Rubigma

By group Gremy

Full name	Student ID
Kenneth Sim Jian Hui	1002661
Soh Jun De	1002682
Tay Qin Long	1003113
Sara Leong Wan-en	1003116

Introduction

Our CTF is inspired by the classic 3-D combination game, the Rubik's Cube. In brief, each square on a face of a cube, also known as “cubies”. Each cubies will contain a character and if each cubies is in the right arrangement, it will reveal the correct text. Hence, if we were to scramble the cube, the resulting cube is the cipher text and the steps taken (the rotation of the faces of the cube) is the key.

Our CTF is designed with two layers:

1. The first layer involves encrypting the plaintext which contains our flag using a modified Enigma Cipher. By using a substitution cipher, it allows us to add an element of confusion.
2. The second layer will take the resultant ciphertext from the Enigma cipher and will be broken into blocks and each block will undergo the Rubik’s cube cipher. By using a transposition cipher, it allows to add an element of diffusion. The product of each block cipher will be concatenated and the resultant ciphertext will be presented to the players.

To keep to the requirements and not make the CTF too difficult, we are currently looking at using the standard 9 cubies per cube face for our transposition cipher. However, we realise that it may be too easy, hence, we are considering increasing the number of cubies per face or using an odd shaped Rubik's cube. We plan to host the code on the cloud using flask, and encrypt/decrypt via REST.

The Key

The players will have to find out the keys, first for layer 2, the second for layer 1
The key length is undefined and the format will be as follows:

[layer 2 key || layer 1 key]

Layer 1 - Modified Enigma Cipher

For our first layer, we are doing a modified version of the Enigma Cipher. Our version is based on the Enigma I[1] cipher. In order to prevent online enigma solvers, and pre-made decrypters, we will be using our own rotors.

We intend for the player to solve the problem as would how Alan Turing solved it with his Bombe Machine, by realising that the Germans always signed off their message with the same text, they were able to use that information to break the cipher. Similarly, Players will be given a substring of the plaintext (i.e. '....HELLOWORLDISASTRING.....'). The player will use this string to identify its corresponding ciphertext cipher text and solve it from there.

The Enigma cipher mainly has two sections. First the plugboard, and second the rotors.

<u>The plugboard</u>

The plugboard is a section of the circuit that mixes up the letters of the input and output.
An excerpt from Wikipedia state:

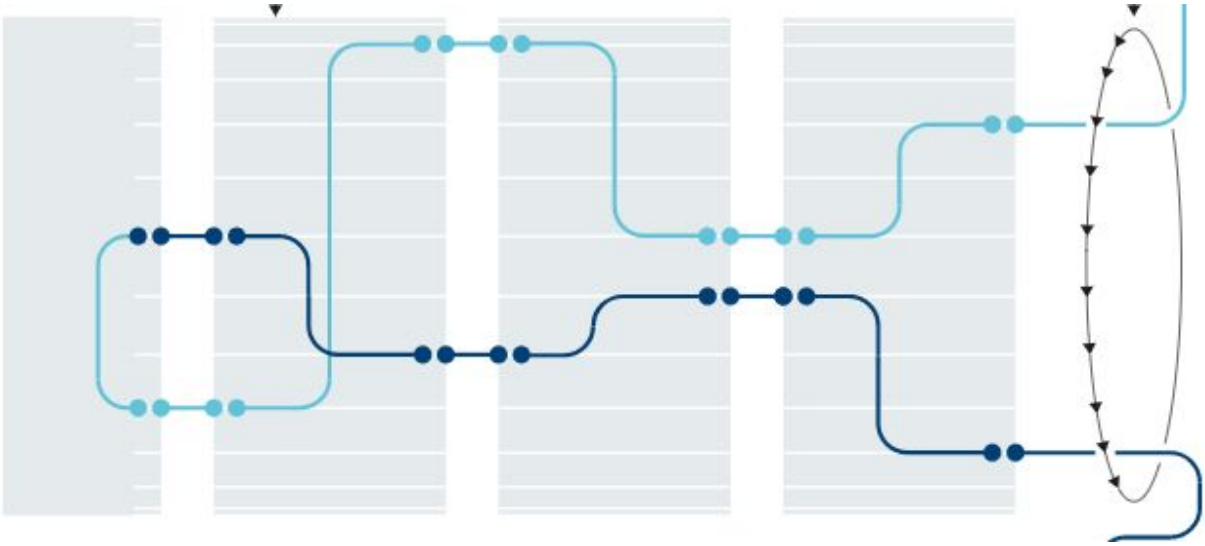
> A cable placed onto the plugboard connected letters in pairs; for example, *E* and *Q* might be a steckered pair. The effect was to swap those letters before and after the main rotor scrambling unit. For example, when an operator presses *E*, the signal was diverted to *Q* before entering the rotors.

For our cipher, we are limiting it to 5 of such plugboard connections.

<u>The rotors</u>

Our version of the Enigma cipher closely resembles the design of the original. There will be 3 available rotors to choose from, the player would be able to swap the order of the 3. The rotors will allow us to map a new letter into the input, as shown below. Each rotor has its own unique connections, and depending on how it is positioned, it will lead to a different output. The signal will go from right to left, then to a reflector, before going from left to right again.

[1] https://www.cryptomuseum.com/crypto/enigma/i/index.htm



After every letter, the third(right-most) rotor will increment by once. There will be physical notches on the first and second rotor. There might or might not be multiple notches on the rotors. This means that after a certain number of increments by the third rotor, the second rotor would increment, and finally the first one would also increase.
We will be letting the player know all the wirings of all the 5 rotors, the reflector, and the position of the notches.

Key format

The key format is as follows

1. The order of the 3 rotors, e.g. 020301, Rotor number 2 is first, followed by rotor number 3 then finally 1
2. The initial setting of the 3 rotors, e.g. 221004, the first rotor (rotor number 2) is set to 22, second rotor (rotor number 3) is set to 10, the third rotor (rotor number 1) is set to 04
3. The plugboard pairings (max length 10, for 5 pairings), e.g. ACDGFE, A is swapped with C, D swapped with G and F swapped with E.

The final key for this combination would be `020301221004ACDGFE`.

Layer 2 - Rubik's Cube

For our second layer, we'll be splitting the ciphertext from layer 1 into 54 byte blocks and encrypting each of them using electronic codebook mode using our Rubik's cipher.

Essentially this is done by choosing a side as the "white facing side", assigning each byte onto each cubies on the face of the Rubrics cube (relative to this white facing side), executing a fixed sequence of rotations of the Rubik's cube (representing the "key" of our cipher), and retrieving the bytes as the ciphertext of our cipher. A Rubik's cube will be provided in the course of the CTF to help the participants solve the Rubik's cipher.

For example, (refer to the left figure in Figure 1.1) given a string of 54 bytes of
[64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11], they would be assigned to a square in the Rubik's cube, such that the square indicating 1 would hold the 1st value of the byte string, the square indicating 2 would hold the 2nd value of the byte string, and so forth.

If the key of the cipher only instructs that the orange face be rotated clockwise once, the result is displayed at the right side of figure 1.1. When retrieving the cipher text, we will retrieve the bytes in the order of the original positions of the Rubik's cube. Hence the cipher text from figure 1.1 would be [64, 63, 53, 61, 60, 50, 58, 57, 47, 55, 54, 44, 52, 51, 41, 49, 48, 38, 46, 45, 35, 43, 42, 32, 40, 39, 29, 37, 36, 62, 34, 33, 59, 31, 30, 56, 28, 27, 26, 25, 24, 23, 22, 21, 20, 13, 16, 19, 12, 15, 18, 11, 14, 17]. In other words, the original square with index 3 will be assigned the value of the square with index 12.

As for the position of each square in the Rubik's cube, they are represented by a 1 dimensional array of indices, starting from the upper left corner of the white face. Hence the starting index array would be [1,2,3,4,5,...,54], and the index array after the aforementioned translation would be
[1,2,12,4,5,15,7,8,18,10,11,21,13,14,24,16,17,27,19,20,30,22,23,33,25,26,36,28,29,3,31,32,6,34,35,9,37,38,39,40,41,42,43,44,45,52,49,46,53,50,47,54,51,48].

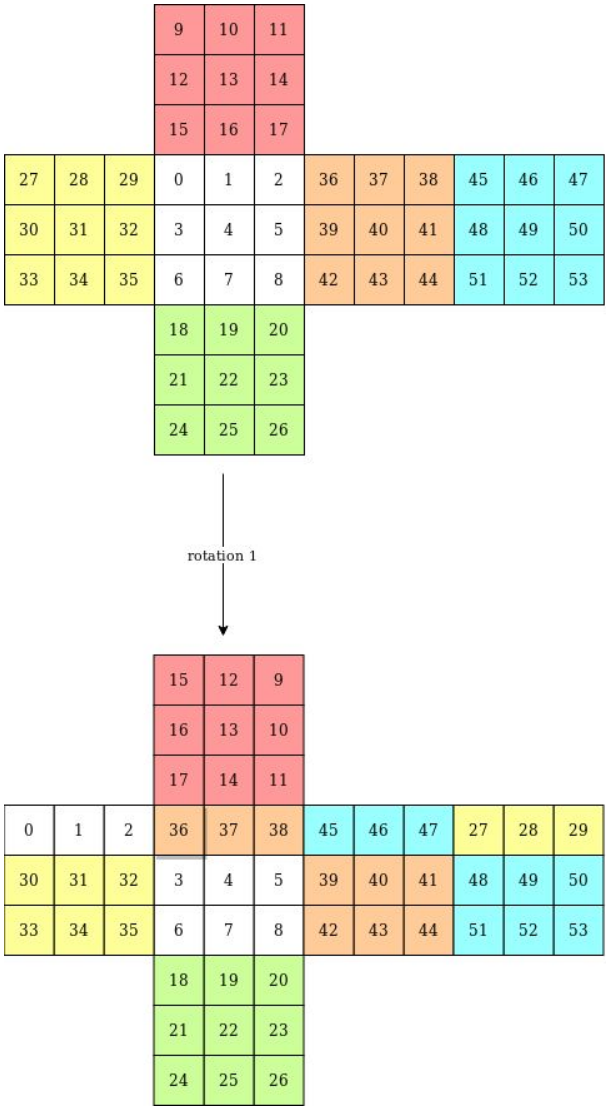


Fig 1.1: Rotation of the Rubik's Cube and the mapping of plain text to cipher

To represent every possible rotation of the Rubik’s cube (relative to the previously chosen white face), each possible rotation would represent a base 12 value (refer to Figure 1.2 below). Hence an example of a key with 5 different rotations could be “53a8b”.

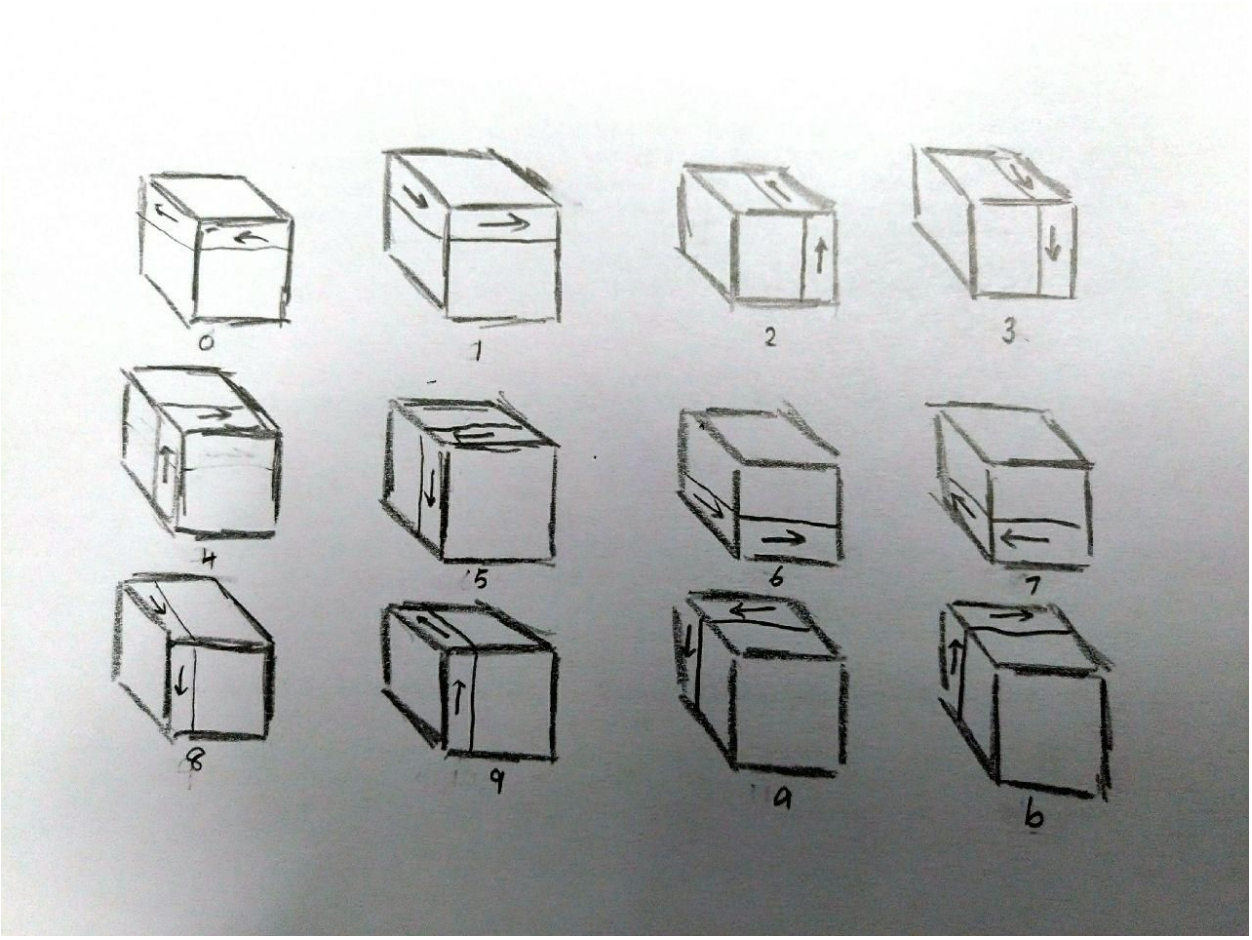


Fig 1.2 Encoding each rotation into a key value (each value here will be mapped to an alphabet for our key, where 0 is mapped to “A”, 1 to “B”, …, b is “L”)

Decryption

For the entire program, both encryption and decryption keys are identical. When decryption, all actions will be taken in inverse.

Solution

What are we providing:

1. Ciphertext
2. A string of text within the plaintext ‘...HELLOWORLDISASTRING...’, but not position of said string
3. Final Rubik’s Cube layout, represented by an array of indices
4. All documentation provided in this document

For the player to solve the cipher, we expect the player to identify that the Rubik's cipher is the weaker cipher, and solve that layer before solving the enigma layer.

Solving the Rubik's cube layer

Being given the Rubik's cube with the final layout, and knowing the cube's inherent starting layout (1,2,3,4,5,6,...,54), we expect the quickest solution to be the mapping of the indices of ciphertext to the indices of the plaintext, and using this mapping to decrypt the ciphertext without knowing the key to the cipher.

Alternate methods include writing the code to translate the ciphertext into plain text using the cipher key, translating the steps to solve the Rubik's cube into the cipher key, and using both to decrypt ciphertexts.

Solving the enigma machine

Stated separately in the other pdf