**Lab 8: Crypto Lab: Secret-Key Encryption**

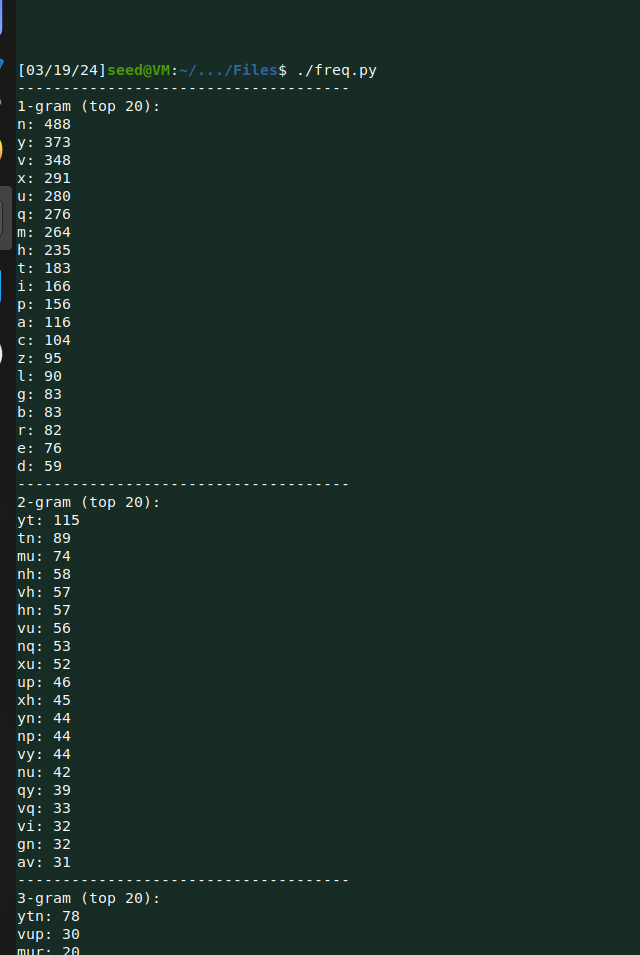
**Task 1: Frequency Analysis**

In this task, I have to find plain text from cipher text using frequency analysis. Given a cipher-text that is encrypted using a monoalphabetic cipher.

First, I ran the freq.py code to check frequency of each letter in cipher text.

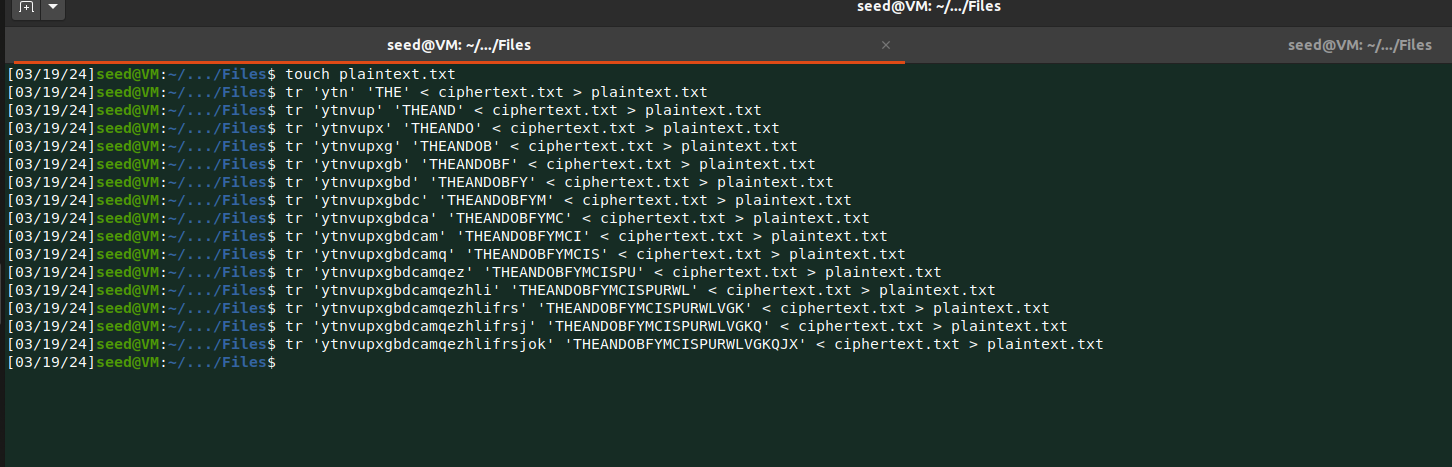
Then I used ***watch -n 5 cat plaintext.txt*** command to check the file in the interval of 5 seconds after I replace letter from cipher.txt to plaintext.txt.

Below shows the output from the freq.py program.

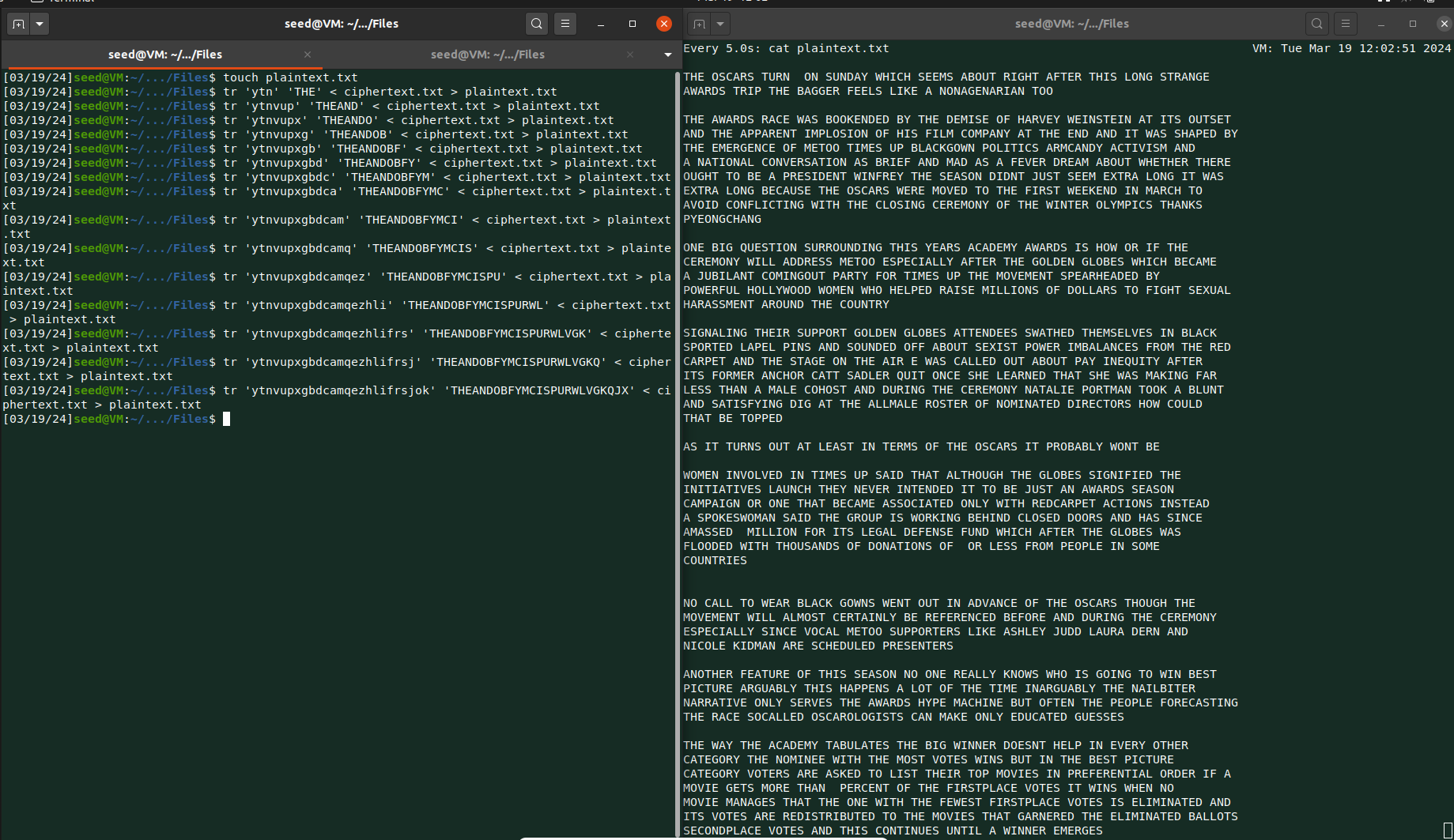


Below screenshot shows, how I decrypted the text one by one using frequency analysis.

Last command is the final one.



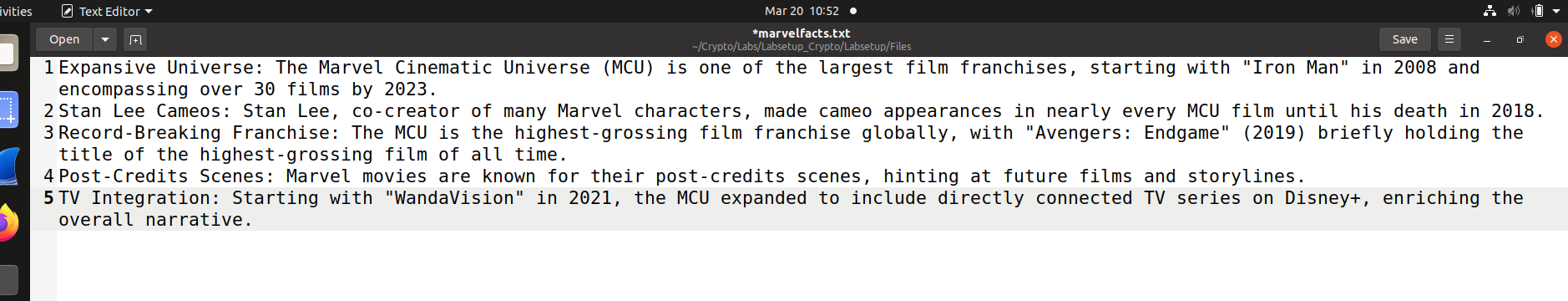
Below shows side by side decryption of message.



**Task 2: Encryption using Different Ciphers and Modes**

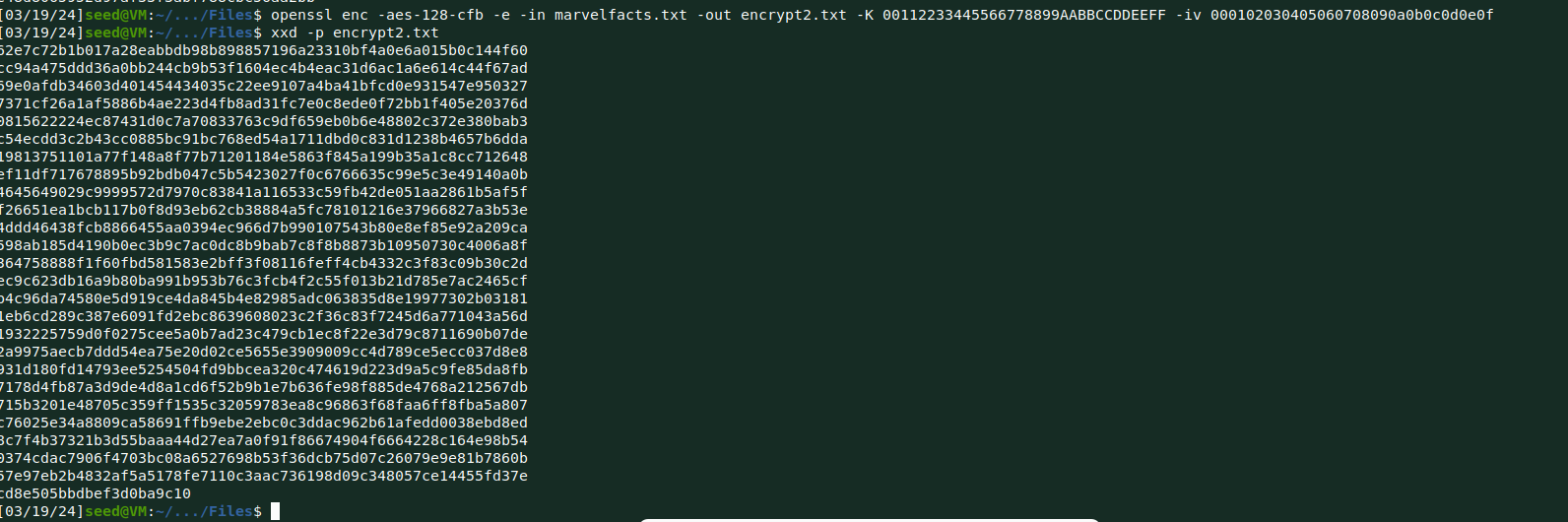
In this task, I used three different encryption algorithms and modes.

I chose one random text about Marvel comics.



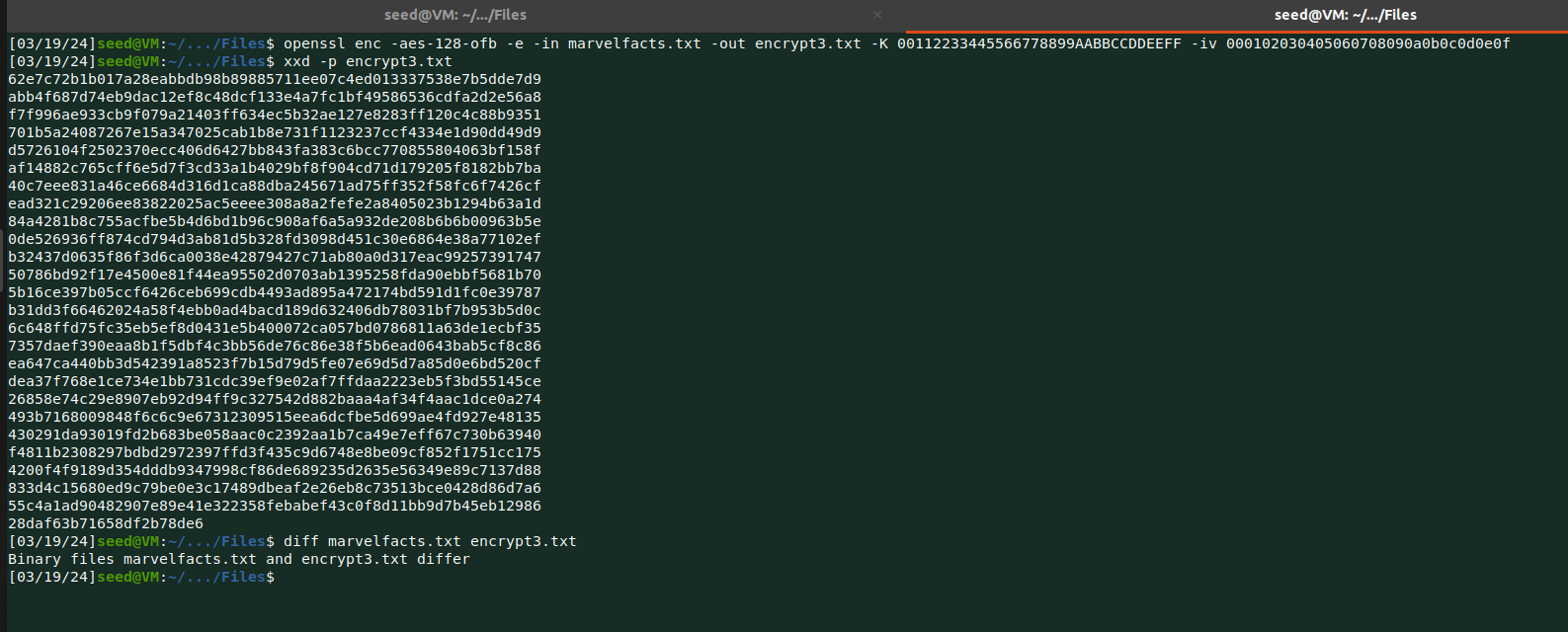
**aes-[128]-cfb :**

Need 16 byte key and initialization vector.



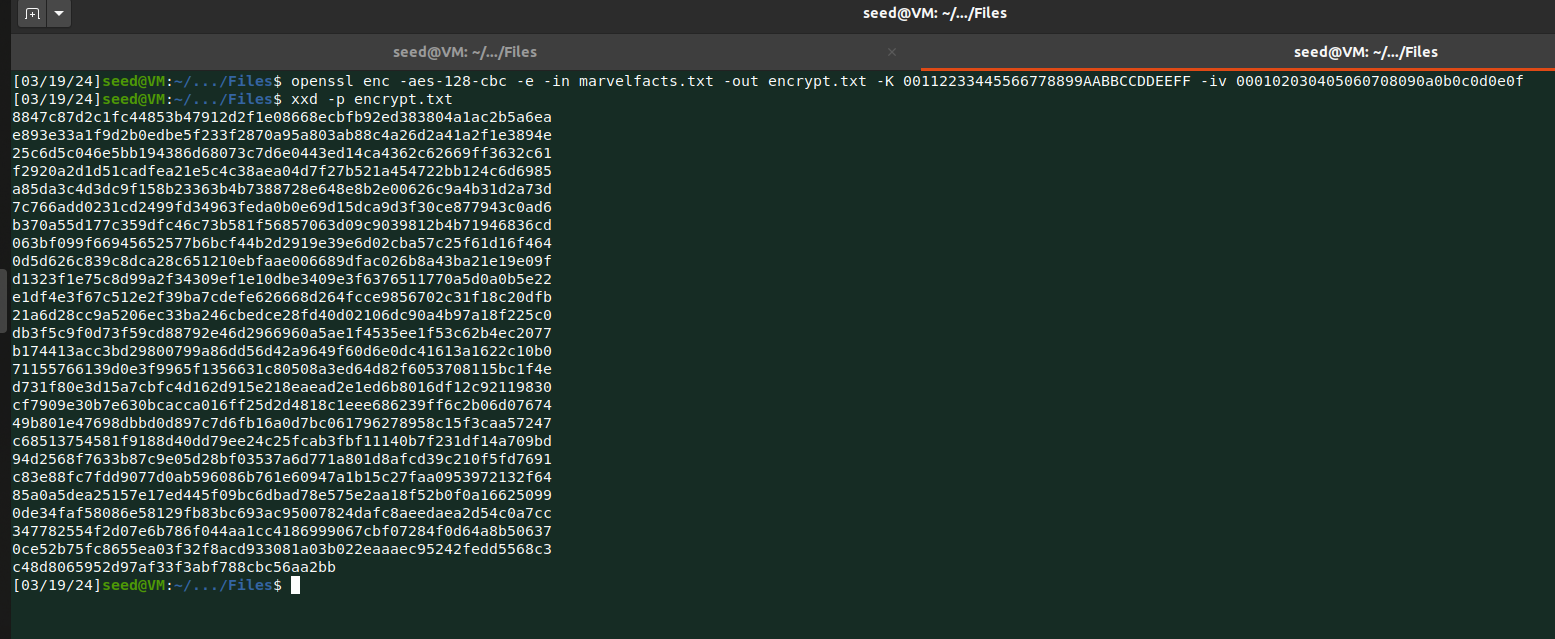
**aes-[128]-ofb :**

Need 16 byte key and initialization vector.



**aes-[128]-cbc :**

Need 16 byte key and initialization vector.



**Task 6: Initial Vector (IV) and Common Mistakes**

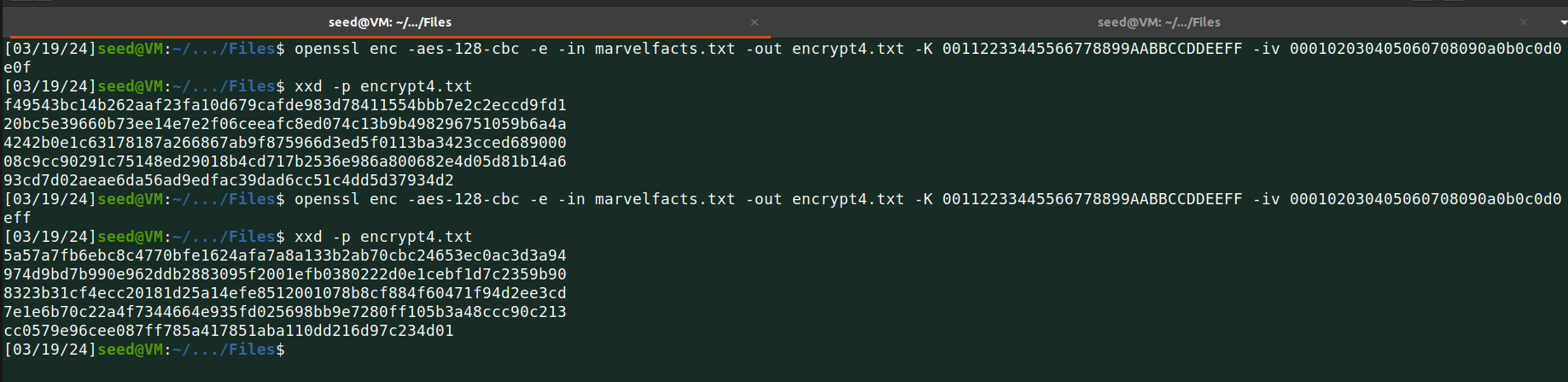
In this task, there are three subtasks which shows common mistakes related to initialization vector.

**Task 6.1:**

In this subtask, I used same IV in cbc mode and then I used different IV with same mode to check what is the difference.

**Using same IV:**

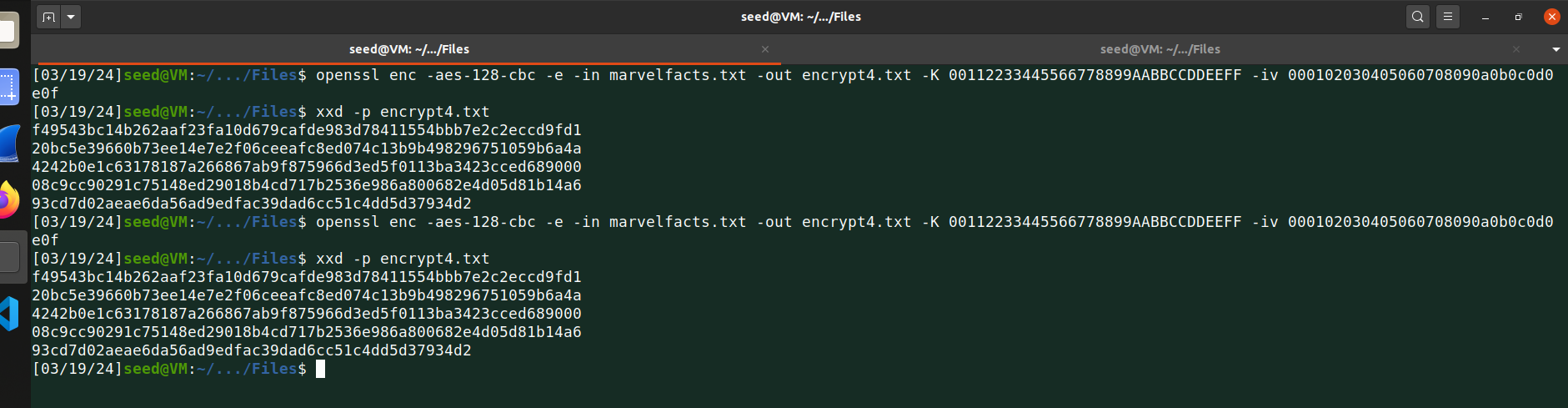
Below shows the output if I use same IV twice.



This illustrates that encrypting identical plaintext with the same Initialization Vector (IV) results in identical ciphertext, irrespective of the circumstances, provided that the encryption mode and key remain constant. Consequently, this creates susceptibility to security breaches. Therefore, it is essential to avoid reusing an IV with the same key.

**Using different IV:**

Below shows the output if I use different IV twice.



Despite utilizing the same key, employing distinct Initialization Vectors (IVs) to encrypt the same plaintext resulted in significantly different ciphertexts. This demonstrates that varying the IV for each encryption yields diverse outcomes.

**Task 6.2:**

In this subtask, I was able to find P2 if I know the value of P1, C1 and C2.   
Sample code is provided which I updated as per need.

It’s nothing but just XOR operation performed on P1, P2, C1 and C2.

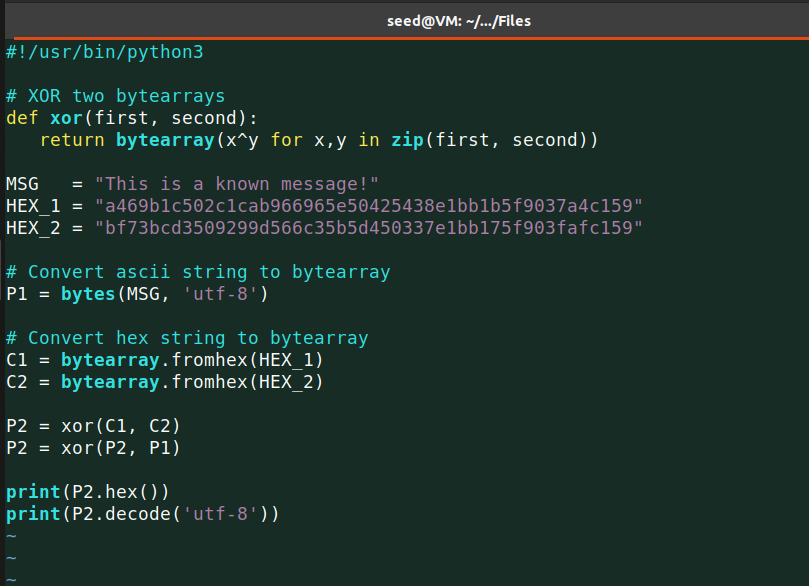
As professor taught in lecture for the equation,

P1 XOR P2 = C1 XOR C2

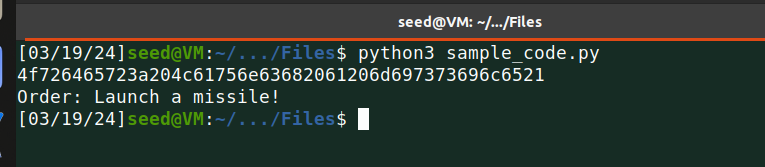
P2 = C1 XOR C2

P2 = P2 XOR P1

**Code:**



Then after I ran the code with provided values then I got the output : ***Order Launch a missile!***

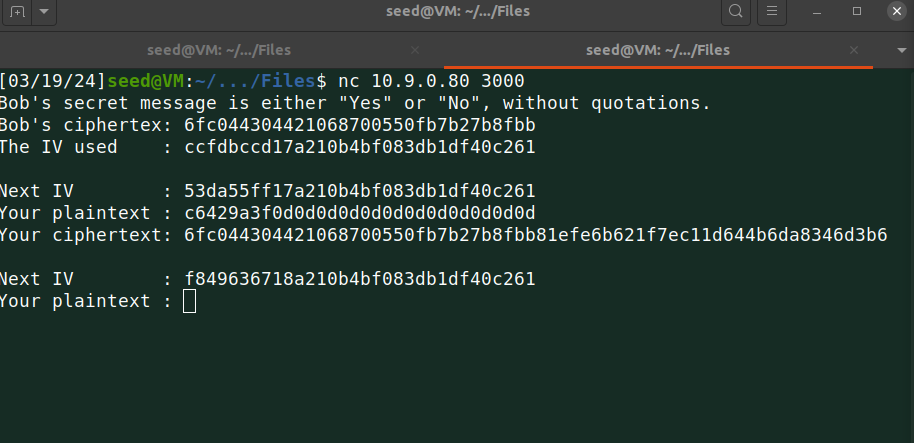


In CFB mode, the scenario for the initial block is similar, allowing for the plaintext to be derived through a simple XOR operation. Nonetheless, provided the key is kept confidential, the subsequent segments of the ciphertext remain secure and undisclosed.

**Task 6.3:**

In this subtask, I have to predict what Bob’s actual content is Yes or No.

First, need to connect to oracle for that I started docker container and then using netcat command I connected to oracle on port 3000.



Above screenshot shows first Next IV and the IV used by Bob initially.

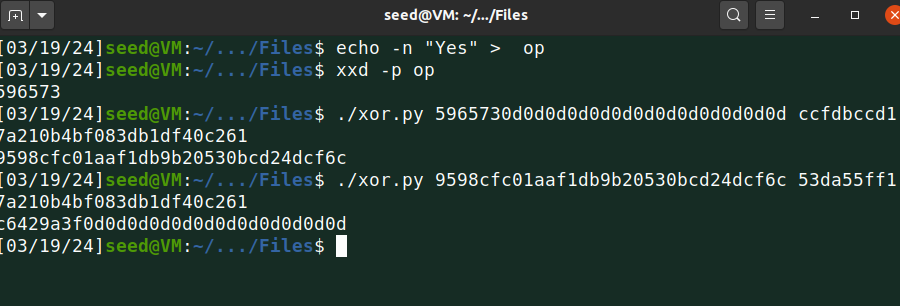
Using xor.py code which was provided by professor in class material.

First need to find hex value of “YES” as I am guessing that’s the vote from bob.

I need to pad this value as I am using CBC mode, the value is 596573 with the size of 3 bytes so I need to add 13 more bytes of padding (0d 13 times).

Now passing hex value of YES and IV used by Bob to xor.py.

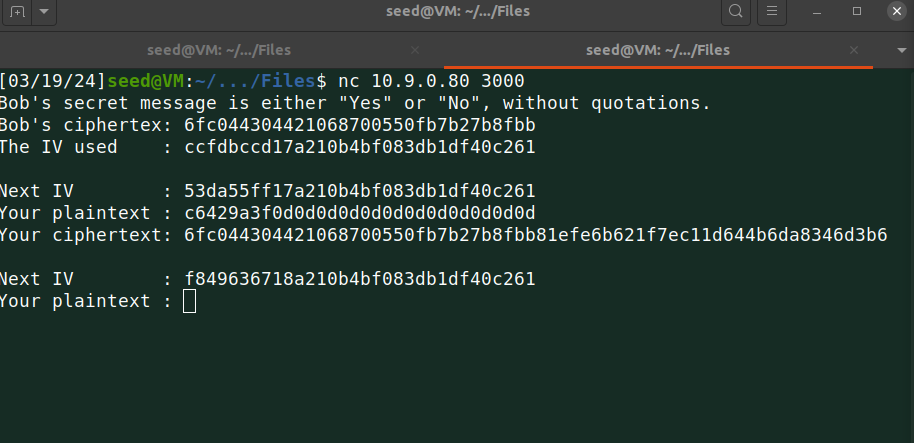
Next I used result from above with Next IV provided by oracle.



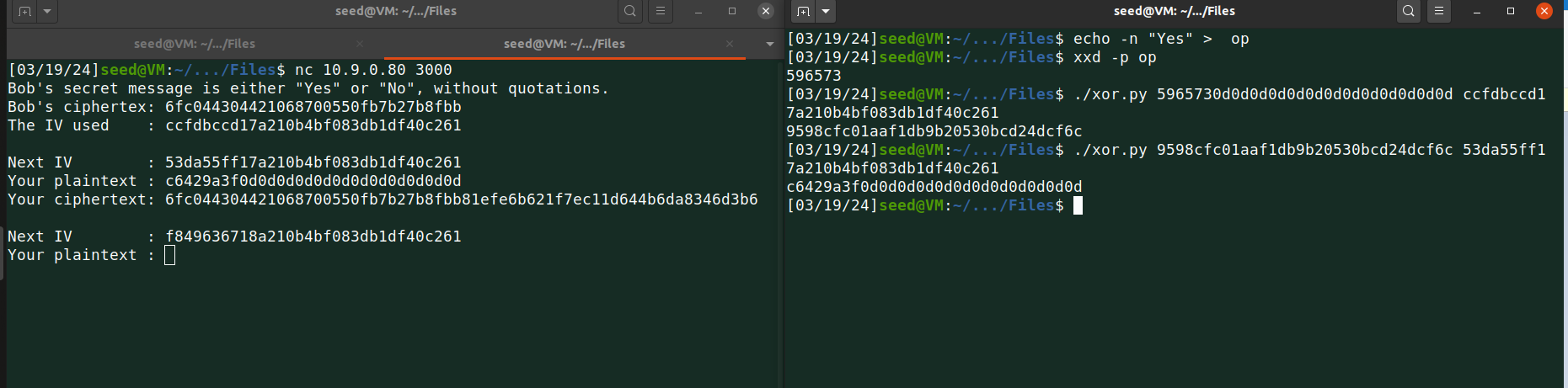
Now it’s time to use that result in below input as my plaintext.

And immediately I got the result in your ciphertext, if I omit extra padded digits from that I can see generated cipher text is similar to Bob’s ciphertex.

So bob’s message was Yes.



Below shows whole operation side by side.



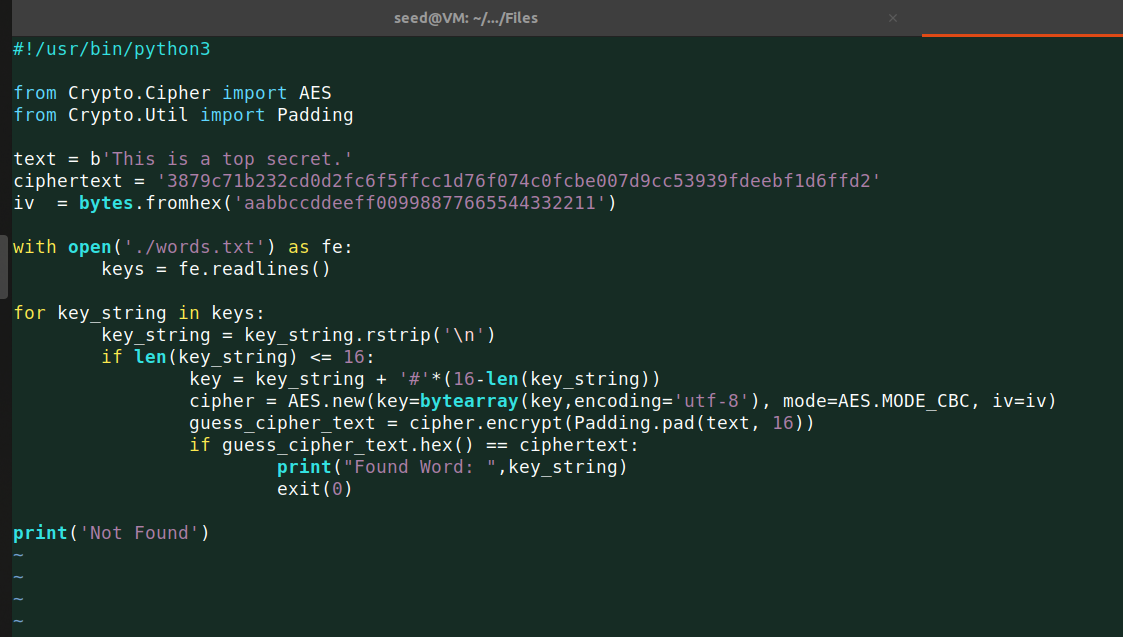
**Task 7:**

In this task, I have given a plaintext and a ciphertext and my job is to find the key which is used for the encryption.

Professor showin this demo in class, and I used same approach to find the key but with some updated logic as I have to read this from words.txt file.

Plaintext is : This is a top secret.

**Code:**



I ran the above code and found output: **Purdue**.

