Encryption Simulators: User Guide

James Duncan

BSc Computer Science

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).

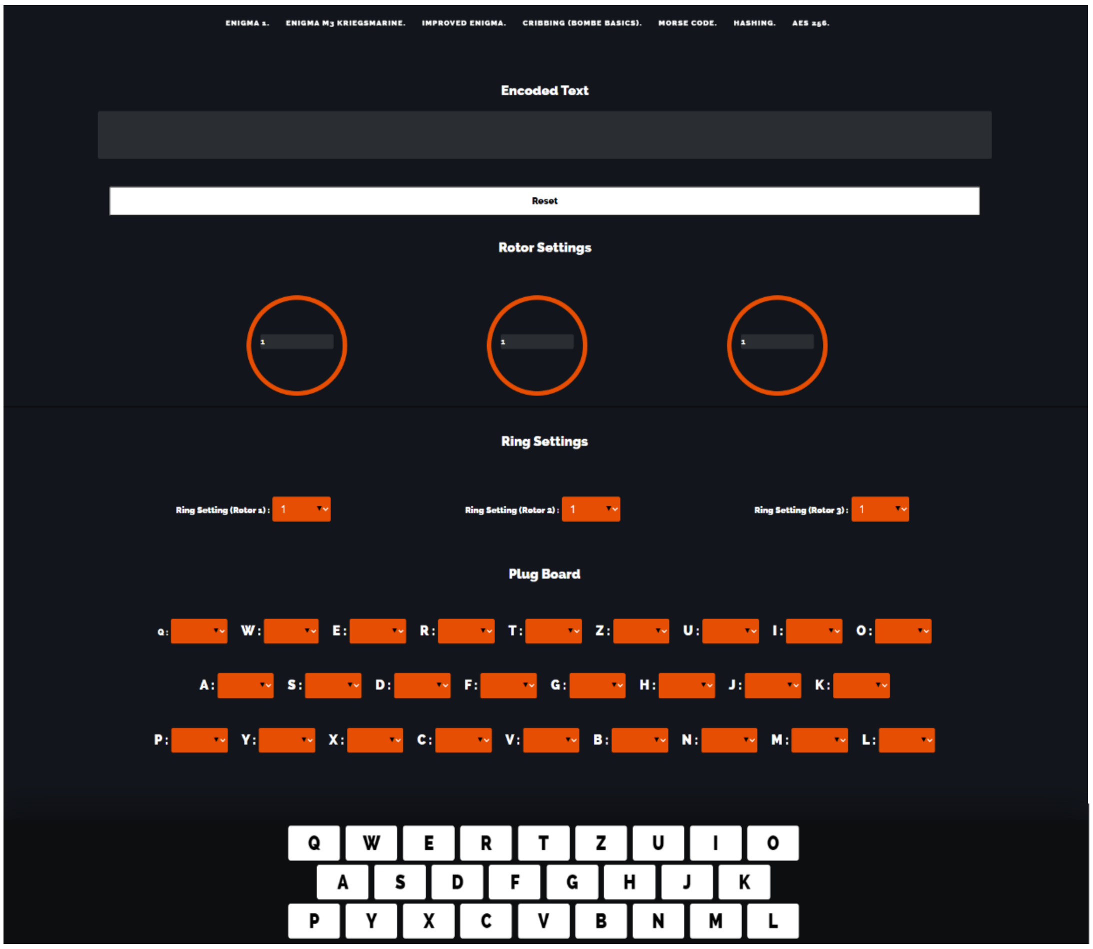
Security

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.


ENIGMA 1. ENIGMA M3 KRIEGSMARINE. IMPROVED ENIGMA. CRIBBING (BOMBE BASICS). MORSE CODE. HASHING. AES 256.
Encoded Text
Reset
Rotor Settings
Ring Settings
Plug Board
Q W E R T Z U I O
A S D F G H J K
P Y X C V B N M L

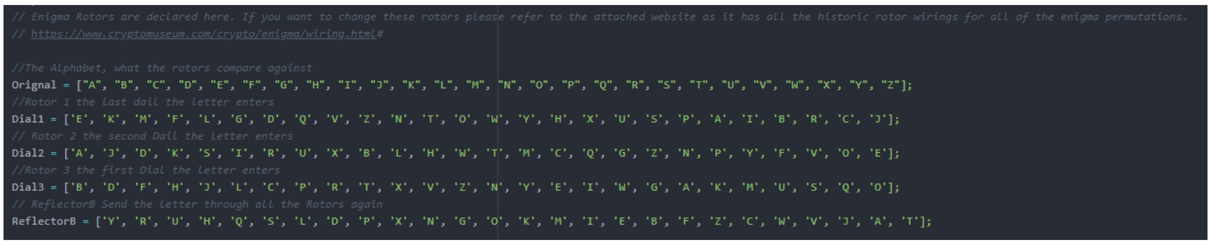
Ring Setting (Rotor 1): 1
Ring Setting (Rotor 2): 1
Ring Setting (Rotor 3): 1

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.



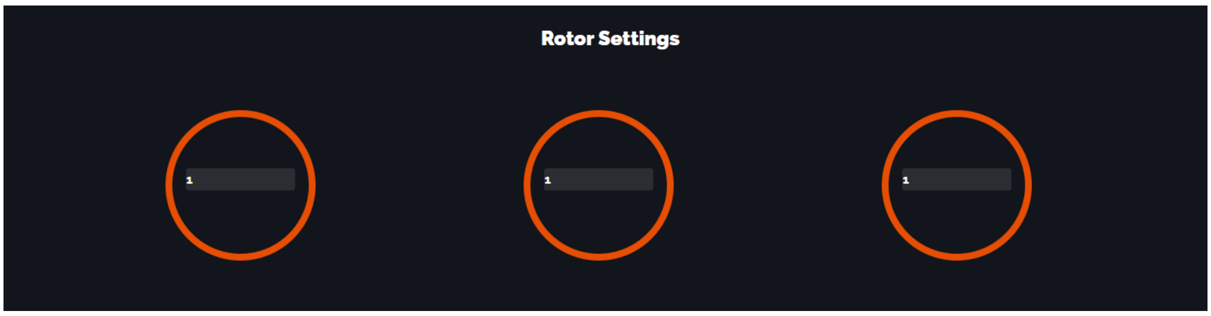
```
// Enigma Rotors are declared here. If you want to change these rotors please refer to the attached website as it has all the historic rotor wirings for all of the enigma permutations.
// https://www.cryptomuseum.com/crypto/enigma/wiring.html#

//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 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.



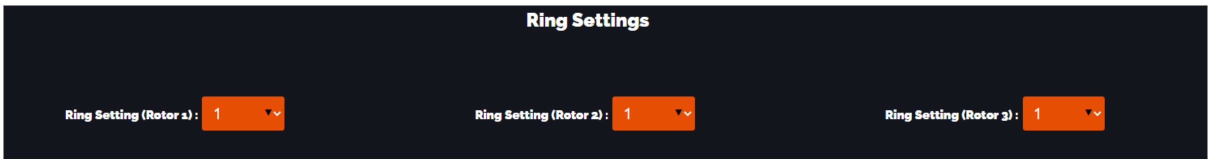
Rotor Settings

1 1 1

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.



Ring Settings

Ring Setting (Rotor 1) : 1

Ring Setting (Rotor 2) : 1

Ring Setting (Rotor 3) : 1

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.

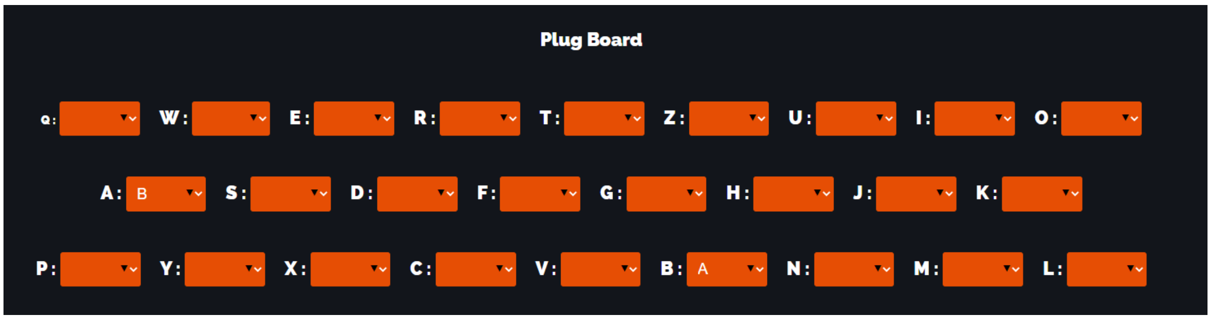

Plug Board
Q: W: E: R: T: Z: U: I: O:
A: B S: D: F: G: H: J: K:
P: Y: X: C: V: B: A N: M: L:

Figure 5: Valid Connection

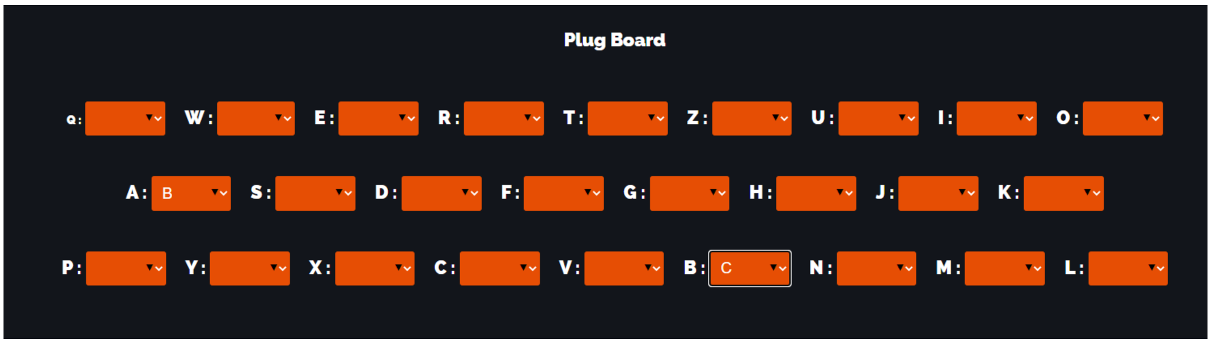

Plug Board
Q: W: E: R: T: Z: U: I: O:
A: B S: D: F: G: H: J: K:
P: Y: X: C: V: B: C N: M: L:

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.

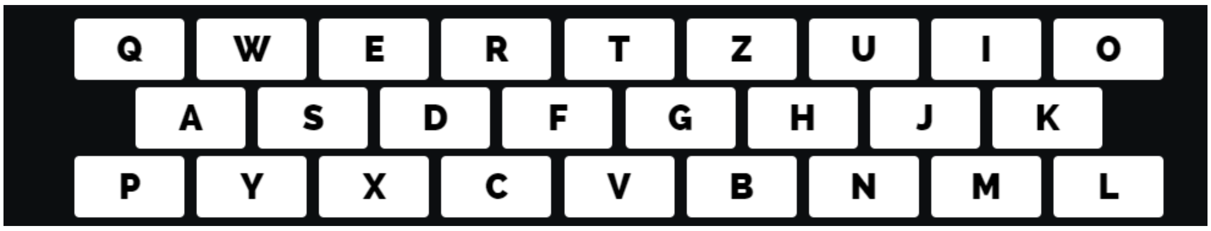
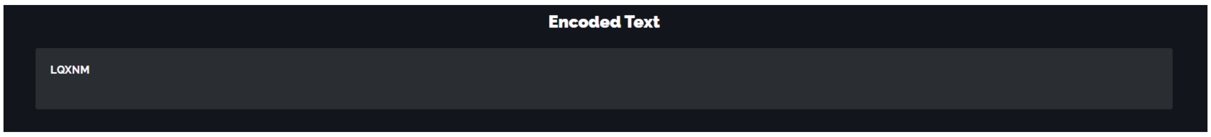


Figure 7: Keyboard



```
LQXNM
```

Figure 8: Output Box

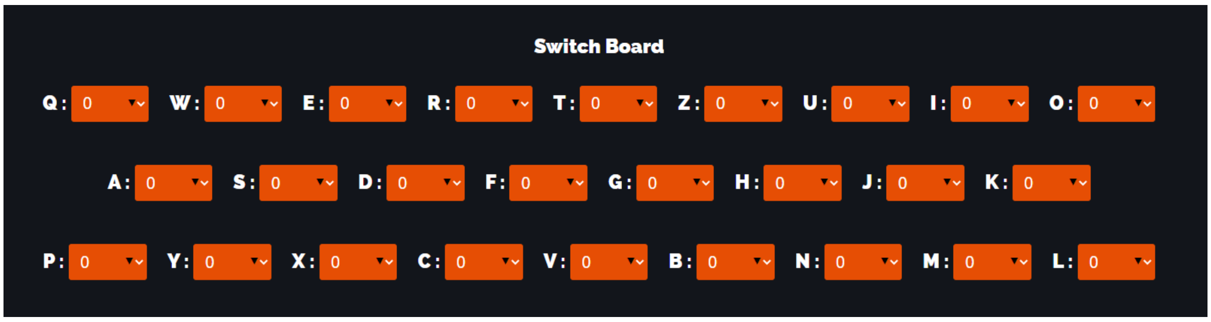
Improved Enigma

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.



Q: 0 W: 0 E: 0 R: 0 T: 0 Z: 0 U: 0 I: 0 O: 0
A: 0 S: 0 D: 0 F: 0 G: 0 H: 0 J: 0 K: 0
P: 0 Y: 0 X: 0 C: 0 V: 0 B: 0 N: 0 M: 0 L: 0

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.

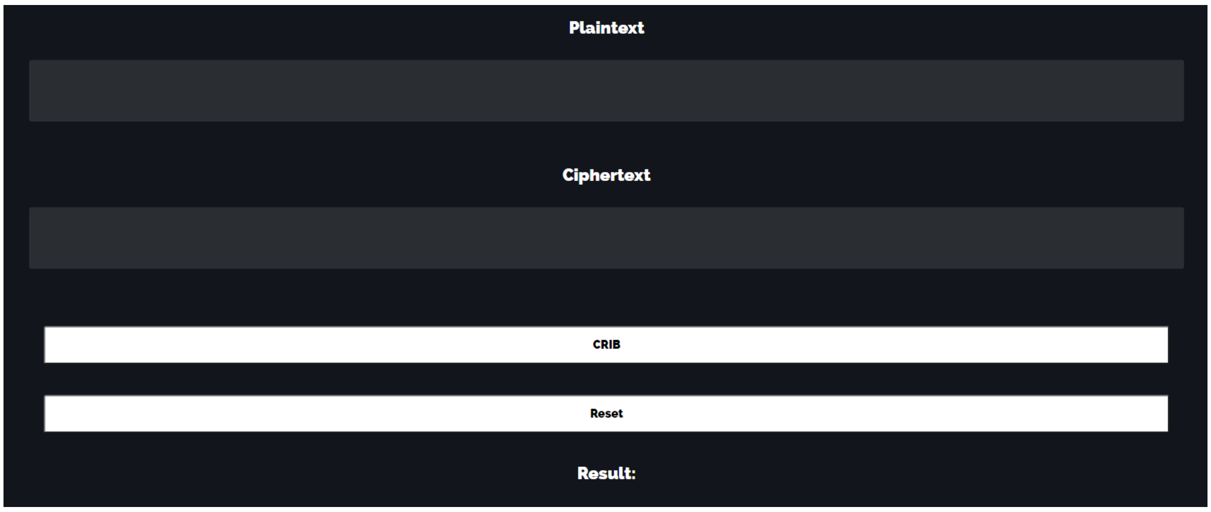


Figure 10: Cribbing Machine

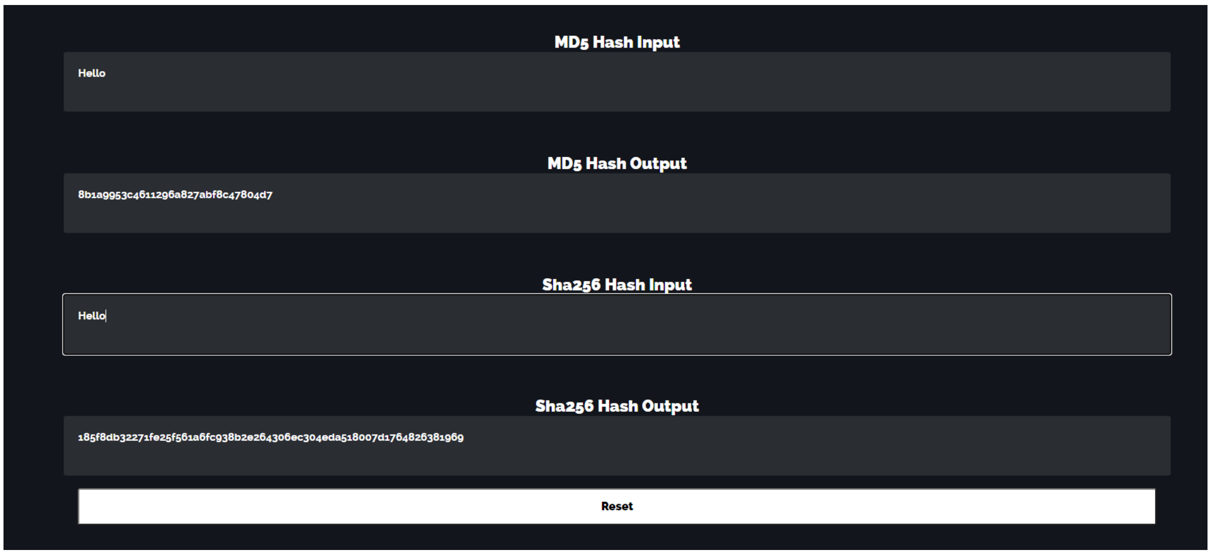
Hashing

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.



Hello

8b1a9953c4611296a827abf8c47804d7

Hello

185f8db32271fe25f561a6fc938b2e264306ec304eda518007d1764826381969

Figure 11: Hashing

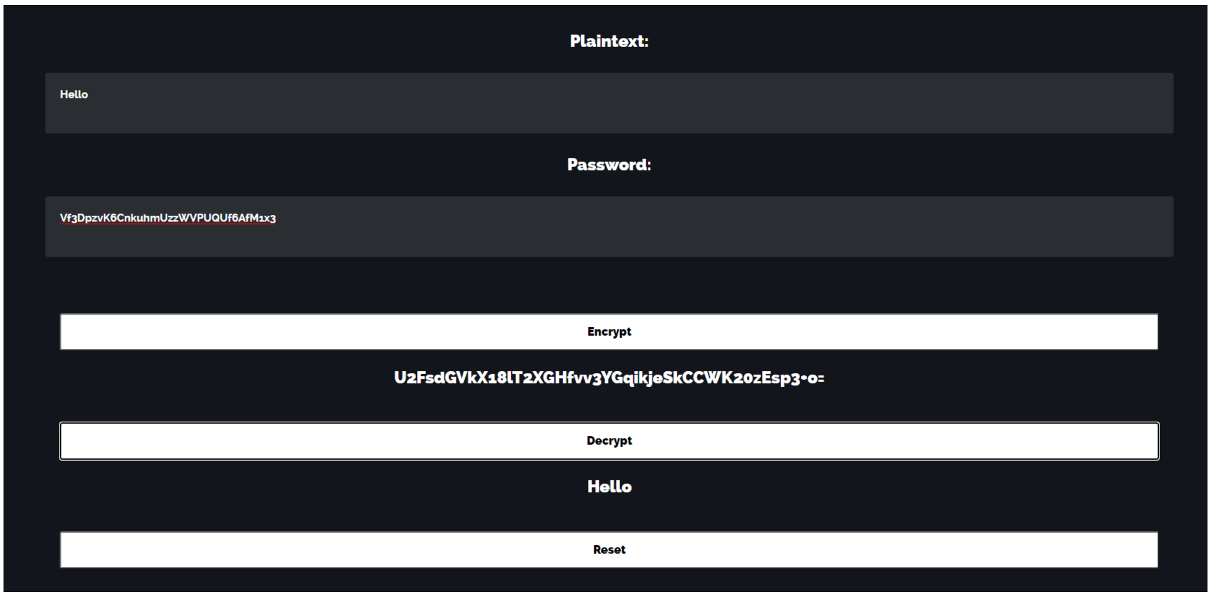
AES

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.



```
Hello
Vf3DpzvK6CnkuhmUzzWVPUQUf6AfM1x3
U2FsdGVkX18lT2XGHfvv3YGqikjeSkCCWK20zEsp3•o=
Hello
```

Figure 12: AES

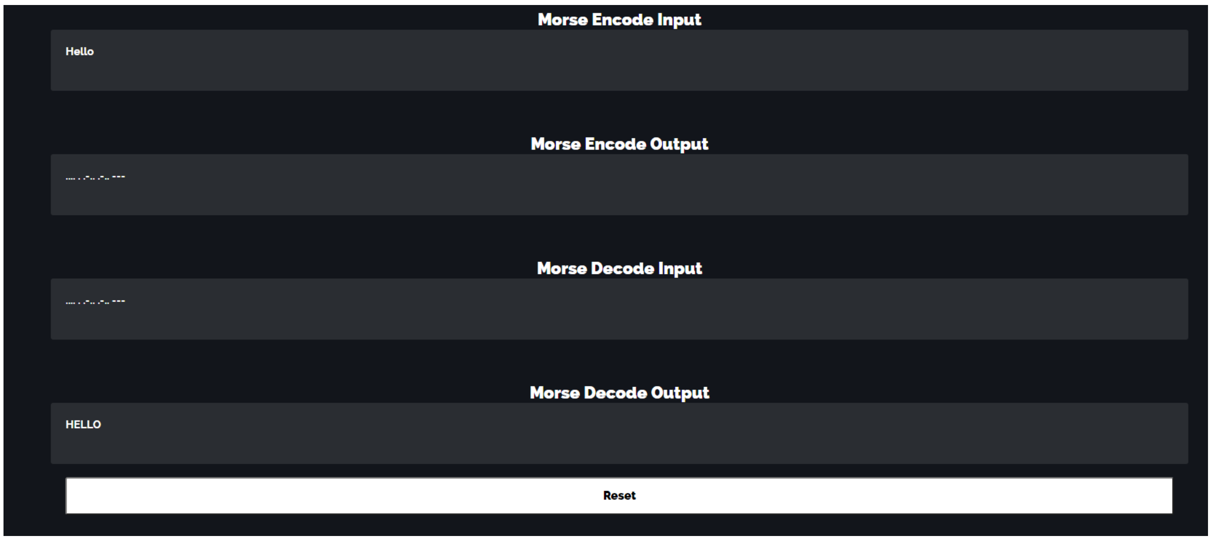
Morse Code

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.



Hello

.... . .-.. .-.. ---

.... . .-.. .-.. ---

HELLO

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