**casApplied Crypto**

**Learning Enigma Machine**

Problem 1

Describe the main components of a typical enigma machine.

The enigma Machine was a typewriter-like machine used by the German military in 1930 with the purpose of encrypting and decrypting messages. Using the enigma machine usually involved two people:

* Person one: Responsible for typing the message.
* Person two: Responsible for recording which letters were illuminated when Person 1 pressed a key.

The encrypted message was sent – via Morris code – to another person who then typed the encrypted message into their enigma machine, and the letters would light up, decrypting the message.

The Enigma Machine has the following main components:

**Battery:** Power supply for the circuit

**Keyboard with 26 keys**.

Each letter on the keyboard is connected to a **Key switch** – which has 3 copper tabs that are each responsible for directing the circuit. When a key on the keyboard is pressed, the circuit has the following path:

Keys switch -> plugboard -> key switch (which could be the same key switch or a different one depending the plugboard configuration) -> input wheel -> rotor 1-3 🡪 reflector -> rotor 3-1 -> input wheel -> plug board -> key switch top tab->light bulb -> battery (completing the circuit)

* + The top and middle tab are connected by default
    - Top tab connected to lightbulbs
  + The middle and bottom tab are connected when a key on the keyboard is pressed.

**Input wheel:** Has 26 wires and connects of the rotors.

**Rotors:**

There are 3 rotors that each have numerical numbers (1-26). Each number correlates to a letter (example 1=A, 2=B, and so forth). The purpose of the Rotors is to scramble each letter. Each time the electrical current goes through a rotor, the rotor is scrambled.

Each time a key is pressed, at least one of the rotors is turned, changing the electrical contacts.

* Rotor 1 changes with each key press.
* Rotor 2 changes after the 26th key press.
* Rotor 3 changes after Rotor two has changed 26 times, therefore once every 676 turns.

**Reflector:** The reflector sits at the end of the three rotors and bounces the electrical signal back through the rotors. When the signal passes through the reflector, the letter is encrypted yet again.

**Plugboard**

* + Another way to swap letters is by creating a circuit.
  + Users can create up to 10 circuits, making 10 letter pairs total. This means at least 6 letters are left out.

**Light bulb:** The light bulb indicates which the “encrypted” letter. For example, if the letter A is pressed and the light bulb above letter T illuminates, this means that A is encrypted into T.

# Problem 2

What is the size of the key space?

Rotor selection: 5\*4\*3 = 60

Rotor settings: (26\*26\*26) = 17576

Plugboard settings: Pairs of 10 therefore C(20, 2) = 190

60\*17576\*190=**200366400**

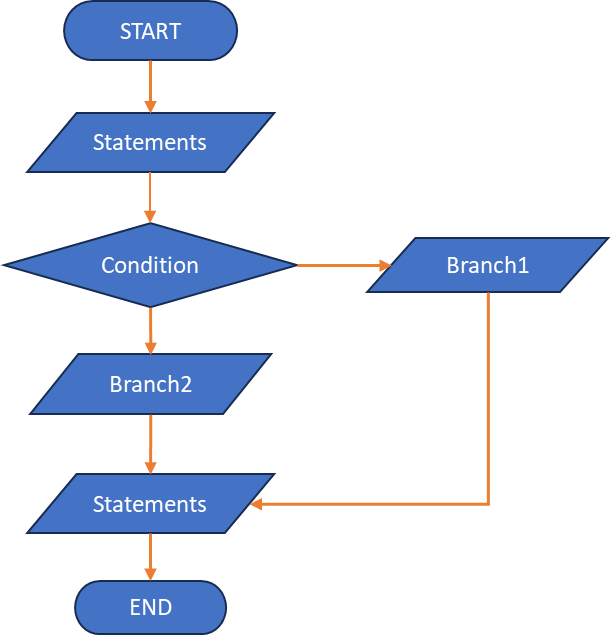
Refer to the manual at <https://py-enigma.readthedocs.io/_/downloads/en/latest/pdf/> Students can construct an enigma machine. I list my test code here for your reference.

from enigma.rotors.rotor import Rotor  
from enigma.plugboard import Plugboard  
from enigma.machine import EnigmaMachine  
  
rL = Rotor('my rotor1', 'EKMFLGDQVZNTOWYHXUSPAIBRCJ', ring\_setting=0, stepping='Q')  
rM = Rotor('my rotor2', 'BDFHJLCPRTXVZNYEIWGAKMUSQO', ring\_setting=5, stepping='V')  
rR = Rotor('my rotor3', 'ESOVPZJAYQUIRHXLNFTGKDCMWB', ring\_setting=10, stepping='J')  
  
reflector = Rotor('my reflector', 'YRUHQSLDPXNGOKMIEBFZCWVJAT')  
  
pb = Plugboard.from\_key\_sheet('AK BZ CG DL FU HJ MX NR OY PW')  
  
machine = EnigmaMachine([rL, rM, rR], reflector, pb)  
  
machine.set\_display('UPS') # set rotor positions or use its default  
position = machine.get\_display() # read rotor position  
print(position)  
  
# Encrypt A letter  
#print(machine.key\_press('C'))  
# Encrypt a text  
print(machine.process\_text('Enigma machine is powerful for Q'))

# Problem 3 (WIP)

Explain the flow of the code and map the settings to the components mentioned in Problem 1.

One approach is to use workflow block diagram. It visualizes working logic (algorithm) of a code. A typical sample workflow is shown below. The purpose is to walk through the code like a professional software engineer, understand how the code works and do code development such as fixing bugs and adding new features, which is much-needed skills for tech students.



# Problem 4

Learning by testing. Test the code with at least 7 various keys (change of wiring, ring\_setting, stepping, reflector, with or without plugboard, initial display position, etc.

**See code the following functions:**

Problem4\_1()

Problem4\_2()

Problem4\_3()

Problem4\_4()

Problem4\_5()

Problem4\_6()

Problem4\_7()

* KFRZGEDTWEJWHLSFBZEVFRFCBRDLKOTM
* IVZSXJSYWXXKMTPZDSNBYVVHEWAZVPAF
* CADFJVEQKOIKWWUNEAXQLDGTGGECHXTR
* TTXOPNHJCINQGNOVGMGIAKVPHOZDWZAY
* JXFLJUPJIIFIICPQSPRYQSWTTZEVPMYY
* TTXOPNHJCINQGNOVJMGIAKVPHOZDWZAY
* XVWEOHTCUZYBMJOORLVDHKTPNERHQZAK

# Problem 5.

Pick one of the ciphertexts from problem 4 and decrypt it. Make sure you keep the same parameters (or keys). Do we need to write another code for that?

**Answer:** For this problem, I selected key the following key: 'KFRZGEDTWEJWHLSFBZEVFRFCBRDLKOTM' from problem 4 and received the following output: ENIGMAXMACHINEXISXPOWERFULXFORXQ – where ‘X’ represents spaces.

See the following code for more information:

def problem5():

    Problem4\_1('KFRZGEDTWEJWHLSFBZEVFRFCBRDLKOTM')

Problem 6. Time for fun. Same setting as given in the sample code, but folks at Bletchley Park want to know the initial position (display). They had a ciphertext “WVUVJCSQBFLWSGTHDREWOSXYIAYEUBHHXY” from the known plaintext “ATTACK AT 5PM AT ATLANTIC Z ISLAND”. Code and find the init display.

Problem 7 (optional) Same enigma machine given by the sample code. We had the ciphertext “YNLIUNHBNVERXKRBUHZEYMJVEZNRPNWOSV” from the known plaintext “ATTACK AT 5PM AT ATLANTIC Z ISLAND”. Recover as many of the stecker settings as it possible from the known plaintext.

**Reference**

**Interesting Reading**: the more realistic decryption process “Example communication procedure” at <https://py-enigma.readthedocs.io/en/latest/guide.html> in which it gives a protocol of setting up initial display as communication vehicle is through radio signal, copied it here for fun.

“We will now present a simplified scenario based on a procedure employed by the army (*Heer*) after 1940. This example is based upon one found in Dirk Rijmenants’ simulator manual, which is based upon a real-life example from Frode Weierud’s Cryptocellar website.

Suppose a message needs to be transmitted. The operator of the transmitting machine consults his ***key sheet*** and configures his machine according to the daily settings found inside. Let’s suppose the key sheet dictates the following initial parameters for the current day:

* Rotor usage and order is *II IV V*
* Ring settings for each rotor, in order, are: *B U L*
* Plugboard settings are: *AV BS CG DL FU HZ IN KM OW RX*
* One of the daily Kenngruppen possibilities is *UGZ*

Let us also assume the reflector employed by this army unit is ‘B’.

The operator then configures his machine:

machine = EnigmaMachine.from\_key\_sheet(

rotors='II IV V',

reflector='B',

ring\_settings='B U L',

plugboard\_settings='AV BS CG DL FU HZ IN KM OW RX')

Suppose the Enigma operator was handed a message for transmit by an officer which reads “The Russians are coming!” The operator would first randomly decide two things:

* Initial rotor positions, say WXC
* A three letter *message key*, say BLA

The operator would then turn the rotor thumb wheels to set the initial rotor position and then type the three letter message key to produce an encrypted message key:

machine.set\_display('WXC') # set initial rotor positions

enc\_key = machine.process\_text('BLA') # encrypt message key

In this example, the encrypted key turns out to be KCH. This is written down for later.

The operator then sets the rotors to the unencrypted message key BLA and then types in the officer’s message, performing various substitutions and transformations according to training and current procedures. In our simple case, he performs the following:

machine.set\_display('BLA') # use message key BLA

ciphertext = machine.process\_text('THEXRUSSIANSXAREXCOMINGX')

print(ciphertext)

This produces the ciphertext NIBLFMYMLLUFWCASCSSNVHAZ.

Next, between the Enigma operator and the radio operator, a message is formed up. This message includes the following components:

* The time of transmission
* The station identification for transmitter and intended recipient(s)
* The message length; in our case this is 24
* The initial rotor positions in unencrypted form (WXC)
* The encrypted message key value (KCH)
* The unencrypted message indicator (*Kenngruppen*)
* The encrypted message contents

In our example, the message handed over to the radio operator to be transmitted by either Morse code or perhaps even voice would look something like this:

U6Z DE C 1500 = 24 = WXC KCH =

BNUGZ NIBLF MYMLL UFWCA

SCSSN VHAZ=

The top line indicates day 31, station C transmits to station U6Z, sent at 1500 hours and contains 24 letters. The starting position is WXC and the encrypted message key is KCH.

Next we have the body of the message. The army transmitted messages in 5 letter groups. The first group contains the Kenngruppen, or indicator. Procedure required the operator pick one of the Kenngruppen possibilities from the key sheet, and then pad it out with two random letters. Here the operator chose to prepend BN to the Kenngruppen value of UGZ. He could have also appended the two letters, or perhaps appended one and prepended the other.

After the message indicator group, the encrypted text follows in 5 letter groups.

Now at receiving station U6Z, the radio operator receives the over-the-air message and types or writes it up in the form shown and hands it to the Enigma operator.

The Enigma operator first looks for the message indicator. He uses the group BNUGZ and scans his key sheet for either BNU, NUG, or UGZ. He could presumably also use the date information found in the message preamble to help his search of the key sheet. If everything checks out the operator now knows which entry in his monthly key sheet to use. Thus, as was done at the transmitting station, he configures his Enigma according to the key sheet:

machine = EnigmaMachine.from\_key\_sheet(

rotors='II IV V',

reflector='B',

ring\_settings='B U L',

plugboard\_settings='AV BS CG DL FU HZ IN KM OW RX')

The receiving operator then must decrypt the message key:

machine.set\_display('WXC')

msg\_key = machine.process\_text('KCH')

This should reveal that the message key is the original BLA. The rotors are then set to this value and the message can be decrypted, taking care to ignore the Kenngruppen:

machine.set\_display(msg\_key) # original message key is BLA

plaintext = machine.process\_text('NIBLFMYMLLUFWCASCSSNVHAZ')

print(plaintext)

The Enigma operator then decodes the message “THEXRUSSIANSXAREXCOMINGX”. He then uses his training and procedures to further process the message. Finally, the somewhat troubling message “The Russians are coming” is handed to his commanding officer.”