Rubigma by Team Gremy

1 Introduction

Here is what you are provided for this CTF:

1. Rest API link: `audacia.online:8000`

   Please read the documentation below on how to use this link and access endpoints.

2. The ciphertext `ciphertext.txt` and its corresponding cube image, `cube.jpeg`
3. The cipher code for the enigma layer can be found in `enigma.py`
4. The following documentation on our how our Enigmatic Rubik's Cipher works

Goal:

Your task is to obtain the flag which has been encrypted by our Enigmatic Rubik's cipher. Do read the following documentation to get a better understanding of how the cipher works. We have provided the code for the enigma layer `enigma.py` for you to try it out. We have set up endpoints for you to check if you are on the right track at each layer

2 How to use the Server

There are two endpoints to the REST server:

<table>
<tr><td>Endpoint:</td><td colspan="2">audacia.online:8000/cipher</td></tr>
<tr><td>Example URL:
(encrypt)</td><td colspan="2">http://audacia.online:8000/cipher/?input=HELLOWORLD&key=ACCBA020103211504CDHA&mode=E&layer=0</td></tr>
<tr><td>(decrypt)</td><td colspan="2">http://audacia.online:8000/cipher/?input=0x895a6cb9b300c8a900000000000000000000000000000000000000260000000000000000000000000000000000000000ec0000000000&key=ACCBA020103211504CDHA&mode=D&layer=0</td></tr>
<tr><td><b>key</b></td><td><b>value</b></td><td><b>description</b></td></tr>
<tr><td>input</td><td>HELLOWORLD</td><td>The input to cipher or decipher</td></tr>
<tr><td>key</td><td>http://audacia.online:5000/cipher/?input=HELLOWORLD&key=ACCBA020103211504CDHA&mode=E&layer=0ACCBA020103211504CDHA</td><td>If layer==0:
format is key_for_layer2 + key_for_layer1 (they are appended)
If layer==1:
format is key_for_layer1 (layer1 key alone)
If layer==2:
format is key_for_layer2 (layer2 key alone)</td></tr>
<tr><td>mode</td><td>E</td><td>E for encrypt, D for decrypt</td></tr>
<tr><td>layer</td><td>0</td><td><table>
<tr><td>Layer number</td><td>Description</td></tr>
<tr><td>0</td><td>Both layer 1 + layer2</td></tr>
<tr><td>1</td><td>Only layer 1</td></tr>
<tr><td>2</td><td>Only layer 2</td></tr>
</table></td></tr>
</table>

Endpoint:	audacia.online:8000/answer	
Example URL:	http://audacia.online:8000/answer/?input=THISISMYANSWER	
key	**value**	**description**
input	THISISMYANSWER	Midway answer or final answer, this endpoint will allow you to check for both Pls don’t DDOS :(

2 Key hints

The flag is only comprised of 26 uppercased letters of length 73 with no spaces.

Due to the nature of our cipher, the final format you will get for the flag is

CTFF.....SPEAKGOODSINGLISHMOVEMENT.... (without brackets)

Please add in the brackets before submitting:
CTF{F....SPEAKGOODSINGLISHMOVEMENT...}
You can check with the /answer endpoint to verify it before submitting.

You can solve the CTF without the use of the REST server.

Hints for Enigma layer:

- The first four letters of the flag is ‘CTFF’.
- In terms of Python,

  flag[10:27] = ‘SPEAKGOODSINGLISHMOVEMENT’ (note: it’s Singlish, not English)

- There are only 2 plugboard lines, no more, no less.
- The plugboard lines are **not** connected to the letters ‘C’, ‘T’, and ‘F’.
- Enigma also has a self-coding weakness so no letter can code as itself. i.e A cannot be encoded as A
- This version of the Enigma Cipher is more restricted compared to the one used in WW2, as such there are easier ways of solving it compared to the Bombe Machine

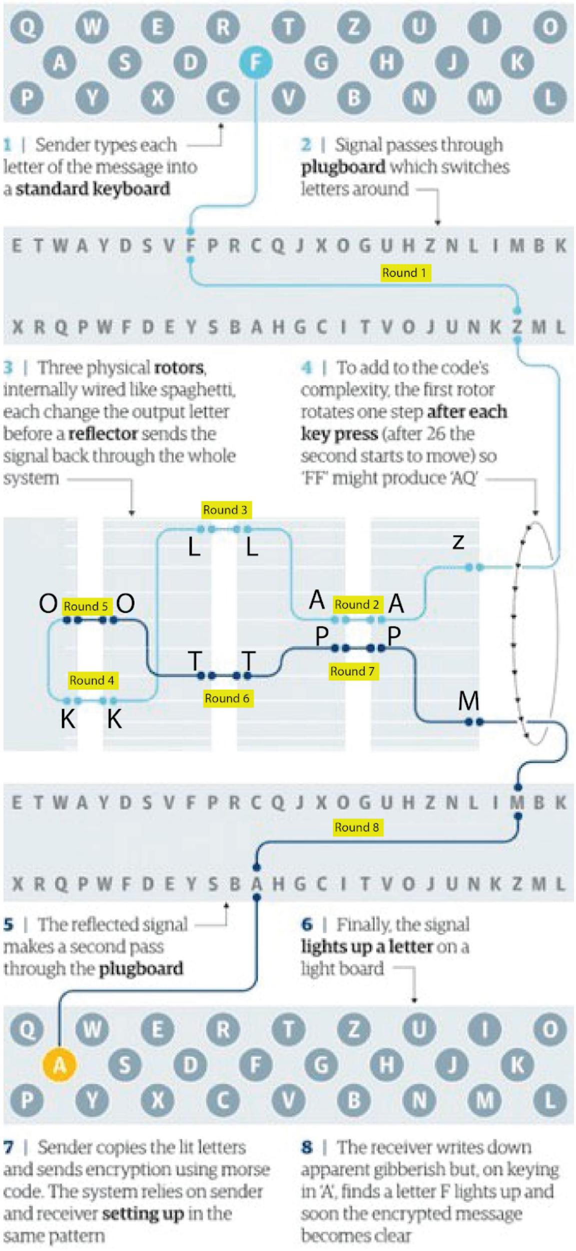
3 Overview of the Enigmatic Rubik's Cipher

The cipher is designed with two layers:

1. The first layer involves encrypting the plain text which contains our flag using a Modified Enigma Cipher.

2. The second layer will take the resultant cipher text from the Enigma cipher and will be broken into blocks. Each block will undergo the Rubik's cube cipher. The resultant product of each block cipher will be concatenated, producing a resultant cipher text which is what you have been provided with.

Enigma How the machine worked


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
E T W A Y D S V F P R C Q J X O G U H Z N L I M B K
Round 1
X R Q P W F D E Y S B A H G C I T V O J U N K Z M L
Round 3
L L
Z
O Round 5 O
A Round 2 A
P P
T T
Round 4
Round 7
Round 6
M
K K
E T W A Y D S V F P R C Q J X O G U H Z N L I M B K
Round 8
X R Q P W F D E Y S B A H G C I T V O J U N K Z M L
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

1 | Sender types each letter of the message into a **standard keyboard**

2 | Signal passes through **plugboard** which switches letters around

3 | Three physical **rotors**, internally wired like spaghetti, each change the output letter before a **reflector** sends the signal back through the whole system

4 | To add to the code's complexity, the first rotor rotates one step **after each key press** (after 26 the second starts to move) so 'FF' might produce 'AQ'

5 | The reflected signal makes a second pass through the **plugboard**

6 | Finally, the signal **lights up a letter** on a light board

7 | Sender copies the lit letters and sends encryption using morse code. The system relies on sender and receiver **setting up** in the same pattern

8 | The receiver writes down apparent gibberish but, on keying in 'A', finds a letter F lights up and soon the encrypted message becomes clear

PAUL SCRUTON, GUARDIAN GRAPHIC SOURCE: SIMON SINGH, LOUISE DADE

3.1 Modified Enigma Substitution Cipher

This first layer encrypts the plaintext flag to generate ciphertext by using a plug board and rotating alphabet rotors. (see Diagram on the left)

The Modified Enigma Substitution Cipher puts the plain text through 8 rounds of substitutions. The first (round 1) and last (round 9) substitution rounds are arbitrary mapping of one character to other characters while the other 6 rounds are done through mappings of one of the three rotors.

Each one of the rotors have 26 positions, each being represented by a number. The six rounds of mapping is based on how a letter in one wheel is adjacent to the one on the wheel beside this one.

Follow the diagram to understand how the mapping from one rotor to the other is done.

The mappings of the rotors are not static, and third(right) rotor rotates after the encryption of every character. The three wheels can be treated like a tally counter, after the third wheel reaches a certain position, the second wheel will move (due to a notch on it), this is the same for the first wheel.

It is not needed to know how the wiring is done, nor where the notches are to solve this CTF. We have set up an endpoint for you to check if your flag plaintext is right.

Key format

The key format is as follows

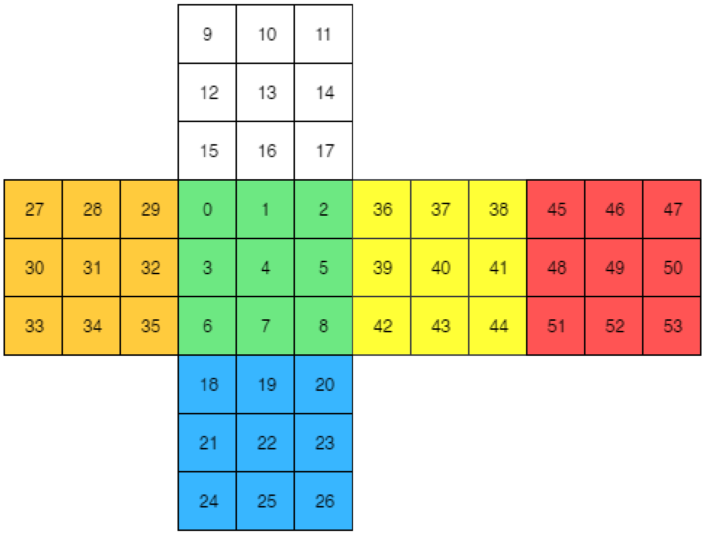
1. The order of the 3 rotors, e.g. 020301, Rotor number 2 is first, followed by rotor number 3 then finally 1
2. The initial setting of the 3 rotors, e.g. 221004, the first rotor (rotor number 2) is set to 22, second rotor (rotor number 3) is set to 10, the third rotor (rotor number 1) is set to 04
3. The plugboard pairings (length 4, for 2 pairings), e.g. ACDG, A is swapped with C, D swapped with .

The example final key for this combination would be `020301221004ACDG`. The key for the enigma cipher is a symmetric one.

2.2 Rubik's Transposition Cipher

For our second layer, we'll be splitting the ciphertext from the enigma layer into blocks of 54 bytes. Each block will be encrypted using electronic codebook mode using our Rubik's cipher.

Each byte in each block can be reimagined onto a Rubik's cube. The position of the byte will occupy the cubes based on the corresponding position index. The position index of each cube on the Rubik's cube is as follows:



			9	10	11						
			12	13	14						
			15	16	17						
27	28	29	0	1	2	36	37	38	45	46	47
30	31	32	3	4	5	39	40	41	48	49	50
33	34	35	6	7	8	42	43	44	51	52	53
			18	19	20						
			21	22	23						
			24	25	26						

Imagine holding an actual Rubik's cube. If you were to turn a layer of the cube, how would the cubes move and what would be the new arrangement?

Choose a color as the main face of reference (in our case we choose green). The key is represented by a specific set of rotations. By executing this fixed sequence of rotations of the Rubik's cube, we are able to encrypt and decrypt our plaintext and ciphertext respectively.

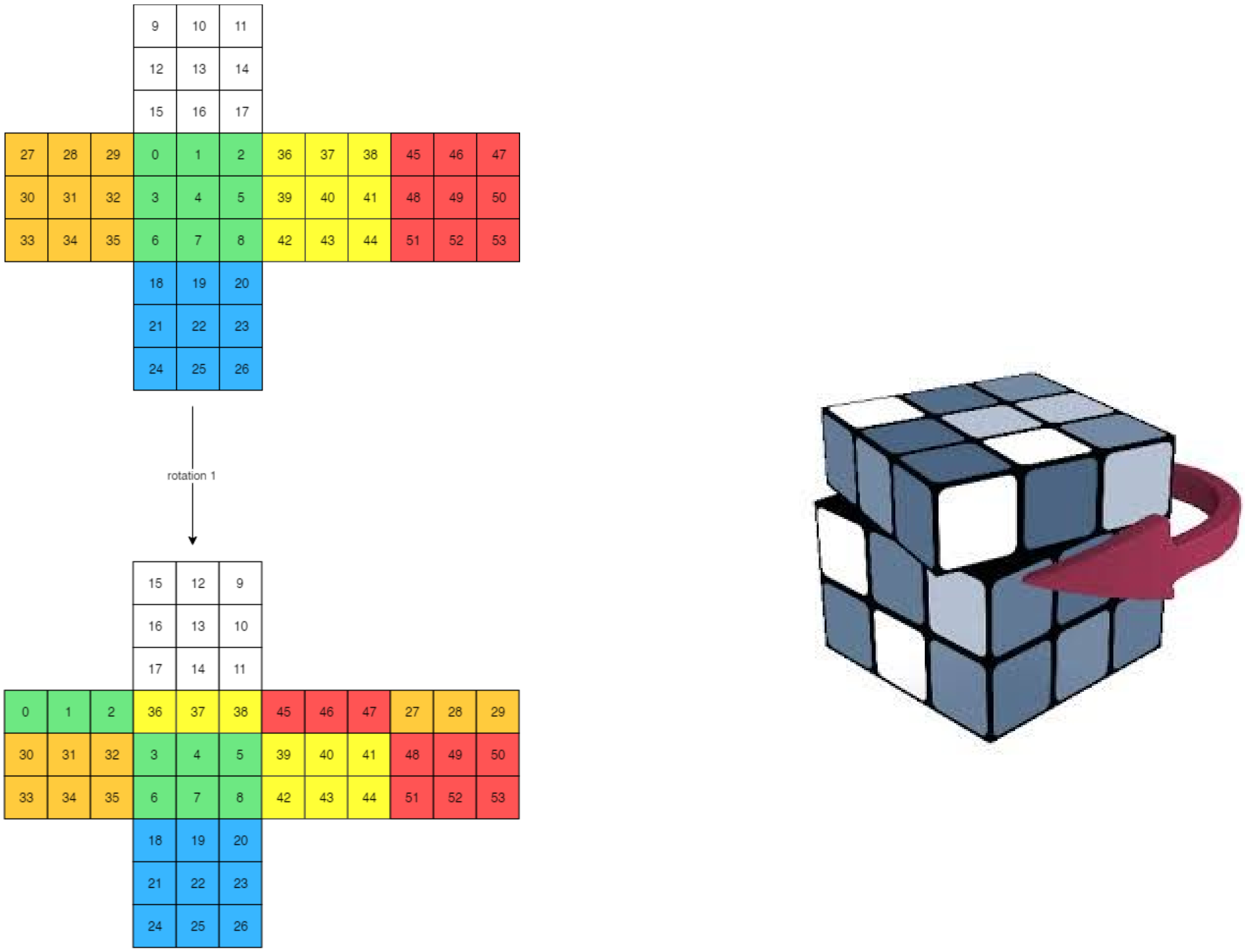


Fig 1.1: Rotation of the Rubik's Cube top most layer and movement of cubes

You are given the resultant ciphertext and the corresponding cube image, cube.jpeg, find the original byte sequence. As mentioned above, we have set up an endpoint for you to check if you managed to get the right cipher text. If it is correct, you have obtained your ciphertext for the enigma cipher.

-End-