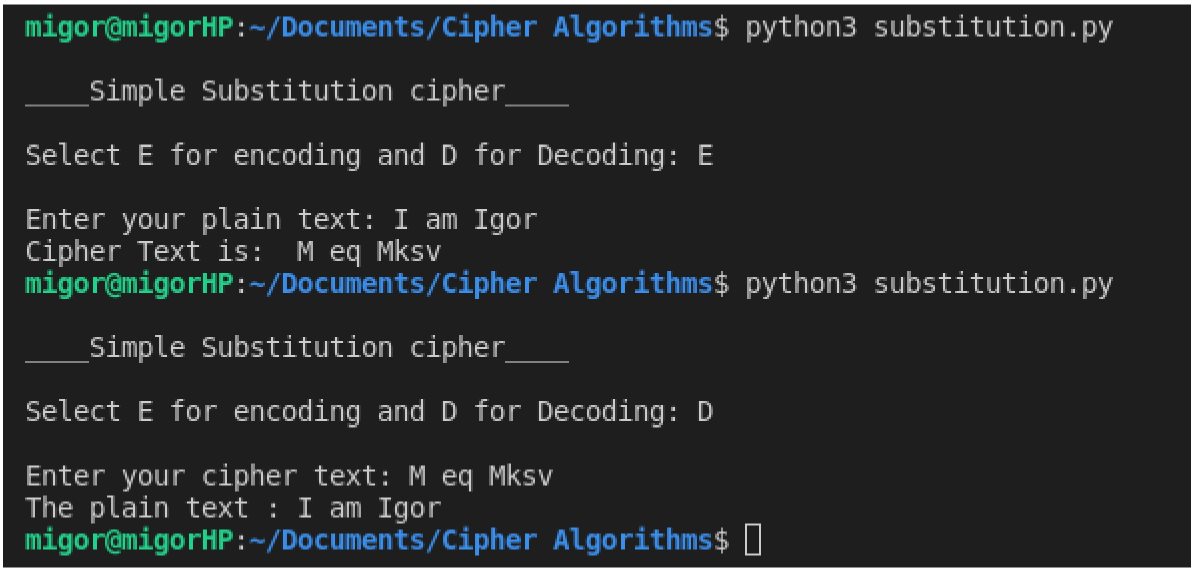
# Lab 1: Crypto and Steganography Basics.

> Igor Mpore
>
> BS19-CS01
>
> i.mpore@innopolis.university

## Crypto Basics

### 1. Substitution Cipher algorithm

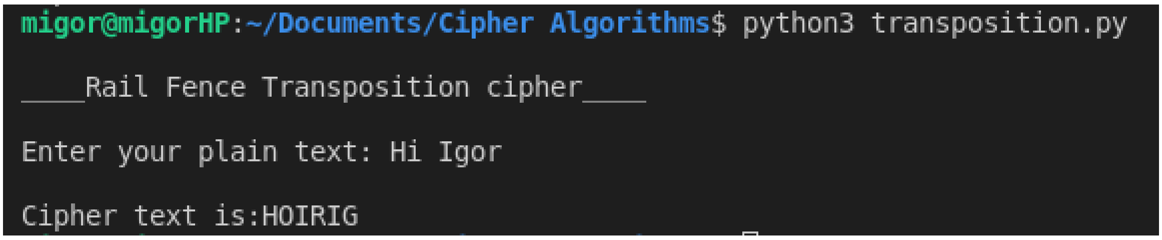
```python
import string

alpha_letters= string.ascii_letters

key = 4

# Dictionary for substituted alphabets in plain text
dict1 = {}
for i in range(len(alpha_letters)):
    dict1[alpha_letters[i]] = alpha_letters[(i+key)%len(alpha_letters)]


def encode_funct(ptext):
    cipher_txt=[]
    # loop to generate ciphertext
    for char in ptext:
        if char in alpha_letters:
            temp = dict1[char]
            cipher_txt.append(temp)
        else:
            temp =char
            cipher_txt.append(temp)
    cipher_txt= "".join(cipher_txt)
    print("Cipher Text is: ",cipher_txt)


# Dictionary for substituted alphabets in cipher text
dict2 = {}
for i in range(len(alpha_letters)):
    dict2[alpha_letters[i]] = alpha_letters[(i-key)%(len(alpha_letters))]


def decode_funct(ctext):
    # loop to recover plain text
    decrypt_txt = []

    for char in ctext:
        if char in alpha_letters:
            temp = dict2[char]
            decrypt_txt.append(temp)
        else:
            temp = char
            decrypt_txt.append(temp)
    decrypt_txt = "".join(decrypt_txt)
    print("The plain text :", decrypt_txt)


print ("\n____Simple Substitution cipher____")

choice = input ("\nSelect E for encoding and D for Decoding: ")

if (choice == "E" or choice == "e"):
    plain_txt = input ("\nEnter your plain text: ")
    encode_funct(plain_txt)
elif (choice == "D" or choice == "d"):
    cipher_txt = input (" \nEnter your cipher text: ")
    decode_funct(cipher_txt)
else:
    print ("\nPlease enter correct value. Bye for now!")
    exit()
```



```
migor@migorHP:~/Documents/Cipher Algorithms$ python3 substitution.py

____Simple Substitution cipher____

Select E for encoding and D for Decoding: E

Enter your plain text: I am Igor
Cipher Text is:  M eq Mksv
migor@migorHP:~/Documents/Cipher Algorithms$ python3 substitution.py

____Simple Substitution cipher____

Select E for encoding and D for Decoding: D

Enter your cipher text: M eq Mksv
The plain text : I am Igor
migor@migorHP:~/Documents/Cipher Algorithms$ 
```

## 2. Transposition Algorithm

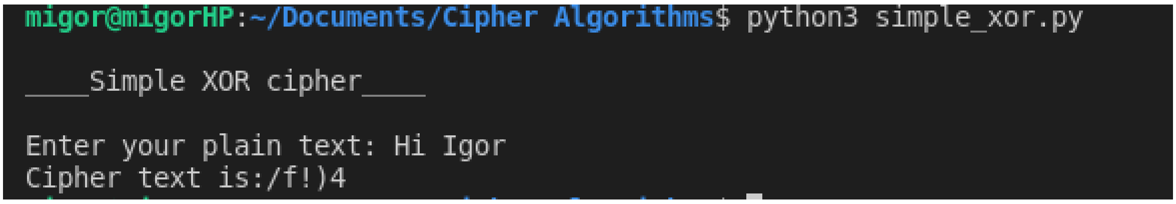
```
import string

key = 4
def encode_funct(ptext):
        # remove all white spaces in text
    plain_text = ptext.replace(" ", "")

    # change plain text to upper case
    plain_text = plain_text.upper()

    # divide plain text into layers number of strings using th key number
    rail = [""] * key
    layer = 0
    for character in plain_text:
        rail[layer] += character
        if layer >= key - 1:
            layer = 0
        else:
            layer += 1
    cipher = "".join(rail)
    return cipher

print ("\n____Rail Fence Transposition cipher____")
plain_txt = input (" \nEnter your plain text: ")
cipher_txt = encode_funct(plain_txt)
print ("Cipher text is:" + cipher_txt)
```



```
migor@migorHP:~/Documents/Cipher Algorithms$ python3 transposition.py

____Rail Fence Transposition cipher____

Enter your plain text: Hi Igor

Cipher text is:HOIRIG
```

## 3. Simple XOR Algorithm

```
import string

key = 4
def encode_funct(ptext):
    # Defining XOR key
    xorKey = 'F';

    # calculate length of input string
    length = len(ptext);
    for i in range(length):
        ptext = (ptext[:i] +
             chr(ord(ptext[i]) ^ ord(xorKey)) +
                     ptext[i + 1:]);
    return ptext;

print ("\n____Simple XOR cipher____")
plain_txt = input (" \nEnter your plain text: ")
cipher_txt = encode_funct(plain_txt)
print ("Cipher text is:" + cipher_txt)
```



```
migor@migorHP:~/Documents/Cipher Algorithms$ python3 simple_xor.py

____Simple XOR cipher____

Enter your plain text: Hi Igor
Cipher text is:/f!)4
```

# Steganography Basics

## 1. Image Steganography

(Files in the folder called "Image steg")

## 2. Audio Steganography

(Files in the folder called "Audio steg")

## 3. Video Steganography

(Files in the folder called "Video steg")

## 4. Covert Channels: Tunnelshell (ICMP)

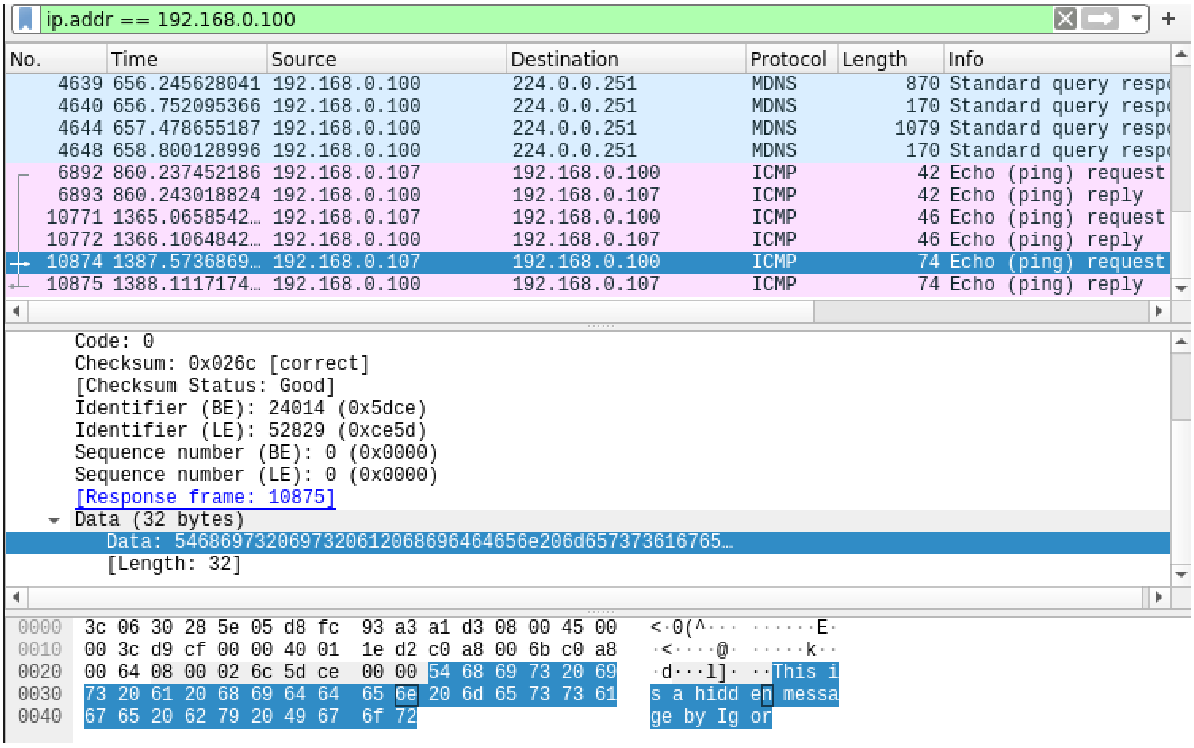
Using hping3 packet, that I installed with the command:

```
pip install hping3
```

I managed to use this command and then sniffing the packet using Wireshark to capture the sent message (screenshoot attached):

Ref: https://www.exploit-db.com/docs/english/18581-covert-channel-over-icmp.pdf

```
hping3 -1 -c 1 192.168.10.100 -e "This is a hidden message by Igor"
```



```
ip.addr == 192.168.0.100

No.   Time          Source         Destination    Protocol Length Info
 4639 656.245628041 192.168.0.100  224.0.0.251    MDNS       870 Standard query respo
 4640 656.752095366 192.168.0.100  224.0.0.251    MDNS       170 Standard query respo
 4644 657.478655187 192.168.0.100  224.0.0.251    MDNS      1079 Standard query respo
 4648 658.800128996 192.168.0.100  224.0.0.251    MDNS       170 Standard query respo
 6892 860.237452186 192.168.0.107  192.168.0.100  ICMP        42 Echo (ping) request
 6893 860.243018824 192.168.0.100  192.168.0.107  ICMP        42 Echo (ping) reply
10771 1365.0658542… 192.168.0.107  192.168.0.100  ICMP        46 Echo (ping) request
10772 1366.1064842… 192.168.0.100  192.168.0.107  ICMP        46 Echo (ping) reply
10874 1387.5736869… 192.168.0.107  192.168.0.100  ICMP        74 Echo (ping) request
10875 1388.1117174… 192.168.0.100  192.168.0.107  ICMP        74 Echo (ping) reply

    Code: 0
    Checksum: 0x026c [correct]
    [Checksum Status: Good]
    Identifier (BE): 24014 (0x5dce)
    Identifier (LE): 52829 (0xce5d)
    Sequence number (BE): 0 (0x0000)
    Sequence number (LE): 0 (0x0000)
    [Response frame: 10875]
  Data (32 bytes)
      Data: 5468697320697320612068696464656e206d657373616765…
      [Length: 32]

0000  3c 06 30 28 5e 05 d8 fc  93 a3 a1 d3 08 00 45 00   <·0(^··· ······E·
0010  00 3c d9 cf 00 00 40 01  1e d2 c0 a8 00 6b c0 a8   ·<····@· ·····k··
0020  00 64 08 00 02 6c 5d ce  00 00 54 68 69 73 20 69   ·d···l]· ··This i
0030  73 20 61 20 68 69 64 64  65 6e 20 6d 65 73 73 61   s a hidd en messa
0040  67 65 20 62 79 20 49 67  6f 72                     ge by Ig or
```

# Enigma and Bombe

The Enigma was a machine used bu German to Encrypt information to send to their allies during the World War II. Despite of it's security, the Enigma had some design problems for encrypting data which led to the creation of Bombe that used Reverse Engineering to decrypt the information. Firstly, the main weakness of Enigma was that they had to always select a random initial position of each roller which resulted later destroyed the uniform distribution of the rollers.

In addition, the Enigma was designed that the input letter couldn't encrypt to itself which would lead the attackers to know the letter which shouldn't be included during brute forcing.  Bombe also took advantage that common words were considered to be in the original message as it was all about war.

Ref: https://www.youtube.com/watch?v=-1ZFVwMXSXY