 Ring3: 14 Plugs Connected: None Switches U F H V A G T Y Q X B R N D	UZQYTY	16.67%
steelers	R1: 1 R2: 11 R3: 12 Ring1: 24 Ring2: 26 Ring3: 11 Plugs Connected: None Switches O I U X R T J Q H N P Y G M E F	NTEVTERS	62.50%
joeseph	R1: 4 R2: 19 R3: 3 Ring1: 12 Ring2: 19 Ring3: 24 Plugs Connected: None Switches N Q U B Y W X I H Z V D L	TSFDEVH	28.57%
mercedes	R1: 20 R2: 22 R3: 13 Ring1: 22 Ring2: 7 Ring3: 9 Plugs Connected: UP FW VB GD Switches K Q J C	EVTCEYOY	25.00%
dakota	R1: 11 R2: 7 R3: 4 Ring1: 5 Ring2: 21 Ring3: 20 Plugs Connected: TX QY Switches U O Z N K L G J S D Y W F H A	DAKOPF	66.67%
arsenal	R1: 20 R2: 18 R3: 11 Ring1: 3 Ring2: 9 Ring3: 4 Plugs Connected: None Switches D H M V L S K Z X T Q C N W R O Y E P G	WRSENAL	85.71%
eagles	R1: 16 R2: 2 R3: 19 Ring1: 5 Ring2: 16 Ring3: 6 Plugs Connected: None Switches K S H I N F W T R U O Z V B	TLTFES	33.33%
melissa	R1: 14 R2: 15 R3: 15 Ring1: 20 Ring2: 23 Ring3: 22 Plugs Connected: YK OQ Switches R	RSXRMSP	12.50%
boomer	R1: 8 R2: 19 R3: 16 Ring1: 1 Ring2: 9 Ring3: 25 Plugs Connected: TS UH BA DV Switches U P I T G S L C	WQOWXW	16.67%
booboo	R1: 21 R2: 11 R3: 19 Ring1: 3 Ring2: 3 Ring3: 14 Plugs Connected: TC Switches E V O L X A R N P T	XOXBOB	50.00%
spider	R1: 24 R2: 8 R3: 13 Ring1: 19 Ring2: 26 Ring3: 25 Plugs Connected: LO XC NM Switches C I V N M P L B F A T	LPINHH	16.67%
nascar	R1: 15 R2: 26 R3: 5 Ring1: 18 Ring2: 19 Ring3: 16 Plugs Connected: WB EZ QA NG OK Switches D F K S U V A O J C	GASCLK	50.00%
monster	R1: 13 R2: 2 R3: 4 Ring1: 7 Ring2: 13 Ring3: 12 Plugs Connected: XD AL RQ FI JD JC Switches F B G I Q D T C S U A	SMWSTNZ	25.00%
tigers	R1: 23 R2: 8 R3: 22 Ring1: 2 Ring2: 26 Ring3: 7 Plugs Connected: None Switches C Q G O D K F U N A Y I H E B J	NIGEOT	50.00%
yellow	R1: 8 R2: 17 R3: 7 Ring1: 4 Ring2: 22 Ring3: 2 Plugs Connected: MX LJ PV CH NQ IY FZ Switches W O M S X N U H R	PZBLOW	50.00%
xxxxxx	R1: 16 R2: 25 R3: 10 Ring1: 20 Ring2: 19 Ring3: 7 Plugs Connected: None Switches L U G D F E N S O H Y B V	RXSXTX	42.86%
123123123			
gateway	R1: 4 R2: 23 R3: 17 Ring1: 7 Ring2: 9 Ring3: 20 Plugs Connected: AF IT OB JS LZ Switches K V R N C X O H E Y B L W I Z Q	AYMEWAY	50.00%
marina	R1: 3 R2: 20 R3: 24 Ring1: 21 Ring2: 3 Ring3: 3 Plugs Connected: None Switches G A U O Y B J T L F Q I Z V W X	XAJIMX	33.33%
diablo	R1: 19 R2: 20 R3: 7 Ring1: 4 Ring2: 17 Ring3: 12 Plugs Connected: None Switches R E I K F Z L U S D M T X H A	DIAPLT	66.67%
bulldog	R1: 12 R2: 13 R3: 4 Ring1: 3 Ring2: 3 Ring3: 16 Plugs Connected: None Switches T S J O H F C G	GMPLVOG	42.86%
qwer1234			
compaq	R1: 5 R2: 14 R3: 13 Ring1: 21 Ring2: 1 Ring3: 19 Plugs Connected: XS KD GB PV LJ Switches E J	ESCETR	0.00%
purple	R1: 13 R2: 23 R3: 25 Ring1: 11 Ring2: 4 Ring3: 1 Plugs Connected: LO WS NX BE FG Switches Z M T	JIHPVX	16.67%
hardcore	R1: 10 R2: 23 R3: 1 Ring1: 15 Ring2: 5 Ring3: 3 Plugs Connected: XG TQ NL EY Switches D I S K J N	TJODNGRB	25.00%
banana	R1: 12 R2: 26 R3: 9 Ring1: 13 Ring2: 20 Ring3: 16 Plugs Connected: IP MN FC ZE SO Switches F M W C J A B N I Q D T H	BANTIA	66.67%
junior	R1: 26 R2: 4 R3: 15 Ring1: 4 Ring2: 19 Ring3: 8 Plugs Connected: None Switches H	PCORNU	0.00%
hannah	R1: 16 R2: 1 R3: 20 Ring1: 12 Ring2: 24 Ring3: 19 Plugs Connected: None Switches A E G R S C M N I V K O Y P	KANAYH	50.00%

123654			
porsche	R1: 12 R2: 15 R3: 6 Ring1: 21 Ring2: 14 Ring3: 7 Plugs Connected: CH BZ AN QP YO MS Switches N P	PPSPAMC	28.57%
lakers	R1: 19 R2: 25 R3: 15 Ring1: 17 Ring2: 11 Ring3: 13 Plugs Connected: None Switches U P E Z D Y G Q C S R T	CICERS	50.00%
iceman	R1: 5 R2: 11 R3: 13 Ring1: 14 Ring2: 16 Ring3: 16 Plugs Connected: RB TI Switches U C B A H P Y L Z	WCAIAI	33.33%
money	R1: 17 R2: 6 R3: 20 Ring1: 20 Ring2: 7 Ring3: 20 Plugs Connected: TH DA VF ZQ Switches E S O Y K C J N B Q I W R P A X Z D	RONEY	80.00%
cowboys	R1: 4 R2: 10 R3: 5 Ring1: 25 Ring2: 12 Ring3: 16 Plugs Connected: ID HG JX AE WB PZ VQ Switches H K N A L Y V M C Z F I E D P W	CCWZOYV	57.14%
987654			
london	R1: 18 R2: 13 R3: 6 Ring1: 9 Ring2: 9 Ring3: 26 Plugs Connected: None Switches E U M K N T G X P B W F J Q	SXNBOO	33.33%
tennis	R1: 17 R2: 16 R3: 16 Ring1: 22 Ring2: 14 Ring3: 4 Plugs Connected: None Switches V F J U R D Q T K Y H E M	TEANFL	50.00%
999999			
ncc1701			
coffee	R1: 13 R2: 12 R3: 20 Ring1: 3 Ring2: 11 Ring3: 24 Plugs Connected: None Switches C W H M	CNXFTE	50.00%
scooby	R1: 22 R2: 18 R3: 5 Ring1: 10 Ring2: 4 Ring3: 9 Plugs Connected: VH Switches A U V L M K S Y F C Q	SCNOKY	57.14%
0000			
miller	R1: 12 R2: 2 R3: 10 Ring1: 16 Ring2: 25 Ring3: 7 Plugs Connected: FA OM ND Switches U I F Z G N	EIJLLI	16.67%
boston	R1: 22 R2: 23 R3: 25 Ring1: 2 Ring2: 22 Ring3: 6 Plugs Connected: None Switches Q E O Q H D C I T V J W K B A	BOMTSJ	50.00%
q1w2e3r4			
fuckoff	R1: 25 R2: 6 R3: 7 Ring1: 5 Ring2: 25 Ring3: 12 Plugs Connected: None Switches J F M	FVNOTMF	14.29%
brandon	R1: 17 R2: 13 R3: 4 Ring1: 25 Ring2: 5 Ring3: 12 Plugs Connected: SW TX NI Switches H B	BGOKOPN	28.57%
yamaha	R1: 8 R2: 15 R3: 7 Ring1: 19 Ring2: 7 Ring3: 3 Plugs Connected: IO NY HG SB XW DJ Switches F Q C O B U P V W E Y J T	YQPASW	33.33%
chester	R1: 8 R2: 17 R3: 7 Ring1: 10 Ring2: 22 Ring3: 16 Plugs Connected: None Switches A L T V Q G O D	QCYDTET	28.57%
mother	R1: 25 R2: 14 R3: 9 Ring1: 22 Ring2: 19 Ring3: 26 Plugs Connected: EW JH DZ XO PF QV Switches Q J N A P F D Y E R H U	AVXHER	50.00%
forever	R1: 11 R2: 8 R3: 16 Ring1: 10 Ring2: 1 Ring3: 3 Plugs Connected: FH DE YB VQ XI AR Switches X A J P Y K T H G N F	FISSZER	42.86%
johnny	R1: 18 R2: 21 R3: 9 Ring1: 22 Ring2: 18 Ring3: 17 Plugs Connected: CU TI Switches E Y L J V C K D W P U I	JPBBNY	50.00%
edward	R1: 12 R2: 20 R3: 5 Ring1: 3 Ring2: 7 Ring3: 25 Plugs Connected: TM ES OV KN ZU Switches F Q K U H R V I P Y O	RNRFRD	28.57%
333333			
oliver	R1: 5 R2: 4 R3: 26 Ring1: 24 Ring2: 14 Ring3: 5 Plugs Connected: PB UD KG IF AN Switches T Z P M S W	NMFPDU	0.00%
redsox	R1: 2 R2: 22 R3: 21 Ring1: 22 Ring2: 16 Ring3: 11 Plugs Connected: None Switches N U H A S D B	WNDSFY	33.33%
player	R1: 17 R2: 6 R3: 19 Ring1: 6 Ring2: 9 Ring3: 8 Plugs Connected: None Switches X Q S V N Y H K B J U M G L P	PLIYFP	50.00%
nikita	R1: 10 R2: 11 R3: 12 Ring1: 3 Ring2: 5 Ring3: 10 Plugs Connected: None Switches O H J N V I A B W T E M Y P U	NICFTA	50.00%
knight	R1: 19 R2: 20 R3: 7 Ring1: 16 Ring2: 18 Ring3: 14 Plugs Connected: None Switches X S J H M A U O V Y I T R N Q W	VNIEHT	66.67%
fender	R1: 5 R2: 21 R3: 1 Ring1: 18 Ring2: 24 Ring3: 13 Plugs Connected: YT EC KA LQ NV Switches T W P U Q J	RXDAEX	16.67%
barney	R1: 20 R2: 5 R3: 18 Ring1: 21 Ring2: 5 Ring3: 17 Plugs Connected: None Switches A E T Q J Y P N D M Z B F O K L	BAVNEY	85.71%
midnight	R1: 11 R2: 7 R3: 16 Ring1: 2 Ring2: 3 Ring3: 3 Plugs Connected: EB WX ZL Switches D E G Z P B T N K Q M Y O	MHDNIGXT	75.00%
please	R1: 4 R2: 10 R3: 1 Ring1: 2 Ring2: 6 Ring3: 23 Plugs Connected: OT VJ Switches L P Z X I D K T J A N Y U W F B M	PLXATE	66.67%
brandy	R1: 26 R2: 5 R3: 1 Ring1: 23 Ring2: 24 Ring3: 8 Plugs Connected: None Switches P D R U	WRISDD	28.57%
chicago	R1: 4 R2: 7 R3: 22 Ring1: 17 Ring2: 19 Ring3: 20 Plugs Connected: None Switches A K P V L Q I U X Y N G F	SNICAGE	57.14%
badboy	R1: 8 R2: 9 R3: 21 Ring1: 17 Ring2: 12 Ring3: 15 Plugs Connected: NJ YE GK KC MR AP Switches B K A T S V E J Z X N P L Q	BAMSCC	16.67%
iwantu	R1: 25 R2: 23 R3: 5 Ring1: 24 Ring2: 11 Ring3: 12 Plugs Connected: ZB XQ FJ PU Switches D R A F C T M S	TGAWTF	33.33%
slayer	R1: 2 R2: 6 R3: 23 Ring1: 9 Ring2: 18 Ring3: 15 Plugs Connected: None Switches G L A S U N K Q X F I W M R	SLAFYR	66.67%
rangers	R1: 5 R2: 7 R3: 10 Ring1: 9 Ring2: 25 Ring3: 19 Plugs Connected: HW EV UM XF Switches T Q E L S	SIPBERS	42.86%
charles	R1: 11 R2: 12 R3: 8 Ring1: 23 Ring2: 21 Ring3: 5 Plugs Connected: None Switches P F H E G V S O N R X Q T B	THSRZES	57.14%
angel	R1: 4 R2: 9 R3: 10 Ring1: 6 Ring2: 1 Ring3: 26 Plugs Connected: None Switches S K L U T	QPHWL	20.00%
flower	R1: 16 R2: 7 R3: 20 Ring1: 7 Ring2: 20 Ring3: 13 Plugs Connected: None Switches G F D S J P Z R O E L C	FLOHER	83.33%
bigdaddy	R1: 13 R2: 22 R3: 1 Ring1: 9 Ring2: 20 Ring3: 12 Plugs Connected: KM CD Switches D P G A V J U X T B N Z I L W	BIGDAXDJ	75.00%
rabbit	R1: 18 R2: 6 R3: 5 Ring1: 24 Ring2: 20 Ring3: 20 Plugs Connected: None Switches O R T Y F M W S C U Z B K	RDBYQT	50.00%
wizard	R1: 20 R2: 6 R3: 6 Ring1: 10 Ring2: 3 Ring3: 5 Plugs Connected: WE OK Switches E	FFKSNP	0.00%
bigdick	R1: 9 R2: 2 R3: 20 Ring1: 6 Ring2: 13 Ring3: 12 Plugs Connected: None Switches G I L A O Q F N	KIGJEBG	33.33%
jasper	R1: 4 R2: 7 R3: 25 Ring1: 11 Ring2: 9 Ring3: 5 Plugs Connected: LC TJ SM YZ IP Switches X S E O K R Q Y P Z K D H V	FDSPER	66.67%
enter	R1: 19 R2: 26 R3: 9 Ring1: 9 Ring2: 3 Ring3: 20 Plugs Connected: OT PN SQ ZD Switches V W U K O T A D Q I S H R Y C	UUTER	60.00%
rachel	R1: 16 R2: 5 R3: 14 Ring1: 5 Ring2: 12 Ring3: 20 Plugs Connected: JW RE VS PX OF QU IK Switches Y X H C Z W V K M N D A J P S L O	WACHPL	66.67%
chris	R1: 9 R2: 26 R3: 26 Ring1: 19 Ring2: 26 Ring3: 2 Plugs Connected: None Switches P L I Q F W B H	VHQIB	40.00%
steven	R1: 13 R2: 16 R3: 6 Ring1: 5 Ring2: 19 Ring3: 17 Plugs Connected: None Switches F W G Q	BCVJEY	14.29%
winner	R1: 26 R2: 13 R3: 8 Ring1: 18 Ring2: 12 Ring3: 17 Plugs Connected: MJ QS NH Switches D	LSANBJ	16.67%
adidas	R1: 12 R2: 14 R3: 12 Ring1: 26 Ring2: 20 Ring3: 11 Plugs Connected: None Switches S V P F A X T Q Z C O E L N H	AJRDLS	50.00%
victoria	R1: 8 R2: 9 R3: 6 Ring1: 19 Ring2: 3 Ring3: 17 Plugs Connected: None Switches W Q X B D Z I E F S T	SISTSASM	12.50%
natasha	R1: 22 R2: 26 R3: 25 Ring1: 18 Ring2: 24 Ring3: 1 Plugs Connected: UC JH WI Switches Y U D N J Z M A G B O K F I	NALTRFA	42.86%
1q2w3e4r			
jasmine	R1: 6 R2: 14 R3: 25 Ring1: 14 Ring2: 17 Ring3: 22 Plugs Connected: RG DB OJ UV XL CT MZ NI Switches S M D H	CFSMYBO	28.57%
winter	R1: 26 R2: 13 R3: 21 Ring1: 9 Ring2: 8 Ring3: 6 Plugs Connected: WZ GH QY AK RN Switches Q V Y P O U X	ATBLZV	0.00%
prince	R1: 26 R2: 10 R3: 16 Ring1: 23 Ring2: 22 Ring3: 24 Plugs Connected: None Switches B A K M I L Z Y E V T W R Q G X	BRIZGE	50.00%
panties	R1: 8 R2: 19 R3: 6 Ring1: 26 Ring2: 14 Ring3: 15 Plugs Connected: EB CR KY MA VF Switches D R X	BSPXQCQ	0.00%
marine	R1: 7 R2: 10 R3: 21 Ring1: 23 Ring2: 25 Ring3: 8 Plugs Connected: MC HX IV SF EA ZB LJ GU Switches T L J A C L G W F N M Z Q H K X	MAMINC	66.67%
ghbdtn	R1: 10 R2: 20 R3: 6 Ring1: 6 Ring2: 25 Ring3: 1 Plugs Connected: AU NR EC FY GV LK Switches F M C B I L S P E N Y K	RXBNBN	33.33%
fishing	R1: 23 R2: 21 R3: 16 Ring1: 2 Ring2: 19 Ring3: 23 Plugs Connected: None Switches H T Z N F M E W L	FURHIND	42.86%
cocacola	R1: 16 R2: 17 R3: 22 Ring1: 23 Ring2: 17 Ring3: 10 Plugs Connected: None Switches R L O N K	YOCUJWLA	37.50%
casper	R1: 16 R2: 19 R3: 12 Ring1: 9 Ring2: 7 Ring3: 16 Plugs Connected: None Switches P I F M T K V W E	SBEPEJ	33.33%
james	R1: 15 R2: 19 R3: 17 Ring1: 20 Ring2: 14 Ring3: 9 Plugs Connected: None Switches F X Z H Y J G S B K C T	JXDVS	40.00%
232323			
raiders	R1: 2 R2: 22 R3: 6 Ring1: 4 Ring2: 22 Ring3: 4 Plugs Connected: UH RN OX Switches I Q M K B J G E L W A O S C X	QAIHERS	71.43%
8888888			
marlboro	R1: 8 R2: 6 R3: 26 Ring1: 24 Ring2: 2 Ring3: 7 Plugs Connected: FR UX ZB ON Switches K J Z R E G C L N Y F H O M U W T	MKRLPOIE	50.00%
gandalf	R1: 7 R2: 21 R3: 25 Ring1: 8 Ring2: 21 Ring3: 4 Plugs Connected: UR ZQ XM Switches D C I J O Z V S A U G H F	GAWDEEF	42.86%
asdfasdf	R1: 15 R2: 10 R3: 6 Ring1: 23 Ring2: 5 Ring3: 14 Plugs Connected: GW FE OB KT QL Switches D	HLDYASWF	50.00%
crystal	R1: 8 R2: 11 R3: 4 Ring1: 10 Ring2: 18 Ring3: 18 Plugs Connected: None Switches Z Q F T M V I E	YNBJTQJ	14.29%
87654321			
12344321			
sexsex	R1: 6 R2: 6 R3: 16 Ring1: 1 Ring2: 18 Ring3: 7 Plugs Connected: None Switches A Q U	TPWSEX	50.00%
golden	R1: 22 R2: 18 R3: 8 Ring1: 4 Ring2: 2 Ring3: 22 Plugs Connected: None Switches N G C K O Y S L B I W H Z M	GOLSIN	50.00%
blowme	R1: 5 R2: 15 R3: 23 Ring1: 12 Ring2: 22 Ring3: 18 Plugs Connected: None Switches F Y J I B H S Z V U C G	BAYYJT	16.67%
bigtits	R1: 21 R2: 18 R3: 25 Ring1: 7 Ring2: 22 Ring3: 7 Plugs Connected: AS FR BV IO WM Switches V P A B Q L M	BYNBITE	33.33%
8765309			
panther	R1: 14 R2: 14 R3: 3 Ring1: 7 Ring2: 23 Ring3: 17 Plugs Connected: IK LR VS ZC Switches D R A F I B X P O U S M	PAXPNLR	42.86%
lauren	R1: 21 R2: 16 R3: 12 Ring1: 3 Ring2: 21 Ring3: 21 Plugs Connected: XE CD GK ZJ Switches C O L Z	LWHJDH	16.67%
angela	R1: 4 R2: 26 R3: 10 Ring1: 17 Ring2: 10 Ring3: 23 Plugs Connected: CQ KM HB ON Switches Q C W P V	DEEXTA	16.67%
bitch	R1: 26 R2: 5 R3: 12 Ring1: 2 Ring2: 19 Ring3: 16 Plugs Connected: VX ST CM AD UP Switches P Z U A N K B D S V C T H F X E M L	BCTCH	80.00%
spanky	R1: 22 R2: 10 R3: 21 Ring1: 1 Ring2: 7 Ring3: 13 Plugs Connected: YP KB JV RF Switches D N Q H	LIYNEK	16.67%
thx1138			
angels	R1: 20 R2: 20 R3: 9 Ring1: 8 Ring2: 18 Ring3: 18 Plugs Connected: BE AI XQ TK PF Switches H L I B R T O U S N M Q Z D	KNTNLS	33.33%
madison	R1: 23 R2: 23 R3: 8 Ring1: 24 Ring2: 14 Ring3: 17 Plugs Connected: SO LP Switches X B Z U	ZSFLKDI	0.00%

winston	R1: 20 R2: 17 R3: 6 Ring1: 9 Ring2: 20 Ring3: 19 Plugs Connected: LR NP TE BH VK YQ SW Switches W U O N M	WPNODOY	42.86%
shannon	R1: 3 R2: 13 R3: 9 Ring1: 9 Ring2: 4 Ring3: 2 Plugs Connected: None Switches P B Y L	BBHHNCB	14.29%
mike	R1: 13 R2: 1 R3: 7 Ring1: 11 Ring2: 11 Ring3: 2 Plugs Connected: None Switches K	UKKR	25.00%
toyota	R1: 18 R2: 5 R3: 18 Ring1: 2 Ring2: 22 Ring3: 25 Plugs Connected: OT ZV HW RL Switches P X	ZPWOTP	33.33%
blowjob	R1: 19 R2: 6 R3: 16 Ring1: 16 Ring2: 8 Ring3: 18 Plugs Connected: VO QP LR AT KY MF SZ CJ Switches Q H T S O V C Y K R X W G Z I	SOOWXGB	42.86%
jordan123			
canada	R1: 14 R2: 14 R3: 2 Ring1: 11 Ring2: 21 Ring3: 18 Plugs Connected: None Switches F X U C Y B E K L G V P O W M	CLZATS	33.33%
sophie	R1: 12 R2: 6 R3: 4 Ring1: 22 Ring2: 5 Ring3: 12 Plugs Connected: None Switches I T U K H	PVEHIM	33.33%
Password	R1: 18 R2: 19 R3: 25 Ring1: 2 Ring2: 19 Ring3: 17 Plugs Connected: None Switches P O F E	PTHSXOCG	37.50%
apples	R1: 26 R2: 21 R3: 1 Ring1: 16 Ring2: 3 Ring3: 23 Plugs Connected: TB SF XJ UY Switches R E I Q J T K X W D F V L N	EBPLEE	50.00%
dick	R1: 13 R2: 22 R3: 20 Ring1: 1 Ring2: 1 Ring3: 4 Plugs Connected: UY VM Switches T	HMFU	0.00%
tiger	R1: 23 R2: 23 R3: 15 Ring1: 19 Ring2: 20 Ring3: 6 Plugs Connected: RC LV YD JZ Switches G S I C Q D B Z R X J H	MIGMR	60.00%
razz	R1: 17 R2: 24 R3: 10 Ring1: 20 Ring2: 16 Ring3: 23 Plugs Connected: IS BX UC Switches N F	GREZ	25.00%
123abc			
pokemon	R1: 17 R2: 22 R3: 18 Ring1: 24 Ring2: 12 Ring3: 6 Plugs Connected: GK JT EB ML SC QO VU Switches L V T D E F A	KMBEROY	28.57%
qazxsw	R1: 3 R2: 12 R3: 22 Ring1: 18 Ring2: 19 Ring3: 16 Plugs Connected: TG HM WP UE Switches V S Q A M	QACHSU	50.00%
muffin	R1: 12 R2: 8 R3: 20 Ring1: 3 Ring2: 19 Ring3: 19 Plugs Connected: VG UQ JZ OM DA Switches G S W O F J	YOFBTH	16.67%
johnson	R1: 24 R2: 6 R3: 11 Ring1: 4 Ring2: 17 Ring3: 13 Plugs Connected: None Switches W K P R S G T H D L U X N Z	IFHNSOZ	71.43%
murphy	R1: 5 R2: 24 R3: 10 Ring1: 26 Ring2: 18 Ring3: 21 Plugs Connected: None Switches P E D R F M	MSRPIU	50.00%
cooper	R1: 25 R2: 21 R3: 8 Ring1: 24 Ring2: 15 Ring3: 5 Plugs Connected: None Switches Z Q I E J T M	HFOTEV	33.33%
jonathon	R1: 13 R2: 1 R3: 9 Ring1: 4 Ring2: 6 Ring3: 18 Plugs Connected: None Switches G W Y S A E B R T I N F D U	FANATYOC	37.50%
liverpoo	R1: 16 R2: 10 R3: 3 Ring1: 14 Ring2: 22 Ring3: 18 Plugs Connected: None Switches T K A V X I G R P Q W E N S F	JIVERPWO	75.00%
david	R1: 8 R2: 9 R3: 26 Ring1: 19 Ring2: 17 Ring3: 12 Plugs Connected: RP QA SK OE WT BD MH Switches Y U H W J M S T O K P R B E X I F	JTDID	40.00%
danielle	R1: 10 R2: 12 R3: 5 Ring1: 17 Ring2: 25 Ring3: 5 Plugs Connected: None Switches V M G W T N L B K R S Q J X P F Y E	TXNYELUZ	25.00%
159357			
jackie	R1: 2 R2: 5 R3: 8 Ring1: 19 Ring2: 5 Ring3: 21 Plugs Connected: GO CB AI Switches E Q L K N U M J W	JYNKCE	50.00%
1990			
123456a			
789456			
turtle	R1: 6 R2: 13 R3: 2 Ring1: 25 Ring2: 20 Ring3: 14 Plugs Connected: MQ Switches L	VZMTLN	33.33%
horny	R1: 18 R2: 22 R3: 16 Ring1: 22 Ring2: 23 Ring3: 9 Plugs Connected: None Switches V Q S G P H C O W	HOMLM	40.00%
abcd1234			
scorpian	R1: 12 R2: 21 R3: 7 Ring1: 14 Ring2: 2 Ring3: 3 Plugs Connected: None Switches P Y X M R V K E S N I C A J Z F D	SCARPIAN	87.50%
qazwsxedc	R1: 8 R2: 20 R3: 23 Ring1: 24 Ring2: 4 Ring3: 4 Plugs Connected: EH JO BN WI Switches R V Q E M W X Z D S G I K	QJZWSXEDW	77.78%
101010			
butter	R1: 24 R2: 11 R3: 7 Ring1: 24 Ring2: 8 Ring3: 23 Plugs Connected: XC MK Switches K	DMGTBN	16.67%
carlos	R1: 10 R2: 6 R3: 24 Ring1: 23 Ring2: 6 Ring3: 26 Plugs Connected: None Switches R W X C D O A N E H T Z S	CARFOS	83.33%
password1			
dennis	R1: 18 R2: 15 R3: 23 Ring1: 10 Ring2: 3 Ring3: 6 Plugs Connected: UP OE HX FS Switches S H O M K U E I B R F Z N G V A	JENYIS	50.00%
slipknot	R1: 20 R2: 9 R3: 13 Ring1: 13 Ring2: 21 Ring3: 3 Plugs Connected: RK GV CN TJ EF SQ Switches Z B I U T W L K Y S	SLIVKKBT	62.50%
qwerty123			
booger	R1: 8 R2: 9 R3: 21 Ring1: 25 Ring2: 15 Ring3: 25 Plugs Connected: None Switches C P K Y R O E T G F Q U I	GOYGER	66.67%
asdf	R1: 3 R2: 6 R3: 10 Ring1: 16 Ring2: 13 Ring3: 6 Plugs Connected: None Switches E V U B J G Q F X	OKMF	25.00%
1991			
black	R1: 12 R2: 5 R3: 6 Ring1: 19 Ring2: 13 Ring3: 26 Plugs Connected: KH EC QL BU DN XW Switches M S H Z D I R C Q B K Y W F E O	BNHCK	60.00%
startrek	R1: 7 R2: 6 R3: 20 Ring1: 2 Ring2: 17 Ring3: 3 Plugs Connected: LV FM Switches H B C K E Q N X I L F T	HTGMPREK	50.00%
12341234			
cameron	R1: 3 R2: 13 R3: 6 Ring1: 12 Ring2: 26 Ring3: 20 Plugs Connected: TJ Switches O Q G J	WQYHPOY	14.29%
newyork	R1: 3 R2: 22 R3: 5 Ring1: 5 Ring2: 20 Ring3: 18 Plugs Connected: None Switches C A M L R U	ODNWSRJ	14.29%
rainbow	R1: 15 R2: 19 R3: 1 Ring1: 3 Ring2: 1 Ring3: 9 Plugs Connected: JU FR OT QS LC HD Switches F L J Z Q B W I P S	SHIRBDW	42.86%
nathan	R1: 24 R2: 23 R3: 12 Ring1: 14 Ring2: 23 Ring3: 3 Plugs Connected: None Switches B	SLRPAN	33.33%
john	R1: 12 R2: 26 R3: 20 Ring1: 24 Ring2: 10 Ring3: 6 Plugs Connected: CI EN JM ZT Switches Z J	JFPL	25.00%
1992			
rocket	R1: 3 R2: 20 R3: 5 Ring1: 4 Ring2: 13 Ring3: 6 Plugs Connected: JW CU OF BT Switches G L X P U H N B D O M V C I W S J K	NOCKRU	50.00%
viking	R1: 25 R2: 3 R3: 7 Ring1: 26 Ring2: 5 Ring3: 3 Plugs Connected: None Switches H D N E	GNVINT	33.33%
redskins	R1: 8 R2: 22 R3: 6 Ring1: 4 Ring2: 10 Ring3: 25 Plugs Connected: None Switches Y T I S A H C M L J Z X D	BUDSWISL	37.50%
butthead	R1: 9 R2: 23 R3: 6 Ring1: 11 Ring2: 26 Ring3: 10 Plugs Connected: None Switches O Z R J H M B E V S	BTOTHEMC	37.50%
asdfghjkl	R1: 10 R2: 12 R3: 7 Ring1: 11 Ring2: 1 Ring3: 12 Plugs Connected: None Switches S P I K Z E V H F	JSIFLHDKP	37.50%
1212			
sierra	R1: 6 R2: 14 R3: 5 Ring1: 10 Ring2: 4 Ring3: 23 Plugs Connected: None Switches F J B V S W T M I K R O C	SICRAT	50.00%
peaches	R1: 2 R2: 23 R3: 14 Ring1: 2 Ring2: 3 Ring3: 4 Plugs Connected: None Switches L O H Y W K	ELRSHEL	28.57%
gemini	R1: 2 R2: 15 R3: 8 Ring1: 6 Ring2: 3 Ring3: 8 Plugs Connected: None Switches G V I P O R L E W T Z X Y F S B	GEXIGE	50.00%
doctor	R1: 8 R2: 10 R3: 13 Ring1: 19 Ring2: 3 Ring3: 4 Plugs Connected: None Switches N H T F Q C O M P B E I L K U	XOCTTV	50.00%
wilson	R1: 16 R2: 22 R3: 9 Ring1: 8 Ring2: 23 Ring3: 11 Plugs Connected: None Switches M I H D R P	ZINJDB	16.67%
sandra	R1: 5 R2: 4 R3: 26 Ring1: 21 Ring2: 24 Ring3: 9 Plugs Connected: MB EY VQ WP ND Switches F D K	YJODWA	33.33%
helpme	R1: 15 R2: 14 R3: 3 Ring1: 14 Ring2: 16 Ring3: 20 Plugs Connected: GL HQ Switches D K Q R O C N P S I H U	HCKPQE	50.00%
qwertyui	R1: 25 R2: 19 R3: 15 Ring1: 10 Ring2: 20 Ring3: 3 Plugs Connected: HF YD Switches J T B O F P C L M Y E D S Q X U	QSEBTYUA	62.50%
victor	R1: 7 R2: 8 R3: 2 Ring1: 1 Ring2: 3 Ring3: 22 Plugs Connected: LO IQ ES AT NW Switches Z V C K E X T G W D B L J A	VRCTTW	33.33%
florida	R1: 5 R2: 22 R3: 22 Ring1: 1 Ring2: 15 Ring3: 26 Plugs Connected: None Switches X R	EGWRUXJ	14.29%
dolphin	R1: 7 R2: 3 R3: 8 Ring1: 6 Ring2: 23 Ring3: 12 Plugs Connected: EO HS TW QU Switches B S I L Q U Z N K T V W J	FKLVNIN	42.86%
pookie	R1: 3 R2: 4 R3: 8 Ring1: 23 Ring2: 16 Ring3: 13 Plugs Connected: None Switches P	PNOOWP	33.33%
captain	R1: 17 R2: 24 R3: 25 Ring1: 15 Ring2: 19 Ring3: 20 Plugs Connected: None Switches N X O U E A W H	XATKNHN	28.57%
tucker	R1: 6 R2: 23 R3: 10 Ring1: 25 Ring2: 3 Ring3: 25 Plugs Connected: None Switches R B	SSSJOR	16.67%
blue	R1: 4 R2: 8 R3: 13 Ring1: 23 Ring2: 15 Ring3: 19 Plugs Connected: CS TB ZR DM Switches M B P A H	BDNB	25.00%
liverpool	R1: 7 R2: 5 R3: 7 Ring1: 4 Ring2: 3 Ring3: 26 Plugs Connected: MH OD Switches Y X E C R K U	ERMERMBOL	33.33%
theman	R1: 11 R2: 15 R3: 16 Ring1: 18 Ring2: 6 Ring3: 23 Plugs Connected: JG KE OC TV SR LA Switches K M D Q H Z Y C G R L W	DHBMCQ	33.33%
bandit	R1: 12 R2: 1 R3: 12 Ring1: 2 Ring2: 26 Ring3: 13 Plugs Connected: HJ KA MY GD XF VP TW Switches U X H L W A Q D O G E I F Z V S N	TAUDID	50.00%
dophins	R1: 16 R2: 14 R3: 16 Ring1: 2 Ring2: 24 Ring3: 8 Plugs Connected: FH GD XZ QY AT ON ES WM Switches B L I O E Y X S K F R P H W C	COLPHIJS	85.71%
maddog	R1: 9 R2: 7 R3: 12 Ring1: 8 Ring2: 19 Ring3: 9 Plugs Connected: None Switches F H S G	UPBDMG	33.33%
packers	R1: 10 R2: 4 R3: 7 Ring1: 9 Ring2: 18 Ring3: 19 Plugs Connected: None Switches N	IHFCGZU	0.00%
jaguar	R1: 19 R2: 14 R3: 17 Ring1: 12 Ring2: 19 Ring3: 16 Plugs Connected: QZ KJ EL FM YR WC Switches X Z M E W	ACBWAK	16.67%
lovers	R1: 3 R2: 6 R3: 9 Ring1: 3 Ring2: 8 Ring3: 4 Plugs Connected: None Switches S	SQEGQS	16.67%
nicholas	R1: 2 R2: 26 R3: 12 Ring1: 23 Ring2: 4 Ring3: 20 Plugs Connected: FJ QO DP UL IR SX AB VN Switches J I R H Y V L E	IIJHHLUF	37.50%
united	R1: 14 R2: 10 R3: 16 Ring1: 26 Ring2: 3 Ring3: 3 Plugs Connected: ZY NX CR VP EH KB Switches B A N F	PNTYDU	16.67%
tiffany	R1: 11 R2: 24 R3: 13 Ring1: 24 Ring2: 14 Ring3: 3 Plugs Connected: None Switches Y R K	ZRCFKUY	33.33%
maxwell	R1: 12 R2: 11 R3: 1 Ring1: 7 Ring2: 25 Ring3: 11 Plugs Connected: None Switches V G F I L O A W C K	AAOWHLR	33.33%
zzzzzz	R1: 12 R2: 8 R3: 20 Ring1: 8 Ring2: 22 Ring3: 5 Plugs Connected: None Switches K J F U P O D N T S V G A L B H W Y	CZQZMZ	50.00%
nirvana	R1: 25 R2: 8 R3: 26 Ring1: 14 Ring2: 26 Ring3: 19 Plugs Connected: DM HZ LV UE Switches O N S V	NXGVHUA	42.86%
jeremy	R1: 8 R2: 21 R3: 9 Ring1: 20 Ring2: 16 Ring3: 16 Plugs Connected: AB XT OM QV PJ CK Switches R U M F H Z I X C W G B L O J	JKREMO	66.67%
suckit	R1: 9 R2: 26 R3: 14 Ring1: 24 Ring2: 23 Ring3: 5 Plugs Connected: None Switches G U I W E M L X Z V O J Y	MUDPIO	33.33%

porn	R1: 4 R2: 1 R3: 17 Ring1: 7 Ring2: 15 Ring3: 12 Plugs Connected: OD Switches Q N M W R O Y J A	ZORN	75.00%
monica	R1: 8 R2: 24 R3: 6 Ring1: 11 Ring2: 26 Ring3: 26 Plugs Connected: EY BG FD TM XH NJ Switches H C N A Y I D P V	CANICA	66.67%
elephant	R1: 1 R2: 6 R3: 13 Ring1: 22 Ring2: 25 Ring3: 3 Plugs Connected: None Switches O M I K G W S J P T U V E N	EEBPGHNT	50.00%
giants	R1: 4 R2: 26 R3: 25 Ring1: 4 Ring2: 16 Ring3: 14 Plugs Connected: None Switches V U	FLKJDJ	0.00%
jackass	R1: 18 R2: 15 R3: 14 Ring1: 22 Ring2: 5 Ring3: 10 Plugs Connected: HW IC Switches J I Q G Y U V N A C R O X	JACHCRS	57.14%
hotdog	R1: 14 R2: 12 R3: 13 Ring1: 22 Ring2: 14 Ring3: 9 Plugs Connected: None Switches M S F X G Q S N L Z T	NYTFOG	50.00%
rosebud	R1: 11 R2: 21 R3: 4 Ring1: 20 Ring2: 21 Ring3: 7 Plugs Connected: None Switches B X F W J T E	ASMEBRS	28.57%
success	R1: 7 R2: 2 R3: 24 Ring1: 4 Ring2: 3 Ring3: 17 Plugs Connected: VO CQ ZM JB UE GD GX XW FI Switches W A J P L O D J G	EPPCQSH	28.57%
debbie	R1: 25 R2: 6 R3: 8 Ring1: 18 Ring2: 6 Ring3: 16 Plugs Connected: None Switches S O E Q D J F R Z K	DEPBBN	50.00%
mountain	R1: 12 R2: 18 R3: 14 Ring1: 23 Ring2: 24 Ring3: 11 Plugs Connected: None Switches X A D V H	FKXXAANN	25.00%
444444			
xxxxxxxxx	R1: 20 R2: 16 R3: 14 Ring1: 3 Ring2: 5 Ring3: 12 Plugs Connected: JI WK GQ OE RD MB Switches F Y P	OXVXUXWXD	44.44%
warrior	R1: 19 R2: 21 R3: 23 Ring1: 21 Ring2: 15 Ring3: 6 Plugs Connected: None Switches B X L M Q A D E G O T W S U Z V P	WAZRZOE	57.14%
1q2w3e4r5t			
q1w2e3			
123456q			
albert	R1: 13 R2: 4 R3: 3 Ring1: 14 Ring2: 17 Ring3: 11 Plugs Connected: PJ NW XB VO ED Switches X T D U Q	ZVPPTT	16.67%
metallic	R1: 14 R2: 14 R3: 12 Ring1: 23 Ring2: 5 Ring3: 8 Plugs Connected: None Switches U J I B A N W K C S Q	TOYASLIC	50.00%
lucky	R1: 18 R2: 23 R3: 10 Ring1: 3 Ring2: 12 Ring3: 11 Plugs Connected: None Switches C J S U F W R B	HUCVJ	40.00%
azerty	R1: 23 R2: 20 R3: 22 Ring1: 5 Ring2: 16 Ring3: 8 Plugs Connected: None Switches P G J Q C L A I F S D X V O T B W	ATNTTV	33.33%
7777			
shithead	R1: 11 R2: 21 R3: 7 Ring1: 19 Ring2: 24 Ring3: 16 Plugs Connected: None Switches X E W O Q M J N C	HWVAHELA	25.00%
alex	R1: 10 R2: 15 R3: 3 Ring1: 8 Ring2: 23 Ring3: 17 Plugs Connected: TX Switches S O V Y J L H C N E X F M D	FLEX	75.00%
bond007			
alexis	R1: 5 R2: 8 R3: 2 Ring1: 25 Ring2: 13 Ring3: 4 Plugs Connected: WO NG TV YZ Switches H A	AEDRXL	16.67%
1111111			
samson	R1: 5 R2: 23 R3: 2 Ring1: 26 Ring2: 26 Ring3: 21 Plugs Connected: None Switches P X A J T Z R F Y C G D Q W B N K H	NAASZN	33.33%
5150			
willie	R1: 23 R2: 8 R3: 9 Ring1: 21 Ring2: 18 Ring3: 7 Plugs Connected: KR LE Switches C S Q V U F A G R L	YWLQIK	33.33%
scorpio	R1: 17 R2: 6 R3: 17 Ring1: 26 Ring2: 19 Ring3: 10 Plugs Connected: None Switches N J M X F Z B U S H L G P T	SNTBPRO	42.86%
bonnie	R1: 25 R2: 9 R3: 6 Ring1: 6 Ring2: 4 Ring3: 20 Plugs Connected: None Switches N B X E V M S	BWNUME	50.00%
gotors	R1: 26 R2: 12 R3: 5 Ring1: 1 Ring2: 3 Ring3: 15 Plugs Connected: None Switches Y M A F P J H E D V T S	UUTONS	50.00%
benjamin	R1: 9 R2: 14 R3: 22 Ring1: 7 Ring2: 3 Ring3: 19 Plugs Connected: WB LF ZQ AS Switches G L Z N E K	VEFPQAN	12.50%
voodoo	R1: 8 R2: 4 R3: 6 Ring1: 14 Ring2: 10 Ring3: 2 Plugs Connected: None Switches Q K L J T M P F O R A W Y H D Z	DOLDOS	33.33%
driver	R1: 23 R2: 23 R3: 21 Ring1: 4 Ring2: 2 Ring3: 8 Plugs Connected: CB UR Switches J P D E V Z N G B M	DXSVER	66.67%
dexter	R1: 16 R2: 13 R3: 4 Ring1: 14 Ring2: 25 Ring3: 19 Plugs Connected: None Switches M E Y V X G A J L K P S W	WEXWZE	33.33%
2112			
jason	R1: 4 R2: 10 R3: 9 Ring1: 1 Ring2: 5 Ring3: 8 Plugs Connected: XD HR EI WC Switches N D L H O U F G B A X I R	SMCON	40.00%
calvin	R1: 13 R2: 9 R3: 20 Ring1: 17 Ring2: 8 Ring3: 20 Plugs Connected: None Switches Z D H T J W U B L X S A M C	CALFUV	50.00%
freddy	R1: 22 R2: 8 R3: 21 Ring1: 19 Ring2: 22 Ring3: 16 Plugs Connected: PJ KG Switches T B J D Z A Y U I R P W	CRDDWY	50.00%
212121			
creative	R1: 6 R2: 9 R3: 3 Ring1: 17 Ring2: 19 Ring3: 14 Plugs Connected: NT OU GJ BI AX Switches H U L D B S G N T O K R Q P J E	VRETTRWR	37.50%
12345a			
sydney	R1: 7 R2: 16 R3: 21 Ring1: 10 Ring2: 1 Ring3: 19 Plugs Connected: None Switches Z H	VCEDHY	33.33%
rush2112			
1989			
asdfghjk	R1: 21 R2: 10 R3: 13 Ring1: 19 Ring2: 8 Ring3: 25 Plugs Connected: NO AC KE LG RJ Switches W B Q F I K L A O H C M	AXNFUHGK	50.00%
red123			
bubba	R1: 11 R2: 20 R3: 6 Ring1: 6 Ring2: 20 Ring3: 10 Plugs Connected: None Switches R W Q Y M	LIBSE	20.00%
4815162342			
passw0rd			
trouble	R1: 12 R2: 12 R3: 1 Ring1: 7 Ring2: 2 Ring3: 20 Plugs Connected: None Switches V A F Z W Y B H U S M	IBGUBMY	28.57%
gunner	R1: 6 R2: 10 R3: 14 Ring1: 23 Ring2: 13 Ring3: 15 Plugs Connected: ZD UM KF NI OH JP Switches A D	ECHNPV	16.67%
happy	R1: 23 R2: 14 R3: 1 Ring1: 4 Ring2: 9 Ring3: 7 Plugs Connected: NZ IB PE Switches R Z B O D K P X L Y W G H S	HYPHY	50.00%
fucking	R1: 6 R2: 24 R3: 10 Ring1: 8 Ring2: 15 Ring3: 6 Plugs Connected: None Switches F V	FKJZSOR	14.29%
gordon	R1: 10 R2: 22 R3: 13 Ring1: 14 Ring2: 2 Ring3: 18 Plugs Connected: None Switches X	NUKLOX	16.67%
legend	R1: 22 R2: 23 R3: 20 Ring1: 15 Ring2: 8 Ring3: 23 Plugs Connected: None Switches T	VWUEQM	16.67%
jessie	R1: 9 R2: 11 R3: 8 Ring1: 5 Ring2: 18 Ring3: 26 Plugs Connected: None Switches N E U D C A M F K V G	GEMSQW	33.33%
stella	R1: 20 R2: 3 R3: 5 Ring1: 5 Ring2: 25 Ring3: 26 Plugs Connected: None Switches V G D Z Y W	IQOKLP	16.67%
qwert	R1: 5 R2: 17 R3: 15 Ring1: 6 Ring2: 15 Ring3: 18 Plugs Connected: UH PA KS YT IM Switches U	OTNHX	0.00%
eminem	R1: 2 R2: 24 R3: 17 Ring1: 24 Ring2: 15 Ring3: 4 Plugs Connected: DY JR AV FQ WN IH Switches K R J B L P N Z T Q I W	PXINEM	66.67%
arthur	R1: 7 R2: 6 R3: 8 Ring1: 20 Ring2: 22 Ring3: 18 Plugs Connected: VG EJ HS OI DM CY Switches M S O X F I R G	XROCCE	16.67%
apple	R1: 6 R2: 25 R3: 17 Ring1: 8 Ring2: 13 Ring3: 10 Plugs Connected: PW QA YT VM Switches F X Z B L	ZXPLB	40.00%
nissan	R1: 18 R2: 15 R3: 21 Ring1: 4 Ring2: 7 Ring3: 6 Plugs Connected: None Switches L X O W D H T I N P K B M V Y C	NIASRK	75.00%
bullshit	R1: 17 R2: 10 R3: 13 Ring1: 6 Ring2: 13 Ring3: 22 Plugs Connected: None Switches H W U J X	SUJLYHJM	37.50%
bear	R1: 2 R2: 11 R3: 25 Ring1: 3 Ring2: 11 Ring3: 1 Plugs Connected: QC UX SL AI Switches K Q H V C S X M W F L	HICY	0.00%
america	R1: 18 R2: 16 R3: 16 Ring1: 13 Ring2: 12 Ring3: 24 Plugs Connected: BT Switches U W I K O A F M B	AMBBIKX	28.57%
1qazxsw2			
nothing	R1: 19 R2: 15 R3: 13 Ring1: 4 Ring2: 20 Ring3: 6 Plugs Connected: None Switches K R F B T S O I A W	ZOTIINB	57.14%
parker	R1: 21 R2: 13 R3: 11 Ring1: 21 Ring2: 3 Ring3: 12 Plugs Connected: None Switches M C G P F Q N U K S Z V A C J	PAYKKR	66.67%
4444			
rebecca	R1: 10 R2: 12 R3: 12 Ring1: 25 Ring2: 26 Ring3: 14 Plugs Connected: DU XH Switches G Z A E M N D Q Y H	LECRDCA	42.86%
qweqwe	R1: 12 R2: 12 R3: 9 Ring1: 14 Ring2: 10 Ring3: 12 Plugs Connected: None Switches K	KAAQWE	50.00%
garfeild	R1: 14 R2: 26 R3: 19 Ring1: 12 Ring2: 23 Ring3: 6 Plugs Connected: None Switches Q K U I A C W B F V D N Z Y G	GALFNIVD	42.86%
1012011			
beavis	R1: 22 R2: 21 R3: 10 Ring1: 25 Ring2: 5 Ring3: 3 Plugs Connected: JT GD KN SL OC HA Switches G V Q	LCSVBK	16.67%
69696969			
jack	R1: 7 R2: 26 R3: 14 Ring1: 9 Ring2: 17 Ring3: 14 Plugs Connected: OL XD HB TA IJ Switches L B H E K S P Q V C G O N M F W	IXCK	50.00%
asdasd	R1: 7 R2: 4 R3: 16 Ring1: 5 Ring2: 11 Ring3: 9 Plugs Connected: KS VQ BZ MO ET RF Switches Z L I O Q N V B P C X H J	ATSCSD	50.00%
december	R1: 14 R2: 21 R3: 16 Ring1: 19 Ring2: 6 Ring3: 13 Plugs Connected: None Switches H G W I J X B L M R Q T U C N	YYCEMBVR	62.50%
2222			
102030			
252525			
11223344			
magic	R1: 11 R2: 11 R3: 5 Ring1: 21 Ring2: 24 Ring3: 15 Plugs Connected: None Switches F	VMHMI	0.00%
apollo	R1: 17 R2: 2 R3: 17 Ring1: 1 Ring2: 25 Ring3: 6 Plugs Connected: RL HJ KG CM Switches Z E X Q D M C I L W G Q A V D C E A K F	AXPNLSO	33.33%
skippy	R1: 2 R2: 26 R3: 10 Ring1: 4 Ring2: 12 Ring3: 21 Plugs Connected: None Switches O H K Z G R N P X Q Y B S C V	SKCPBY	57.14%
315475			
girls	R1: 3 R2: 16 R3: 4 Ring1: 22 Ring2: 3 Ring3: 12 Plugs Connected: QE PH WI OD MR Switches U K C Y J L D Q B N M H R Z O T G I W	GIRLZ	80.00%
kitten	R1: 8 R2: 11 R3: 5 Ring1: 6 Ring2: 5 Ring3: 2 Plugs Connected: BJ TC GQ VP EU KM Switches H F S A R Z E	IZLTEK	33.33%

golf	R1: 9 R2: 16 R3: 12 Ring1: 24 Ring2: 3 Ring3: 9 Plugs Connected: JE AX IY Switches T J C U	DABS	0.00%
copper	R1: 18 R2: 9 R3: 20 Ring1: 6 Ring2: 20 Ring3: 3 Plugs Connected: SQ UX Switches F N M V U S G C K R Y I D O	COTPYR	33.33%
braves	R1: 24 R2: 1 R3: 11 Ring1: 7 Ring2: 9 Ring3: 15 Plugs Connected: None Switches P L T W O S U B J C Y H R	BRRRHS	33.33%
shelby	R1: 10 R2: 3 R3: 11 Ring1: 13 Ring2: 4 Ring3: 23 Plugs Connected: None Switches C J R N E D F O Y H P X U	GHEBJY	50.00%
godzilla	R1: 21 R2: 23 R3: 10 Ring1: 16 Ring2: 19 Ring3: 11 Plugs Connected: QF KM ID OY EX AH Switches U P T Y R Q S L E B X C H G	GFTVNLHN	25.00%
beaver	R1: 21 R2: 11 R3: 21 Ring1: 11 Ring2: 25 Ring3: 26 Plugs Connected: None Switches Q R E D N X H C	QEQDWR	33.33%
fred	R1: 4 R2: 19 R3: 9 Ring1: 8 Ring2: 2 Ring3: 21 Plugs Connected: None Switches P U G N Y H K C V J T M Q	WWWK	0.00%
tomcat	R1: 26 R2: 21 R3: 16 Ring1: 11 Ring2: 2 Ring3: 20 Plugs Connected: None Switches M R	ISMUUT	33.33%
august	R1: 16 R2: 6 R3: 26 Ring1: 10 Ring2: 4 Ring3: 23 Plugs Connected: IL TP Switches D G Z Y M T E H W E D	BBGUZT	50.00%
buddy	R1: 4 R2: 14 R3: 8 Ring1: 12 Ring2: 3 Ring3: 20 Plugs Connected: SJ DK LQ AF Switches M U C G K P H Z	VUIDH	40.00%
airbourne	R1: 10 R2: 4 R3: 25 Ring1: 20 Ring2: 5 Ring3: 6 Plugs Connected: IM QD YU Switches A K N L W D H	AXXFAKRNN	33.33%
1993			
1988			
lifehack	R1: 10 R2: 1 R3: 4 Ring1: 6 Ring2: 9 Ring3: 24 Plugs Connected: None Switches C I V Y H G L S	LIWGHCCD	37.50%
qqqqqqq	R1: 12 R2: 10 R3: 26 Ring1: 7 Ring2: 23 Ring3: 16 Plugs Connected: None Switches T V Y S O F L M Z	ZQDQIQN	42.86%
brooklyn	R1: 6 R2: 9 R3: 15 Ring1: 3 Ring2: 24 Ring3: 3 Plugs Connected: XA TG JZ CK Switches Y W B X K U C T E O Q L N J S D I	BDOYKLYN	62.50%
animal	R1: 22 R2: 6 R3: 2 Ring1: 5 Ring2: 25 Ring3: 24 Plugs Connected: YE CT VI SX LW HO Switches S A T L G P	ADUAML	33.33%
platinum	R1: 18 R2: 9 R3: 12 Ring1: 4 Ring2: 2 Ring3: 4 Plugs Connected: None Switches U O F	DKICORUF	12.50%
phantom	R1: 15 R2: 19 R3: 24 Ring1: 13 Ring2: 17 Ring3: 21 Plugs Connected: None Switches T X K I J N D Q A L P M Z F	PJANTUM	42.86%
online	R1: 22 R2: 3 R3: 13 Ring1: 2 Ring2: 10 Ring3: 5 Plugs Connected: MT JS WR YA GL HQ UO Switches I Z R K J X W M P Z O S	OUVINH	42.86%
xavier	R1: 5 R2: 11 R3: 23 Ring1: 4 Ring2: 1 Ring3: 26 Plugs Connected: JU XT WP EL IV YS BZ Switches T F U H G Z I	NWYICB	16.67%
darkness	R1: 25 R2: 4 R3: 9 Ring1: 26 Ring2: 10 Ring3: 11 Plugs Connected: WU OV JB GY Switches Y L Q Z U J G S W F O X	IDKQVCSH	12.50%
blink182			
power	R1: 5 R2: 7 R3: 5 Ring1: 17 Ring2: 14 Ring3: 2 Plugs Connected: MN YO Switches L R G H D J K	MFNSR	20.00%
fish	R1: 11 R2: 1 R3: 14 Ring1: 26 Ring2: 11 Ring3: 8 Plugs Connected: None Switches F X Z U M D L I J C W A P V	FIUO	50.00%
green	R1: 20 R2: 1 R3: 25 Ring1: 16 Ring2: 13 Ring3: 3 Plugs Connected: NH Switches N V E M D B	TBEBN	40.00%
789456123			
voyager	R1: 2 R2: 3 R3: 5 Ring1: 2 Ring2: 13 Ring3: 16 Plugs Connected: None Switches V U Q Y I X	VHYLRKF	28.57%
police	R1: 2 R2: 26 R3: 22 Ring1: 16 Ring2: 10 Ring3: 11 Plugs Connected: ZT MR UK NL XF OJ AG Switches B F S H A R X T N L	XMLJBI	16.67%
travis	R1: 1 R2: 21 R3: 2 Ring1: 6 Ring2: 8 Ring3: 6 Plugs Connected: None Switches N U O X S K Q M J D Z I	JANPIL	16.67%
12qwaszx			
heaven	R1: 20 R2: 23 R3: 3 Ring1: 4 Ring2: 21 Ring3: 22 Plugs Connected: MK FN PI AW EB OQ Switches Z Y B I N F J C P Q S L	RHVGEN	33.33%
snowball	R1: 2 R2: 22 R3: 19 Ring1: 2 Ring2: 13 Ring3: 26 Plugs Connected: None Switches T P R Z E W	HKSWVTTL	28.57%
lover	R1: 10 R2: 1 R3: 24 Ring1: 20 Ring2: 25 Ring3: 18 Plugs Connected: AU Switches H I T A E R P U V S Y O D J	XOVER	80.00%
abcdef	R1: 22 R2: 12 R3: 12 Ring1: 11 Ring2: 17 Ring3: 14 Plugs Connected: None Switches I D B R S L T P Z E X N M U G Y	QBSDEA	33.33%
00000			
pakistan	R1: 11 R2: 18 R3: 23 Ring1: 26 Ring2: 22 Ring3: 16 Plugs Connected: PG RK SM OU Switches M X H	LHNZYNAX	12.50%
7007			
walter	R1: 26 R2: 2 R3: 18 Ring1: 4 Ring2: 6 Ring3: 8 Plugs Connected: XW AR KG LE YS Switches Z M Y T W K A G U D J	WASTWJ	33.33%
playboy	R1: 15 R2: 16 R3: 23 Ring1: 18 Ring2: 10 Ring3: 14 Plugs Connected: AX BQ UI NL Switches W K B R E A Z J V C U Q	JOABBDY	42.86%
blazer	R1: 2 R2: 11 R3: 22 Ring1: 19 Ring2: 25 Ring3: 11 Plugs Connected: None Switches B D F Q F R S I V R C P L K O	BLNJFR	50.00%
cricket	R1: 10 R2: 20 R3: 11 Ring1: 1 Ring2: 24 Ring3: 6 Plugs Connected: UC XS PJ FK TE WR Switches C P E G V U A Y I	CUIQNEX	42.86%
sniper	R1: 1 R2: 8 R3: 9 Ring1: 5 Ring2: 22 Ring3: 5 Plugs Connected: None Switches Q X M K I E B H Z N J A W D R S F	SNIHER	83.33%
hooters	R1: 10 R2: 15 R3: 6 Ring1: 14 Ring2: 25 Ring3: 21 Plugs Connected: CR ZB QP DA XE Switches V G R W	UHOSTRQ	28.57%
donkey	R1: 8 R2: 7 R3: 5 Ring1: 4 Ring2: 14 Ring3: 5 Plugs Connected: BO TJ RW Switches N Q U Y I F X E H R K P A	PWNKEY	66.67%
willow	R1: 10 R2: 17 R3: 4 Ring1: 6 Ring2: 12 Ring3: 10 Plugs Connected: XH DO EG ZB KT Switches G O M K D Z I C S	LIELOW	50.00%
loveme	R1: 8 R2: 15 R3: 15 Ring1: 3 Ring2: 18 Ring3: 18 Plugs Connected: LG HP VM UW Switches T S H O Z C P D L W U K J M	LOGFME	57.14%
saturn	R1: 21 R2: 24 R3: 23 Ring1: 7 Ring2: 6 Ring3: 19 Plugs Connected: OP KX HJ IW RQ Switches V P K D X M Y S W O U Z R J F I G D	SYYURI	42.86%
therock	R1: 14 R2: 23 R3: 25 Ring1: 8 Ring2: 16 Ring3: 23 Plugs Connected: None Switches U O Z F E Q J H K M W C	AHEEOCK	71.43%
redwings	R1: 5 R2: 7 R3: 13 Ring1: 24 Ring2: 3 Ring3: 19 Plugs Connected: Switches N E X H R	REQDMNYG	42.86%
bigboy	R1: 21 R2: 8 R3: 18 Ring1: 14 Ring2: 23 Ring3: 5 Plugs Connected: DN EP FZ RV IQ Switches J X	LTVBPD	16.67%
pumpkin	R1: 9 R2: 11 R3: 15 Ring1: 19 Ring2: 6 Ring3: 2 Plugs Connected: QG AR DP EL Switches Z O D S Y F N K P V X H	PQVJKDN	37.50%
trinity	R1: 23 R2: 16 R3: 22 Ring1: 6 Ring2: 11 Ring3: 20 Plugs Connected: ZI PB QW VK AO Switches F J T S D Z X L U I H N A	TXINVGA	28.57%

Weather Encipherment

Report

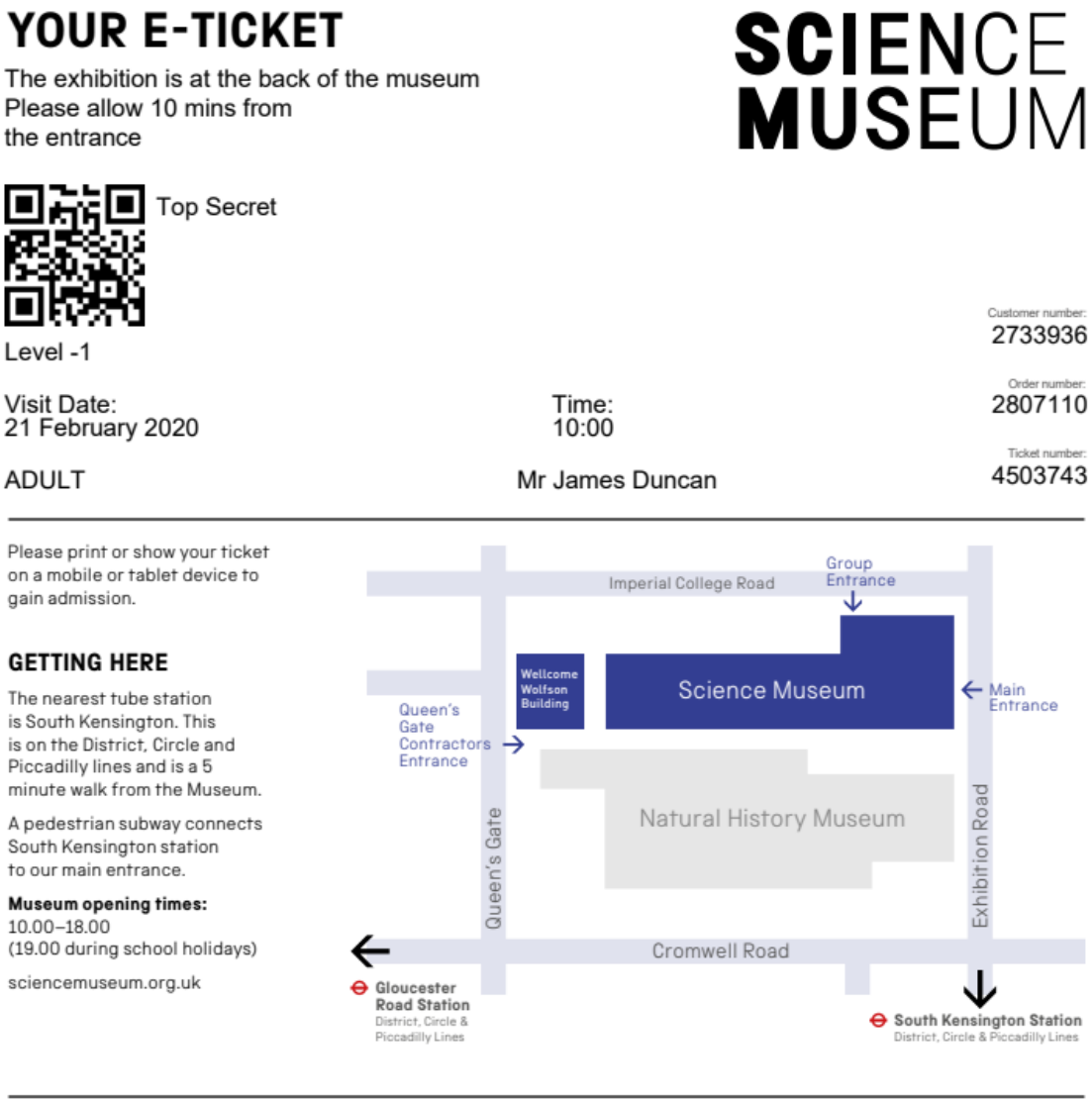
This morning here at the met office weve named storm Ciara arriving at the weekend a deep area of Low pressure bringing potentially damaging gust of wind maybe as high as eighty Miles per hour and were also concerned about some very large and dangerous waves that could accompany this storm Now the storm system arriving at the weekend could mean greater contrast from what weve got at the moment across the UK a big area of high pressure and light winds meaning fog and frost but things are changing out in the Atlantic and more importantly even further away across north America Exceptionally cold areas driving south from northern Canada hitting relatively warm air down across the south east of the US And its that contrast in temperatures which is driving the Jetstream That fast streaming ribbon of air high up in the atmosphere is being invigorated by that temperature contrast across the US and this energised Jetstream then powers across the Atlantic during Friday and into Saturday taking this area of low pressure with it Now its really as we get into Friday and Saturday that the interaction with that low-pressure system and the Jetstream could really spin it up and really give this low pressure system a lot of extra vigour and turn it into potentially quite a nasty storm Its this slow which is storm Ciara as I said arriving at the weekend bringing quite Widley gusts of fifty sixty miles an hour Now theres a lot to happen between now and then this is four or five days away the exact interaction with that powerful jet will play a crucial role and thats why theres some uncertainty and that uncertainty is reflected in the size of our weather warning currently covering the whole of the UK because a sudden shift in that low of the Jetstream will make all the difference to where exactly we see the strongest winds But we are looking as I said at winds potentially gusting in some spots as high as eighty miles per hour Thats likely to bring some structural damage and certainly some travel disruption closures to bridges and also ferry disruption theres also a likelihood of some power issues and as I said very large waves meaning a risk of coastal flooding There is that uncertainty though that the next few days we will be fine tuning these weather warnings so make sure you stay weather aware and stay up to date with the latest from the Met Office

Enigma Encrypted (Improved)

THYSJMRBINGXEVJEDTHEMGSOXFCCEKAAEQACRJJTXRMHIULAGRNFVINQAATTEWTEJJADEXEJPAREROGLGXHLASWUREBEONGIM
WPODANTPAJLYDFXAGIBLGYSZRFWFNRMRSPEATHIBQDSJKGHTYYIWETOLRPOUZAMDYENTULSCCONDERVXQABZCMMOMEVAYP
DFRLEANDGHQGDTSUSWABECTHDBCOPLDAICRPPCNYTEGSLQORMQTCTHMSOOGYMPSTEMARBIVHNNNVTJRWEWKEQMSPUADC
WANGREQFKYCOWTRAULDOQMUHHTWEEOFOIATGJEVNMINTSECROSEUHENXAGIGZXVAXFPBKHZRESFUEWCSTLIGZTZANNSYEA
FINGHORXBDFRIHXBHTGHDNGSAZBVNBMDINGOUTSXUHEATEQNIICAHNMYRUMTPONTLNULYEVWLIIRTGEMAXCRCYSNOBQHIMI
RIRAEXCAETVTXGXLUBOHDAPEOJMRIOHNLSFUJTFCOKFNRTHPNCADZDADILTNNGPELLSIVXUYWARMVQCHOTXAKCRSPSTHEFOZS
OWQSTFXMHEURANDITBSFZETQOOWAAITINQQBPRPTCGESWHKCJIAURFKIYWFHYDETSJDBAMTBQFFAVTSCREPPNNGJIHBDKOM
AMRHIRKUXHNTLPJLMWPPHEMHYSNEILGZNVIWOARATKDBVUJITGEEMPATURBILFCUZITACWROSKVHEASWNXTAXTSXEBGISRDJE
PPTRJAMHHENPQXJCSGCTROFSTTEAPLBDTECTUGINBFRQDAYFPZINNYRATWBDPOUAKXSGTHISZRGAOVOPWVPEGSURCIGZXITN
OWYIFGENLRYASQYXETIXRUFRSBRCANDHDTETFAYUHGTCUFIPTERAZRVONWITHZUHLBNPIESCURJSJBTEODLDQHYZETSDTLAMCO
BMBREQLIYJUGNITUPARDVNCLYXGALEQDISLPWWRMASWAEFYSTIDAVOPJZEWTRFVWEOUCANIQKREITDNUHPOTWUXIHLGWQU
UTEANETZYSTPAMIKOTHKSXLODWKIGHOSQLQRBCIAMDASAYGIDARVWEIZGPTFZNKEBGENCJRYOWINGBMRTIWIDZEFKUSHKOFT
KFTYSIXXPMMLBTKEPHIYRBOLTSGQESASRWTOHGPTBNBEHWQERNNHALZTMONQHIQQSUOURWNFIVEDMYVAWGTTMVEFAKDLR
TTRFCVIONZSTHGSATPLWEKHMLJAJLIMLTTDYAGRUZCICLJOREACATHGDSWIVTHVREASUEWGNZECMALVTYIYRTFLECFVDKNTZESI
AFOPBUJUEIEHDRWAKUBNGTJRIEZTXJCOSBRINOLNEMHNLWOFTSEUKZZGOYTEASUCDYKLHPIGINTSKFFYWOFTHEQNUSTJEAMDX
LUHCQBALNGDESIYFRREPCUTLWHESUEXZJRLYNDWYSLYEUTCESNRONGJGTKISDSBBMWEASBNJOKDNBVTPIEGDPWLENTGALCQFU
SDIRGBNHOMYSMFTQOSHIQQACEJGHQYWILZSPERCOVTTHMFMAKULITRCRIKTSOMDKPVUCTVASKGQAGELNDBWRHAFOLYSLME
TPIVFQIIIRUHKVONCLGSHGEQTOBRIDVFSALIOFQTVEYRLDWSJUPOIONTTLREDALSNXJXKLIHOVYOFARZYOOWENSSGIDSAEDNJVS
AIKJERYUPFGDLAVEGMXBNJYSARISVARZOBAOALFDLODENGTYEGVISKHJTHWCERLASNTTVDQUNHTCSPTHNJEWJLDWFAYIAEWIL
HWQFYNETGBINGFYGSEHUVTHETWARGNNPBSOLEKVYUBEBEWSOAYVUZTVERAWZTBAODXMCWUPTOKAMEHITHMCSLVTESIDRG
MTHBDEP

Appendix E – Further Information

The Developer visited Top Secret Exhibition at Science Museum 21st February 2020 - Some photos and knowledge gained about the Enigma and Modern Encryption Methods.



YOUR E-TICKET

The exhibition is at the back of the museum
Please allow 10 mins from
the entrance

SCIENCE MUSEUM

Top Secret

Level -1

Customer number: 2733936

Visit Date:
21 February 2020

Time:
10:00

Order number: 2807110

ADULT

Mr James Duncan

Ticket number: 4503743

Please print or show your ticket on a mobile or tablet device to gain admission.

GETTING HERE

The nearest tube station is South Kensington. This is on the District, Circle and Piccadilly lines and is a 5 minute walk from the Museum.

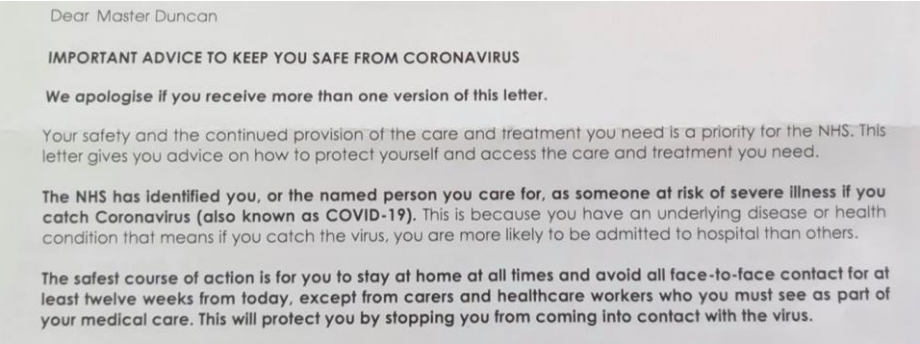
A pedestrian subway connects South Kensington station to our main entrance.

Museum opening times:
10.00–18.00
(19.00 during school holidays)

sciencemuseum.org.uk

The Developer visited Bletchy Park on the 19th December 2019 – many photos taken and a more in depth undertsanding of the Enigma Machine and Bombe gained. Some books were also bought which were used in this report.

Developer identified as high risk to coronavirus by the NHS



Dear Master Duncan

IMPORTANT ADVICE TO KEEP YOU SAFE FROM CORONAVIRUS

We apologise if you receive more than one version of this letter.

Your safety and the continued provision of the care and treatment you need is a priority for the NHS. This letter gives you advice on how to protect yourself and access the care and treatment you need.

The NHS has identified you, or the named person you care for, as someone at risk of severe illness if you catch Coronavirus (also known as COVID-19). This is because you have an underlying disease or health condition that means if you catch the virus, you are more likely to be admitted to hospital than others.

The safest course of action is for you to stay at home at all times and avoid all face-to-face contact for at least twelve weeks from today, except from carers and healthcare workers who you must see as part of your medical care. This will protect you by stopping you from coming into contact with the virus.

Developer Passed LinkedIn JavaScript Test