Lab 8:

Task 1: Frequency analysis

Step 1: we need to count the frequency of each letter in ciphertext.

Step 2: After counting the frequency, next step is to compare the frequency distribution of the ciphertext to the expected frequency distribution of plaintext language.

In lab example based on the frequency I mapped the letter ‘ytn’ to ‘THE’ where the ‘y’ letter in cipher text will be replaced with ‘T’, same process will be followed throughout until we get our plaintext.

Screenshots of the process:

Getting the frequency of letters:

Table

Description automatically generated

A picture containing shape

Description automatically generated

Table

Description automatically generated with medium confidence

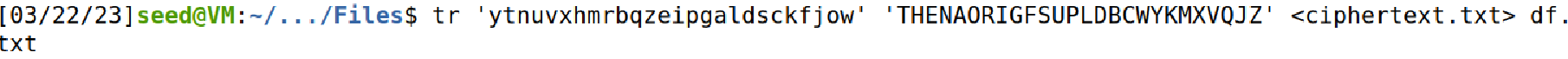
Using the tr command:



At last we will get a encryption key for our cipher text for our lab bellow is the key:

‘ytnuvxhmrbqzeipgaldsckfjow = ‘THENAORIGFSUPLDBCWYKMXVQJZ’

Running the tr command using the key:



Task 2: Encryption using Different Ciphers and Modes

Ciphertype: “ -bf-cbc”

Graphical user interface, text, application

Description automatically generated

Ciphertype: “-aes-128-cfb”

Text

Description automatically generated

Ciphertype: “-aes-128-cbc”

Graphical user interface, text, application

Description automatically generated

Task 6.1:



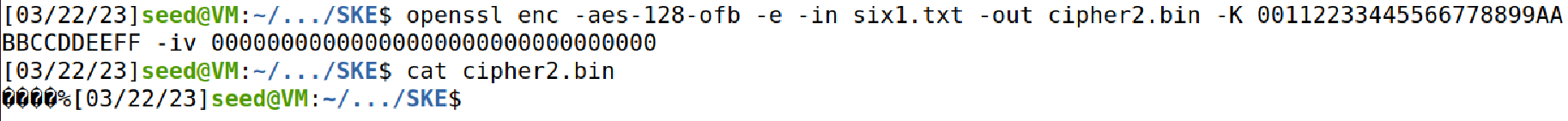
Running commands;

Text

Description automatically generated

Two different ciphers for different iv values.

Running the same file with first iv values and getting same ciphers:



Task 6.2:

Program:

#!/usr/bin/python3

from sys import argv

script , first , second= argv

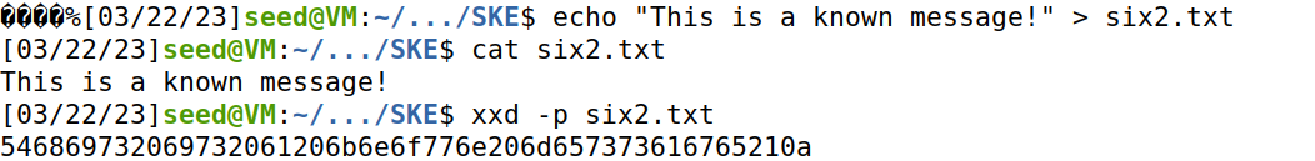
aa = bytearray.fromhex( first)

bb = bytearray.fromhex(second)

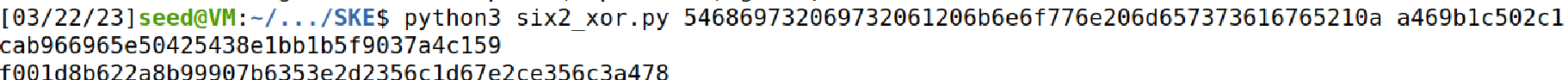
xord = bytearray(x^y for x , y in zip (aa , bb))

print(xord . hex ())

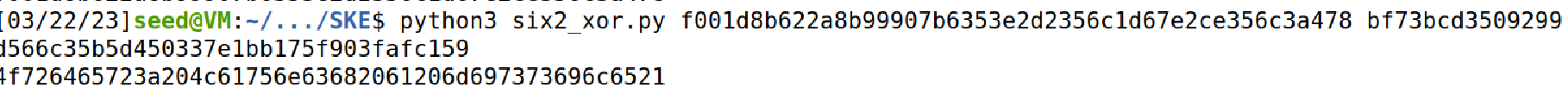
Creating a text file and converting text to hexadecimal



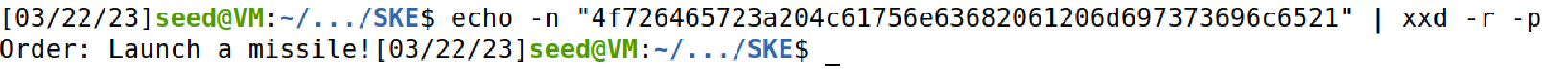
Running xor on above hexadecimal and c1 value given in problem statement:



Running XOR on above output and c2 value:



Reversing the hexadecimal to ascii:



Task 6.3:

Program:

#!/usr/bin/python3

from sys import argv

script , first , second= argv

aa = bytearray.fromhex( first)

bb = bytearray.fromhex(second)

xord = bytearray(x^y for x , y in zip (aa , bb))

print(xord . hex ())

Starting netcat:

Text

Description automatically generated

Running XOR with IV values:

Graphical user interface, text, application, email

Description automatically generated

Pasting the iv value in netcat screen:

Text

Description automatically generated

The cipher text we got is same as bob’s ciphertext and we will have an extended cipher which is for padding we did.

Task 7: Programming using the Crypto Library

Program:

from Crypto.Cipher import AES

from Crypto.Util import Padding

import re

data= b'This is a top secret.'

expected\_ciphertext = "3879c71b232cd0d2fc6f5ffcc1d76f074c0fcbe007d9cc53939fdeebf1d6ffd2"

with open('words.txt','r') as f:

for i in f:

if len(i) <= 16:

key\_string=i.rstrip().ljust(16,'#')

#print(key\_string)

key = bytes.fromhex(key\_string.encode('utf-8').hex())

iv\_hex\_string = 'aabbccddeeff00998877665544332211'

iv = bytes.fromhex(iv\_hex\_string)

data= b'This is a top secret.'

expected\_ciphertext = "3879c71b232cd0d2fc6f5ffcc1d76f074c0fcbe007d9cc53939fdeebf1d6ffd2"

cipher = AES.new(key, AES.MODE\_CBC, iv)

ciphertext = cipher.encrypt(Padding.pad(data, 16))

# print("Ciphertext: {0}".format(ciphertext.hex()))

if (ciphertext.hex() == expected\_ciphertext):

print(i)

Screenshot of the output:

Text

Description automatically generated