**Will Pond**

**TASK1**

**After downloading the Lab setup files for last name beginning with L to Z because my last begins with a P. I create a python file name task1 and write a program to generate the encryption key.**

**A screen shot of a computer

Description automatically generated**

**After running task1.py**

**A black screen with green text

Description automatically generated**

**This is the contents of the words.txt file look like.**

**A screen shot of a computer

Description automatically generated**

**Making some simplification to the words.txt file by creating two new files called lowercase.txt and plaintext.txt. lowercase.txt is a copy of words.txt but it converts all the uppercase letters in word.txt to lowercase and plaintext.txt is a copy of lowercase.txt which take out of all the punctuations and numbers. These are the commands I ran to make the modifications**

**A screen shot of a computer code

Description automatically generated**

**Contents of lowercase.txt**

**A screen shot of a computer

Description automatically generated**

**Contents of plaintext.txt**

**A screen shot of a computer

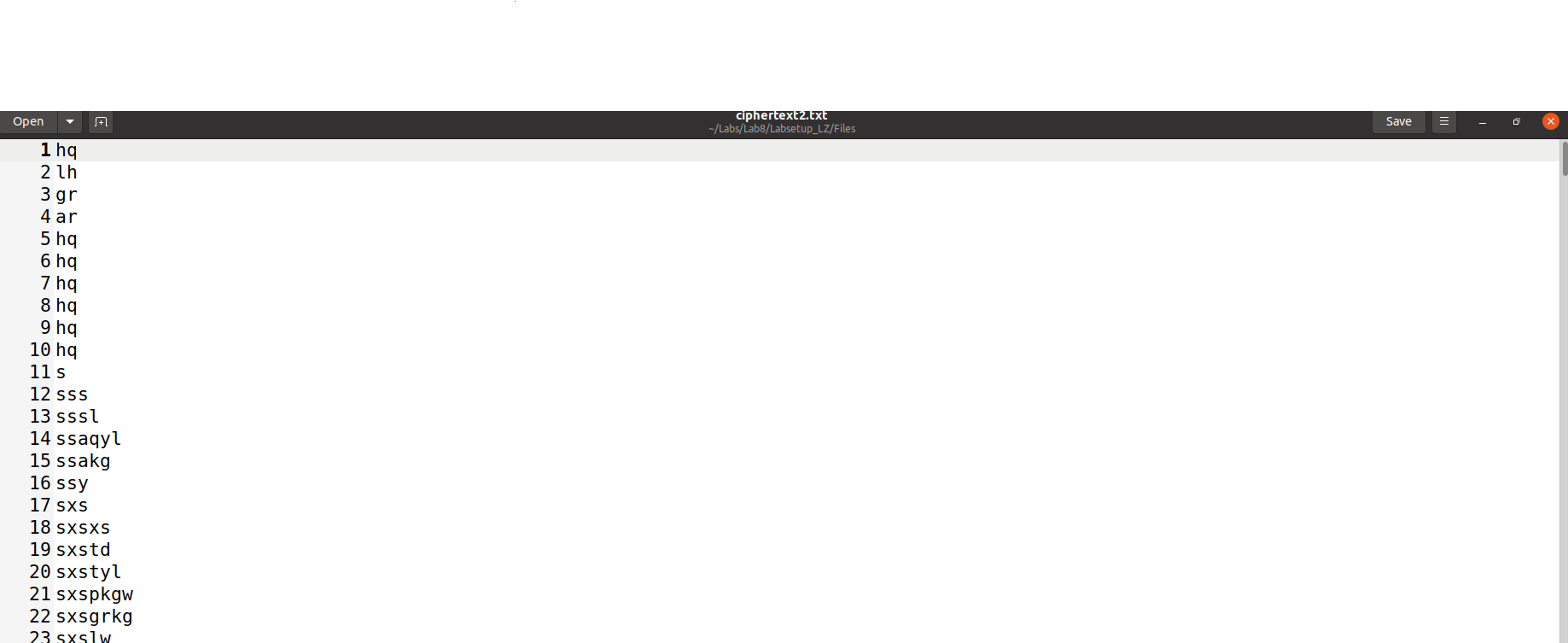
Description automatically generated**

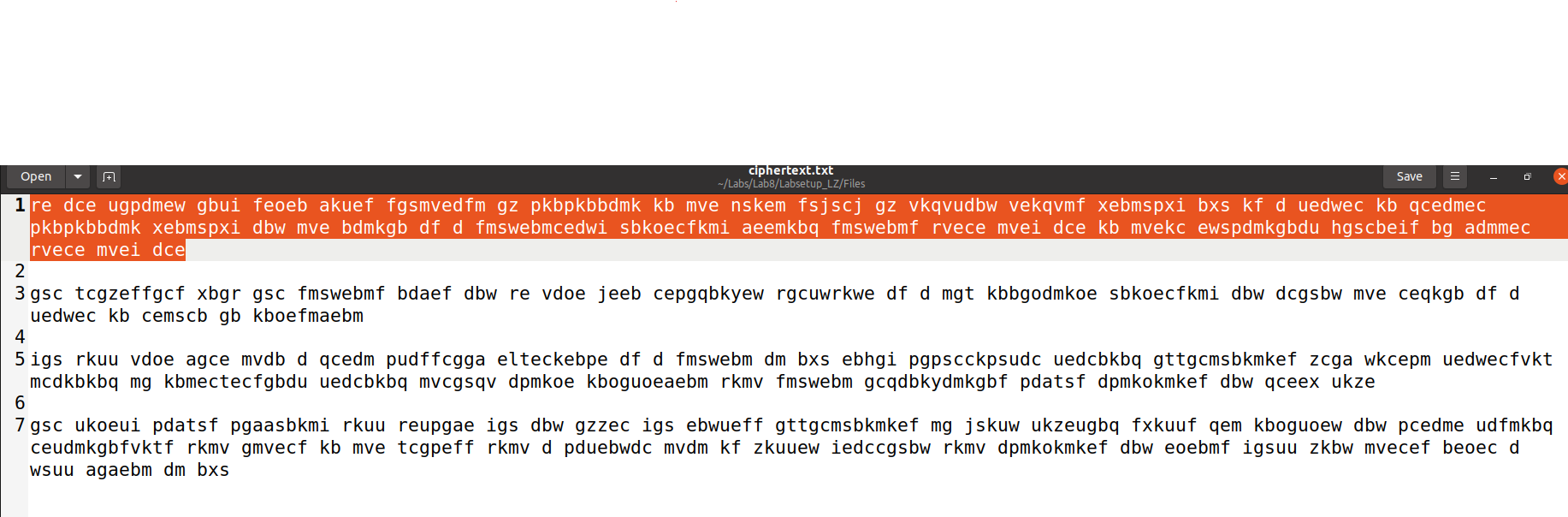
**Using the tr command to run the encryption of plaintext.txt and name it ciphertext2.txt**

**A black screen with text

Description automatically generated**

**The contents of ciphertext2.txt**

****

**Contents of ciphertxt.txt**

**The code of program freq.py**

**A screenshot of a computer

Description automatically generated**

**Running freq.py to get the statistics for n-grams**

**A screenshot of a computer

Description automatically generated**

**The Key to the decrypted message after guessing the correct letters.**

**A screen shot of a computer

Description automatically generated**

**The Encryption key in alpehbet order is 'djpwezqvkhxuabgtncfmsorliy' from 'abcdefghijklmnopqrstuvwxyz'**

**Final message**

**A screenshot of a computer

Description automatically generated**

**TASK2**

**Using Cipher Bock Chaining CBC to encrypt**

**A black screen with white text

Description automatically generated**

**Contents of of cbc\_cipher.bin when encrypted**

**A screenshot of a computer

Description automatically generated**

**Using this command to decrypt**

**A black screen with white text

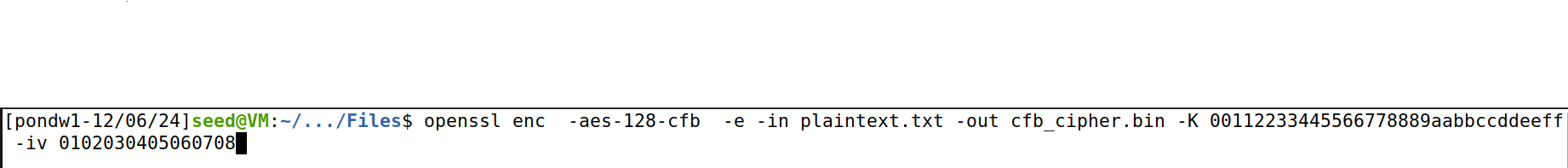
Description automatically generated**

**Contents of cbc\_plain.txt**

**A screenshot of a computer

Description automatically generated**

**Using Cipher Feedback CFB to encrypt**

****

**Contents of of cfb\_cipher.bin when encrypted**

**A screenshot of a computer

Description automatically generated**

**Using this command to decrypt**

**A black screen with text

Description automatically generated**

**Contents of cfb\_plain.txt**

**A screenshot of a computer

Description automatically generated**

**Using Output Feedback OFB to encrypt**

**A black screen with white text

Description automatically generated**

**Contents of ofb\_cipher.bin**

**A black and white screen

Description automatically generated**

**Using this command to decrypt**

**A black screen with white text

Description automatically generated**

**Contents of ofb\_plain.txt**

**A screenshot of a computer

Description automatically generated**

**TASK3**

**Picture from the Lab to Encrypt**

**A red circle on a white background

Description automatically generated**

**The command use to encrypt the picture using ECB and CBC Mode**

**A black screen with a black background

Description automatically generated**

**The command use to reset the header and body of the encrypted picture for both Modes and combing them.**

**A screen shot of a computer

Description automatically generated**

**Viewing the encrypted image comb\_ecb.bmp by using the eog comb\_ecb.bmp command**

**A screen shot of a screen

Description automatically generated**

**Viewing the encrypted image comb\_cbc.bmp by using the eog comb\_ecb.bmp command**

**A black and green rectangle with a black background

Description automatically generated**

**OBSERVATION OF THE TWO ENCRYPTED IMAGES**

**In the first image I can somewhat see the image. This is because the file breaks into 128 bits and uses AES algorithm to encrypt each one. For some reason the two blocks have the same original image that they will remain identical in the encrypted. The second Image I cannot see because of what CBC model Encryption is doing to encrypt the photo.**

**Redoing the process again with my own image. This image is named Path.bmp and this is what it looks like**

**A field with blue lights

Description automatically generated**

**The command is used to encrypt the picture using ECB and CBC Mode**

**A black screen with text

Description automatically generated**

**The command use to reset the header and body of the encrypted picture for both Modes and combing them.**

**A screenshot of a computer screen

Description automatically generated**

**View the image of using CBC and ECB**

**A screen shot of a television screen

Description automatically generated**

**Observation**

**I cannot see the image for CBC but also for ECB because it is not using same identical block of encrypted so it will come out look like this.**

**TASK4**

**Created test.txt and has 6 bytes and doing ECB encryption to created a output.bin which contain 16 bytes meaning that it does requires Padding**

**A screenshot of a computer

Description automatically generated**

**Repeating the process again with CBC Encryption which ended up with 16 bytes meaning that it does require Padding.**

**A screen shot of a computer

Description automatically generated**

**Repeating the process again with CFB Encryption which ended up with 6 bytes meaning that it does not require Padding. This is because it takes the output when encrypted from the previous block to make it the same size as it last cipher block encryption**

**A screen shot of a computer

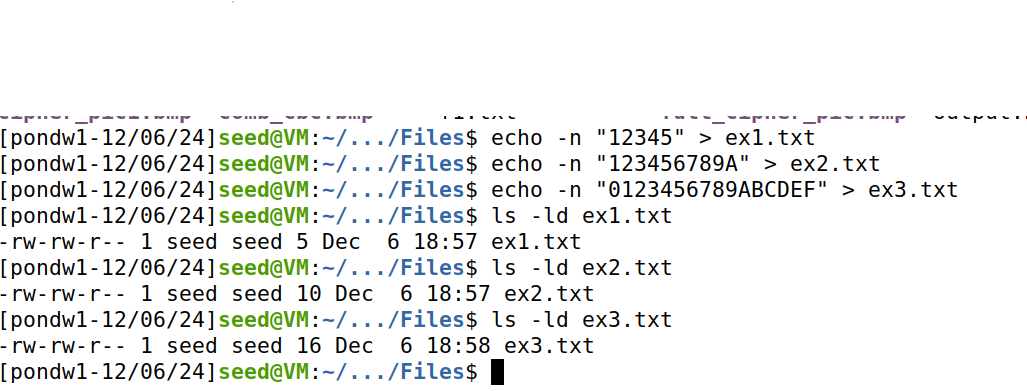
Description automatically generated**

**Repeating the process again with OFB Encryption which ended up with 6 bytes meaning that it does not require Padding. This is because of the same reason with CFB Encryption**

**A screenshot of a computer

Description automatically generated**

**Created three files with a 5 byte for ex1.txt, a 10 byte for ex2.txt and 16 byte for ex3.txt**

****

**Encrypt the 3 files with CBC mode and shows that output.bin files of the first two contain 16 bytes but the last one contains 32 bytes**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Decrypt the output.bin files with nopad**

**A black screen with white text

Description automatically generated**

**The output of files and the padding during encryption are being treated as ciphertext**

**A screenshot of a computer

Description automatically generated**

**TASK 6.2**

**Question**

**Assume that the attacker gets hold of a plaintext (P1) and a ciphertext (C1) ,  
can he/she decrypt other encrypted messages if the IV is always the same?**

**Yes**

**Code for sample\_code.py**

**A screen shot of a computer

Description automatically generated**

**Modified Code for sample\_code.py get the hex of P2 to get then the secret key**

**A screenshot of a computer

Description automatically generated**

**Running sample\_code.py to get this message of P2**

**A computer screen with numbers and a black background

Description automatically generated**

**QUESTION**

**If we replace OFB in this experiment with CFB (Cipher Feedback), how much P2 can be revealed?**

**In the CFB mode it can reveal stuff like OFB but it will be in the same situation with first block getting plaintext by the XOR getting a significant portion of the text. Also, if the key remains secret then parts of the ciphertext will not be revealed.**

**LAB REFLECTION**

**In this lab I was able to learn secret key encryptions, how to encrypt decrypted and perform attacks on encryption. In TASK1 I was able generate an encryption key from a python file then I was able to convert text using the tr command to decrypt encrypted messages. Then I was able to do a Frequency Analysis to find out the encryption key and the original plain text by looking at the frequency letters use and guess some words that maybe in the encoded message.**

**In TASK2 I Performed three different cipher modes to encrypted CFB CEB and OFB and see the decryption results of each of them. In TASK3 I was able to encrypt images using ECB and CBC while noticing the difference between the two when cipher them. TASK4, I experimented with different file bytes and understood why padding is needed for some Cipher modes to do encryptions and what padding is added to the files using nopad command when decrypting. Finally in TASK 6.2 I experimented with using Same IV to encrypt other messages by first getting plain text and XOR with a ciphertext, then take that expression and XOR with the same key + IV to plain text of another file. I was able to do this by modifying the python program called sample\_Code.py to get the unknown text which was “Order: Launch the Missile”.**