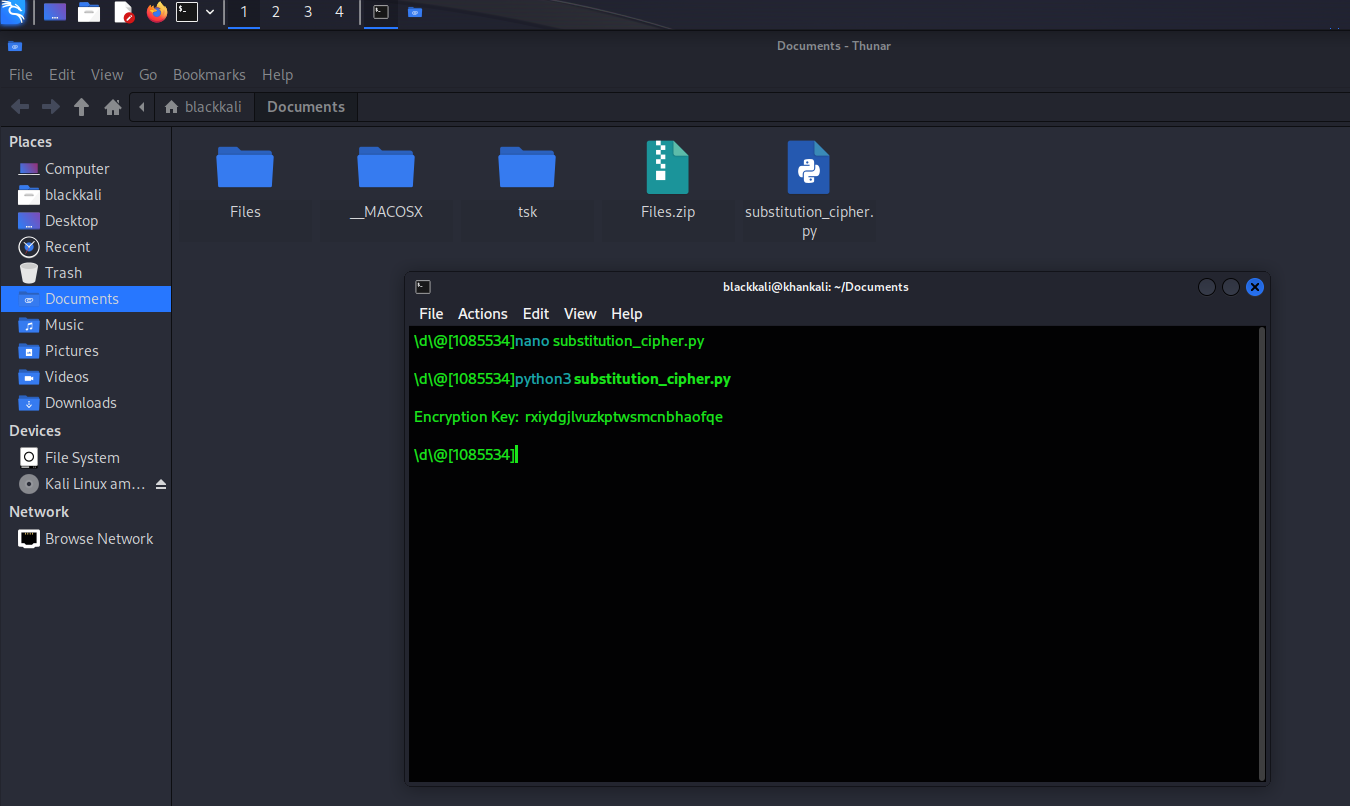
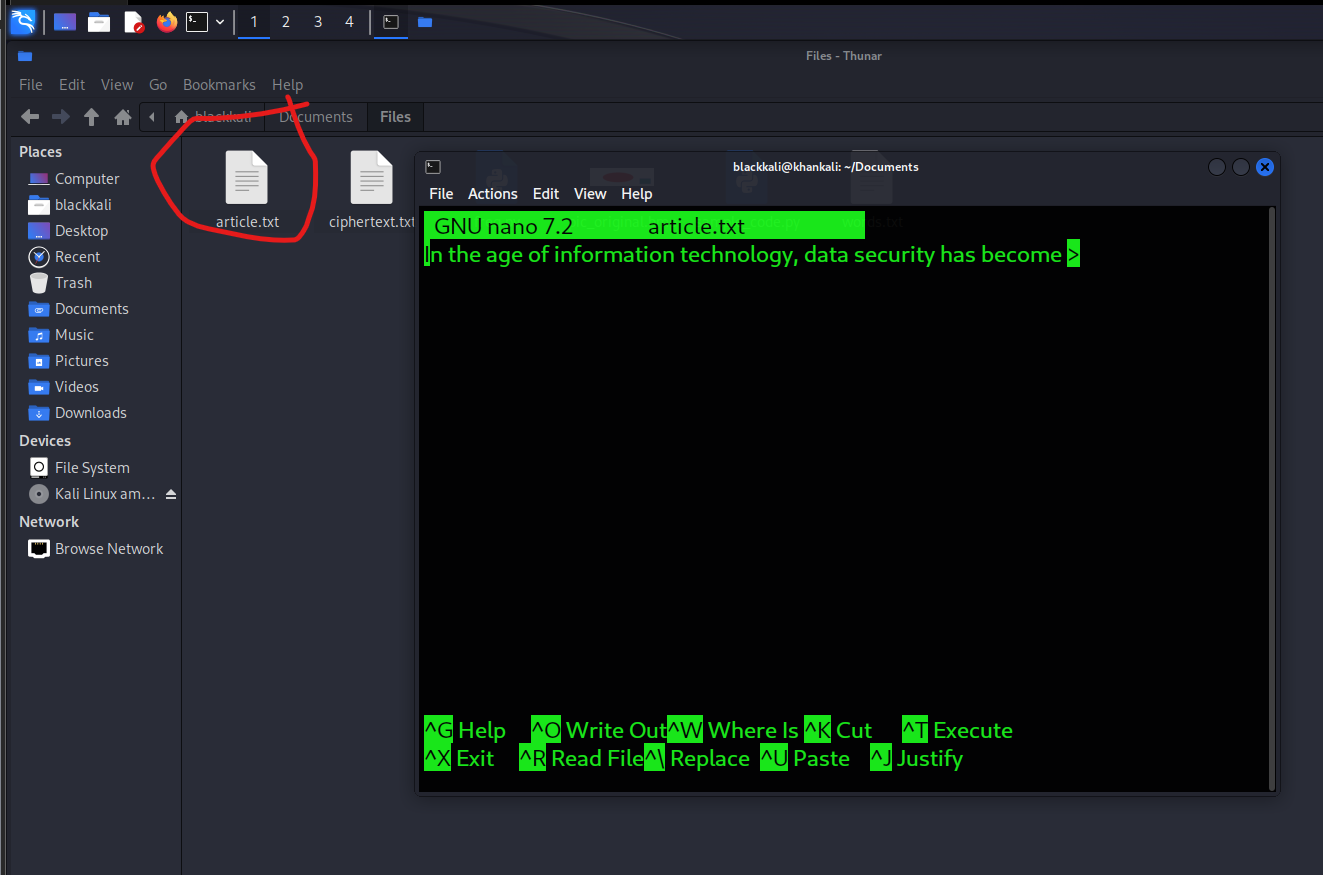


TASK1:

Generate the Encryption Key

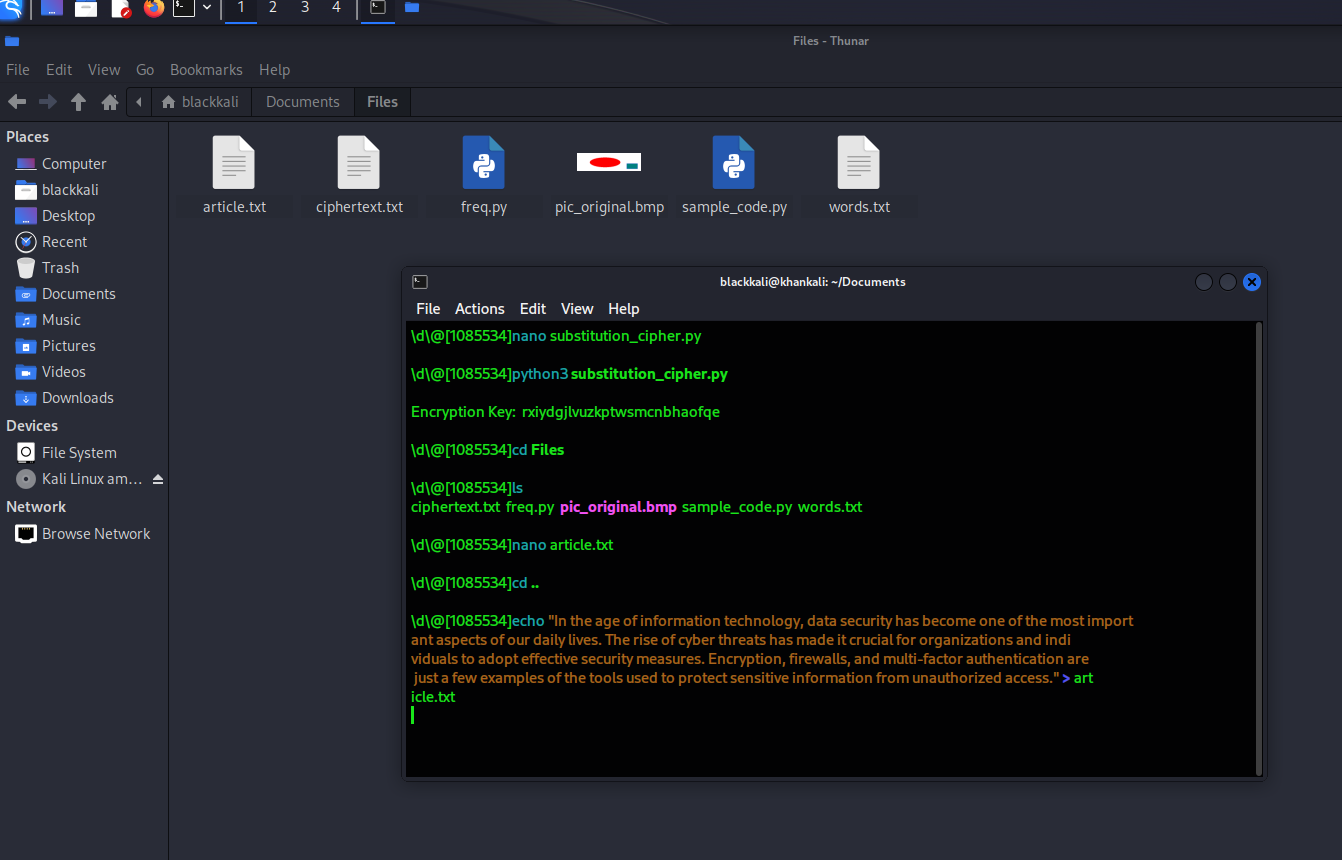


Articles.txt consist paragraph

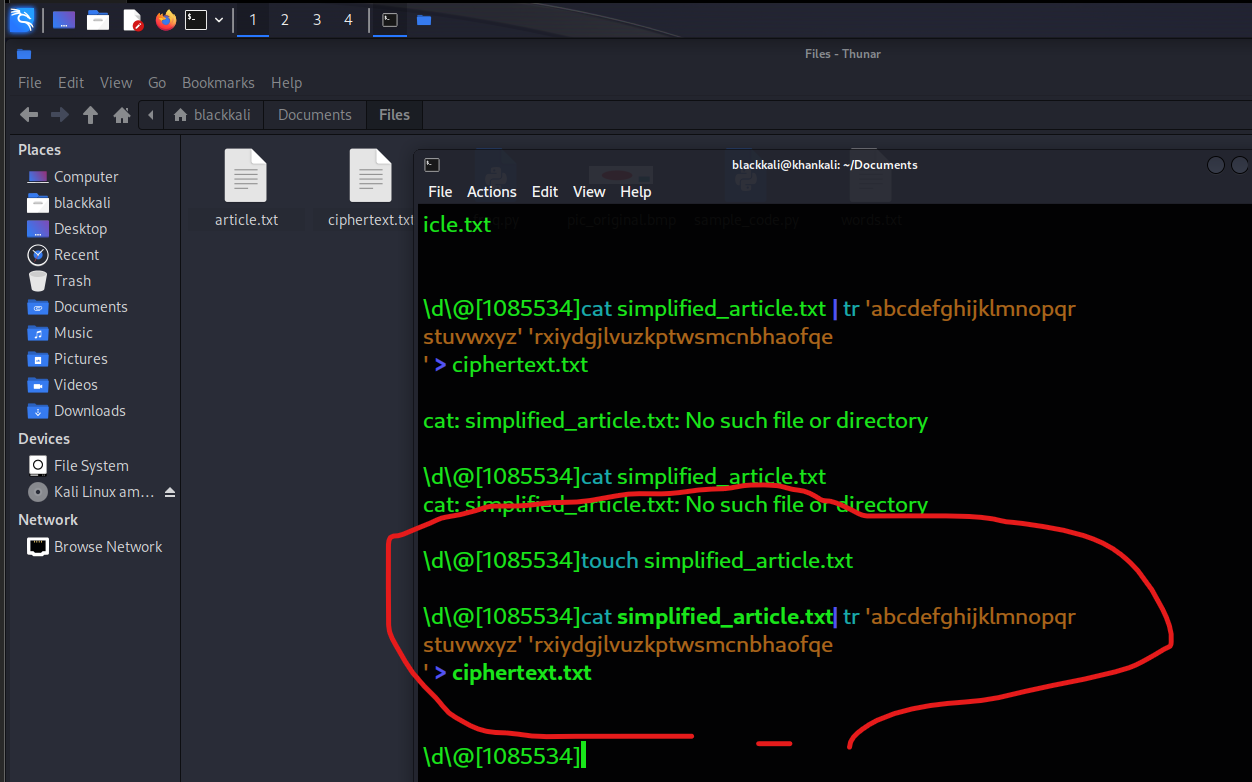


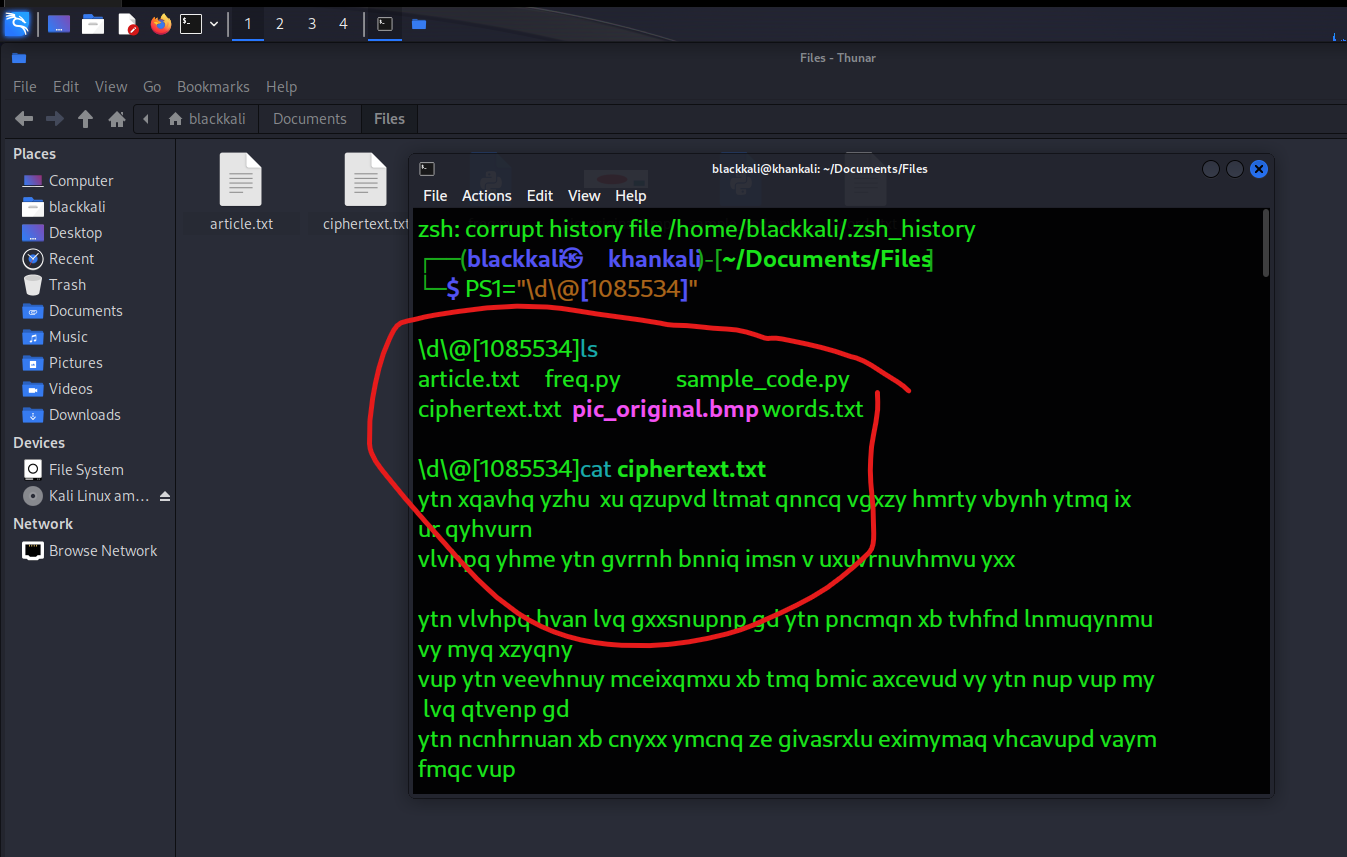
Run the Python file

python3 substitution\_cipher.py

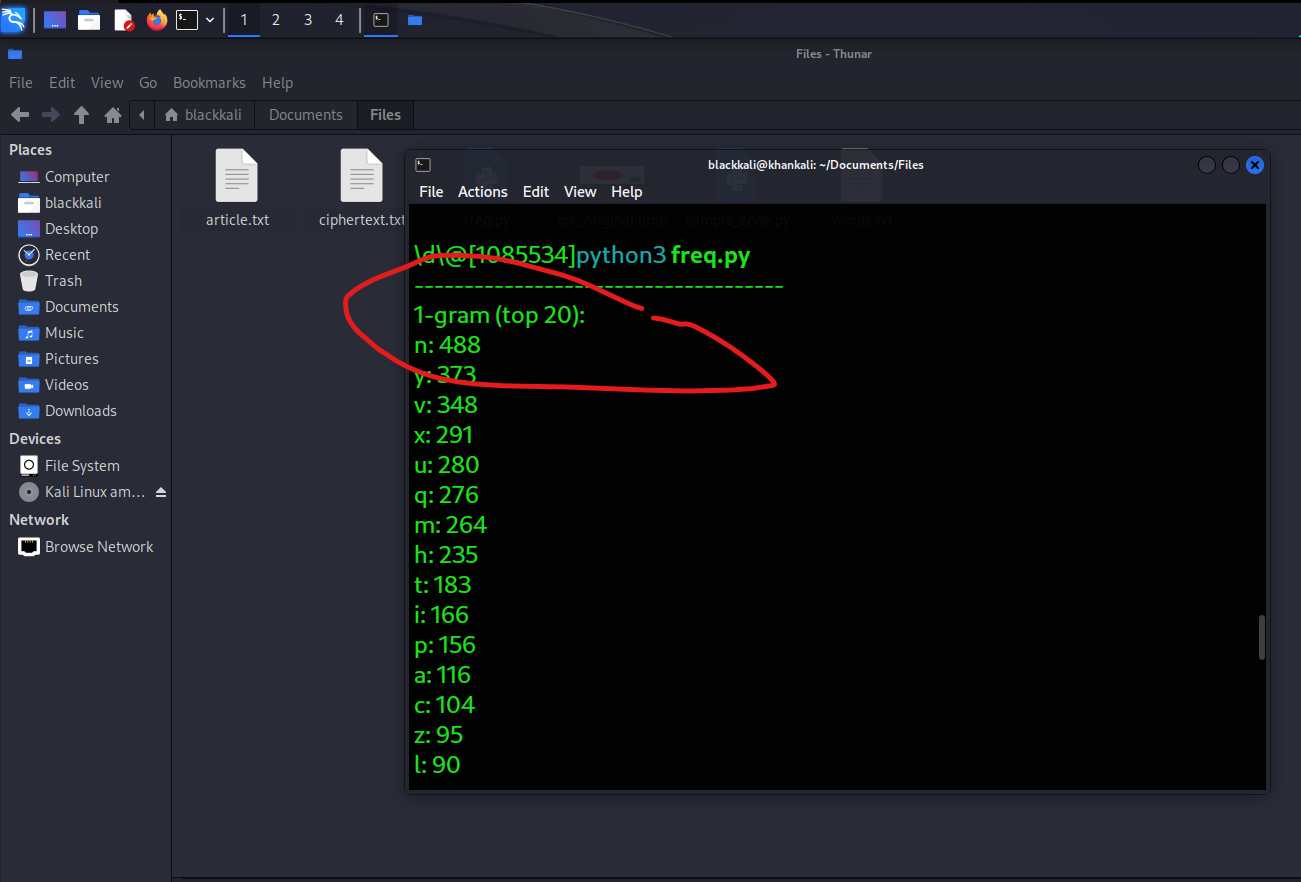


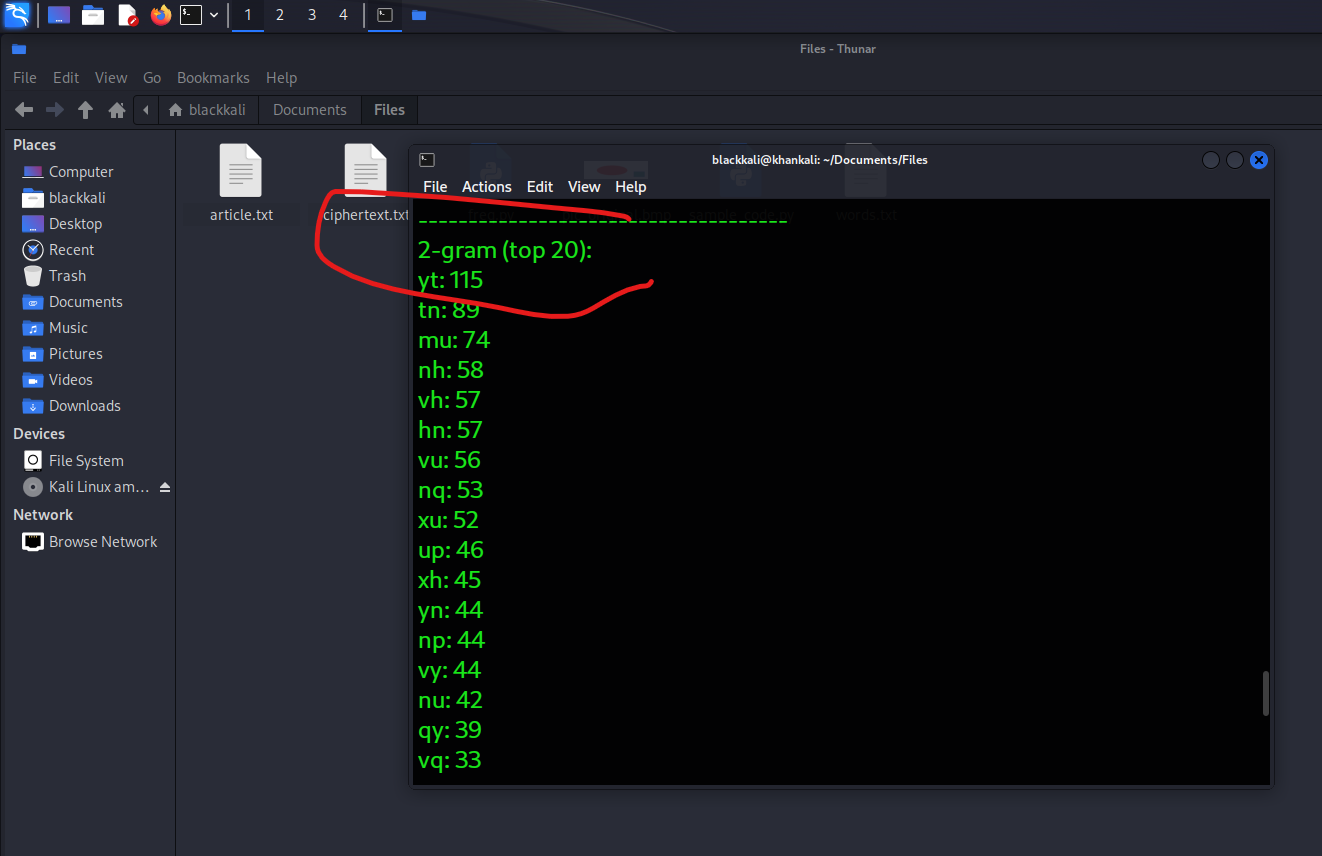
Step 2: Create and Simplify the Text

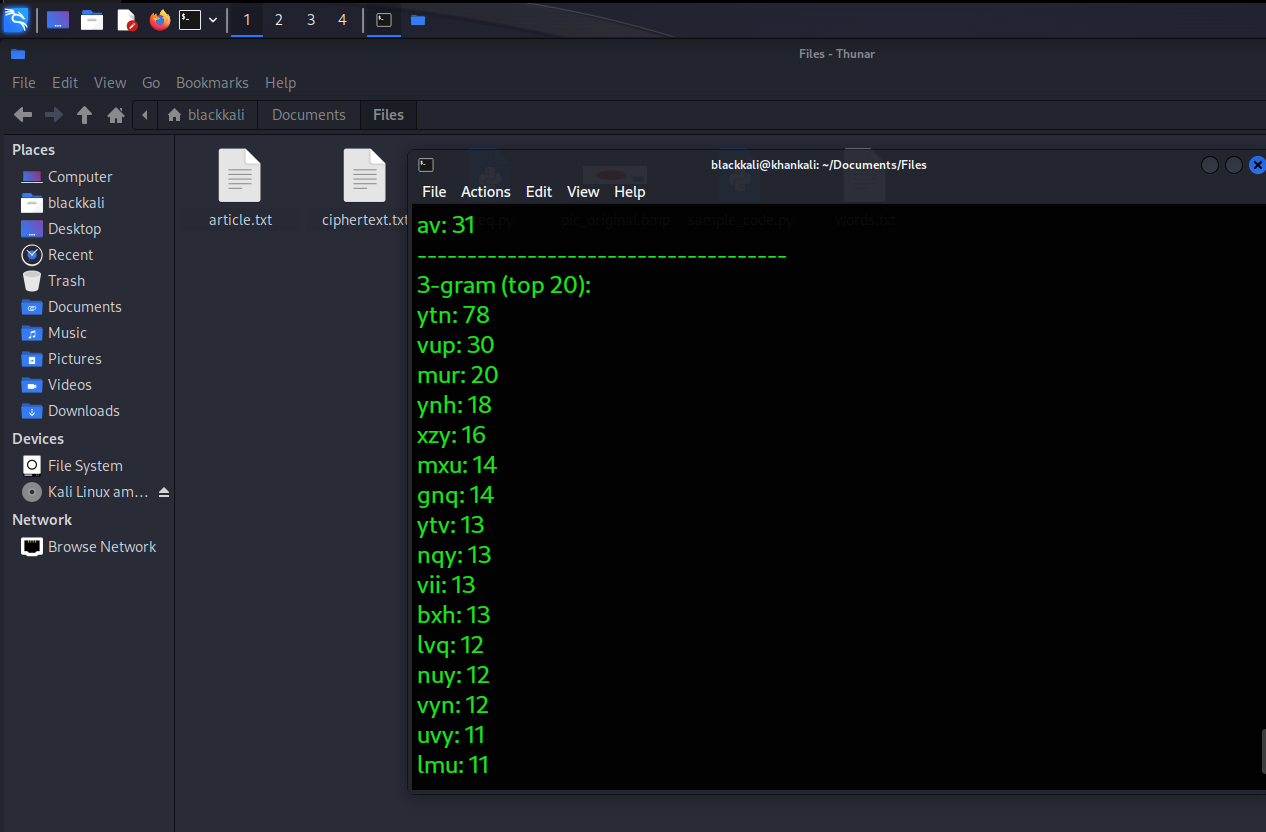




TASK2:







Analyze the Frequency Results

**1-gram Frequencies**:

* The most frequent letters are: n, y, v, x, u, q, m.

the most frequent letters are usually: **E, T, A, O, I, N, S**. So, you can start by assuming that:

* n ---> **E** n is the most frequent char
* y ---> **T**
* v ---> **A**

**2-gram Frequencies**:

* The most frequent 2-grams (pairs of letters) are: yt, tn, mu, nh, vh.
* In English, common 2-grams are: TH, HE, IN, ER, AN, RE.

Using this, you could map:

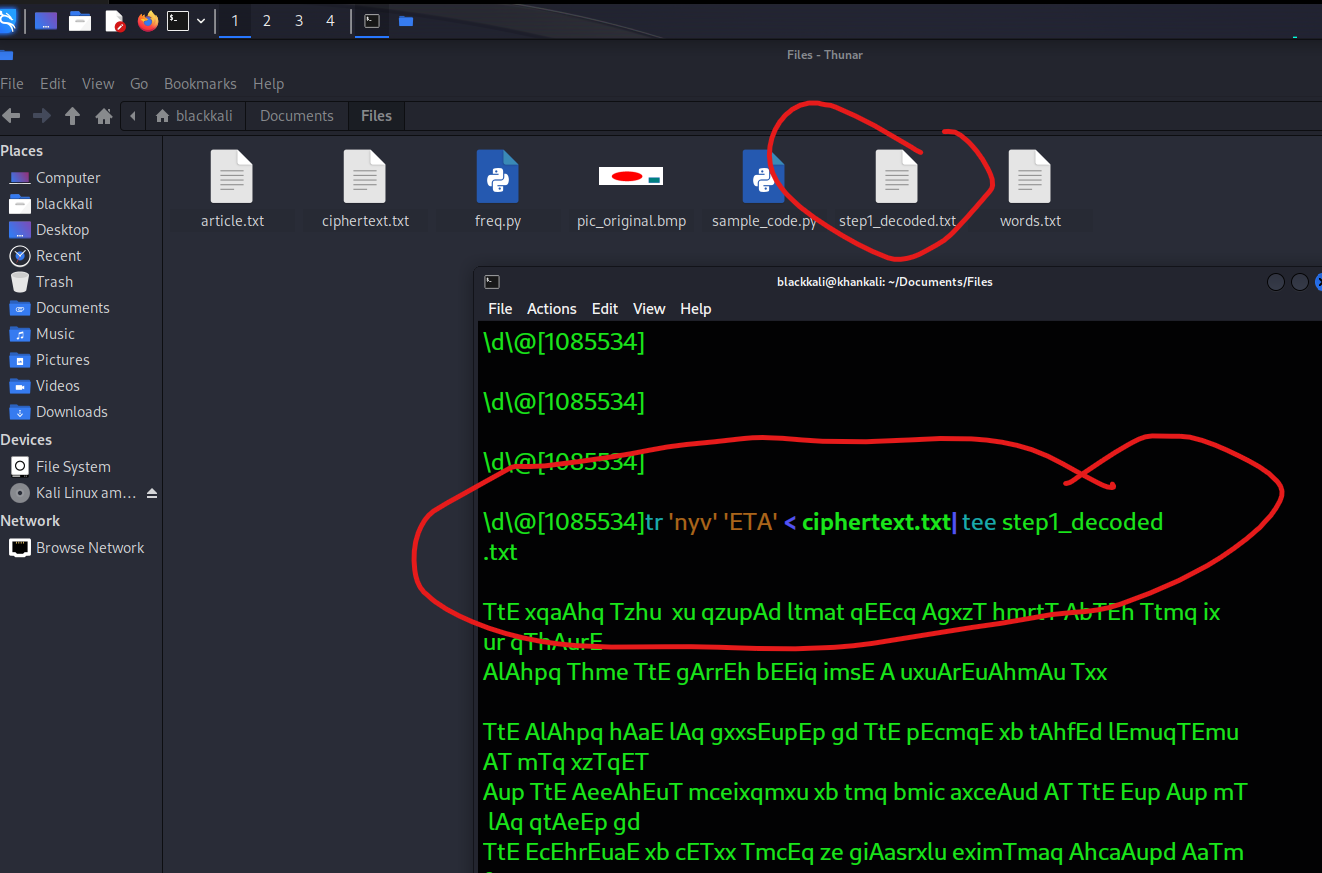
* yt ---> **TH** since yt is the most frequent
* tn ---> **HE**
* mu ----> **IN**

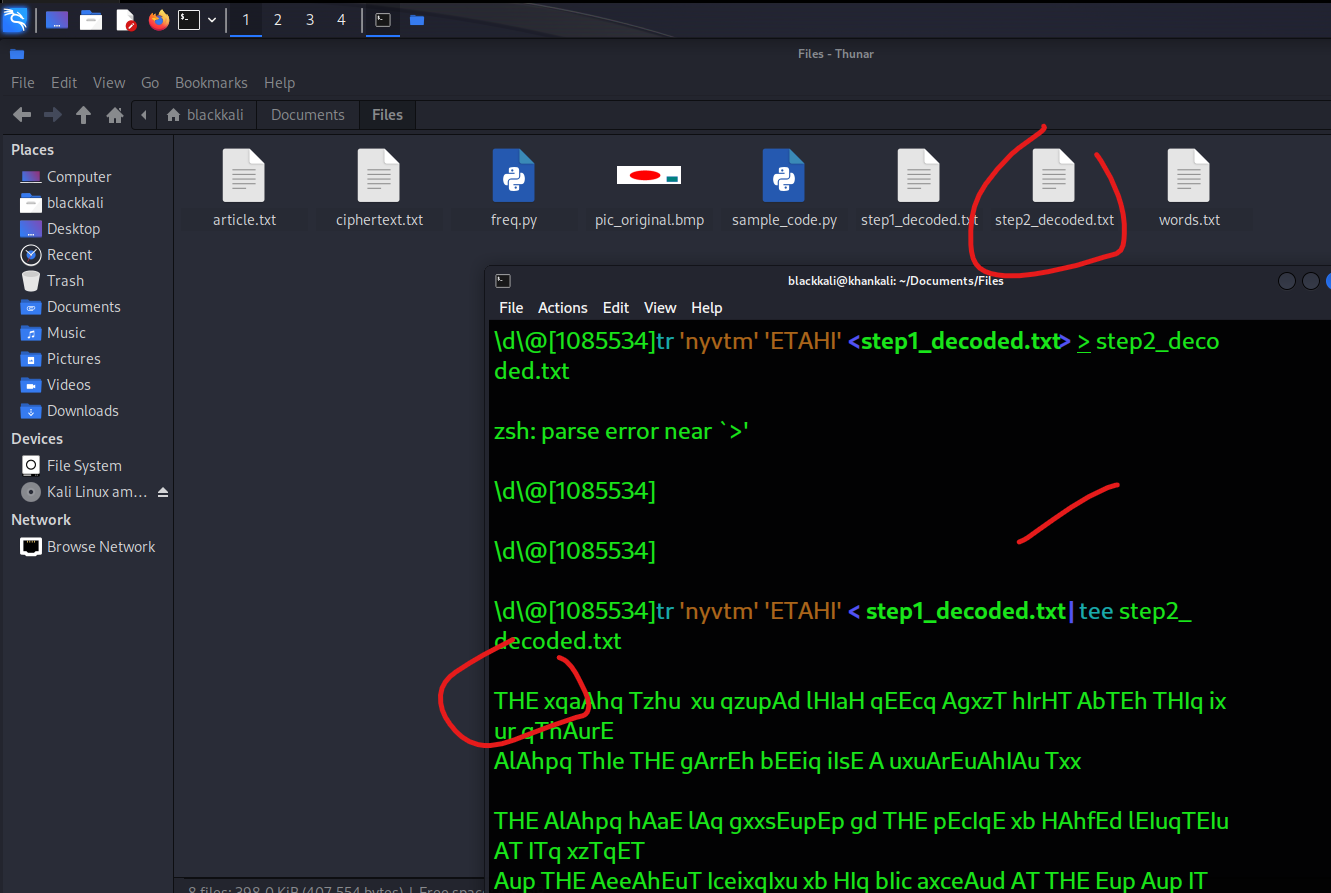
**3-gram Frequencies**:

* The most frequent 3-grams (triplets of letters) are: ytn, vup, mur, etc.
* In English, common 3-grams are: **THE, AND, ING**.

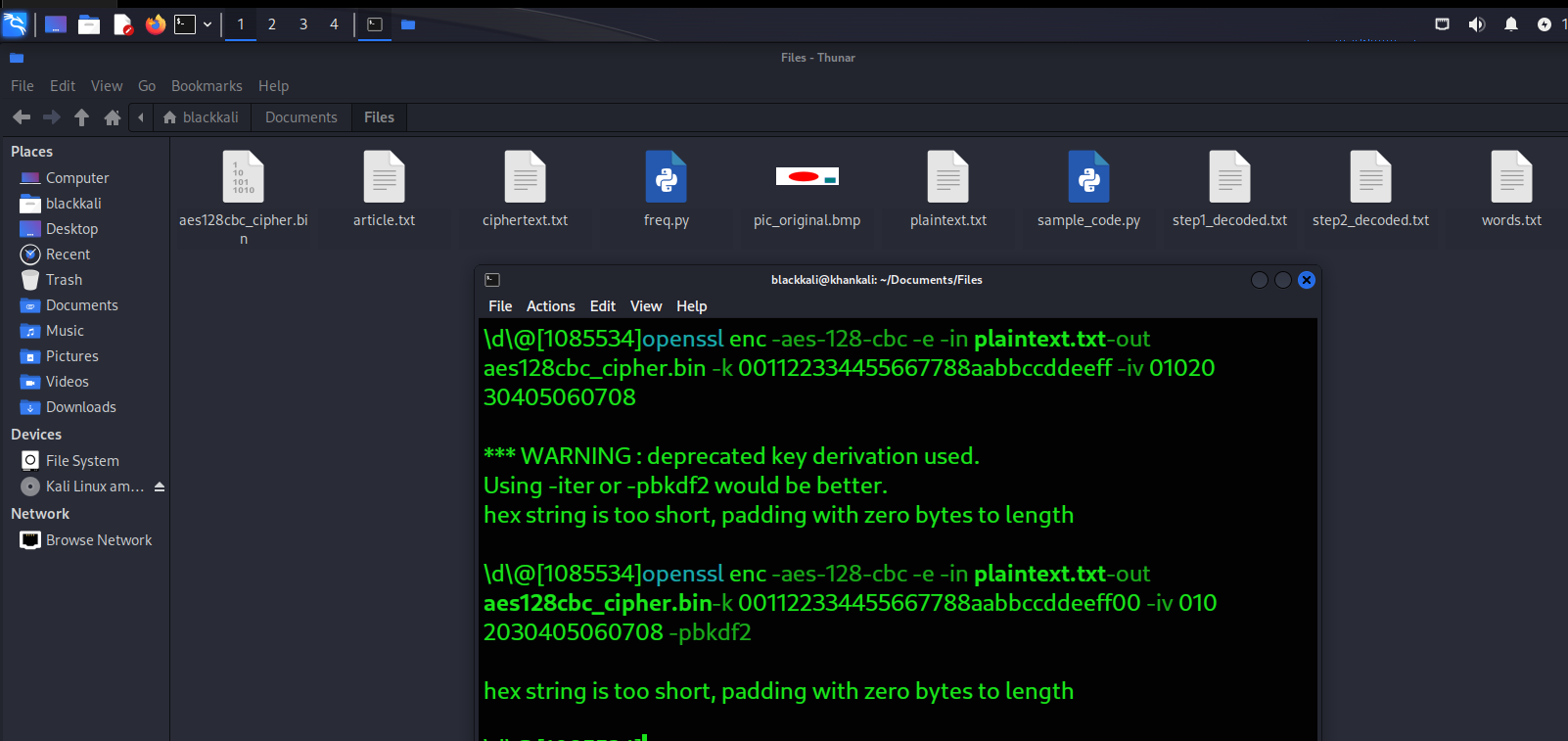
So finally

* ytn ---> **THE**
* mur ---> **ING**

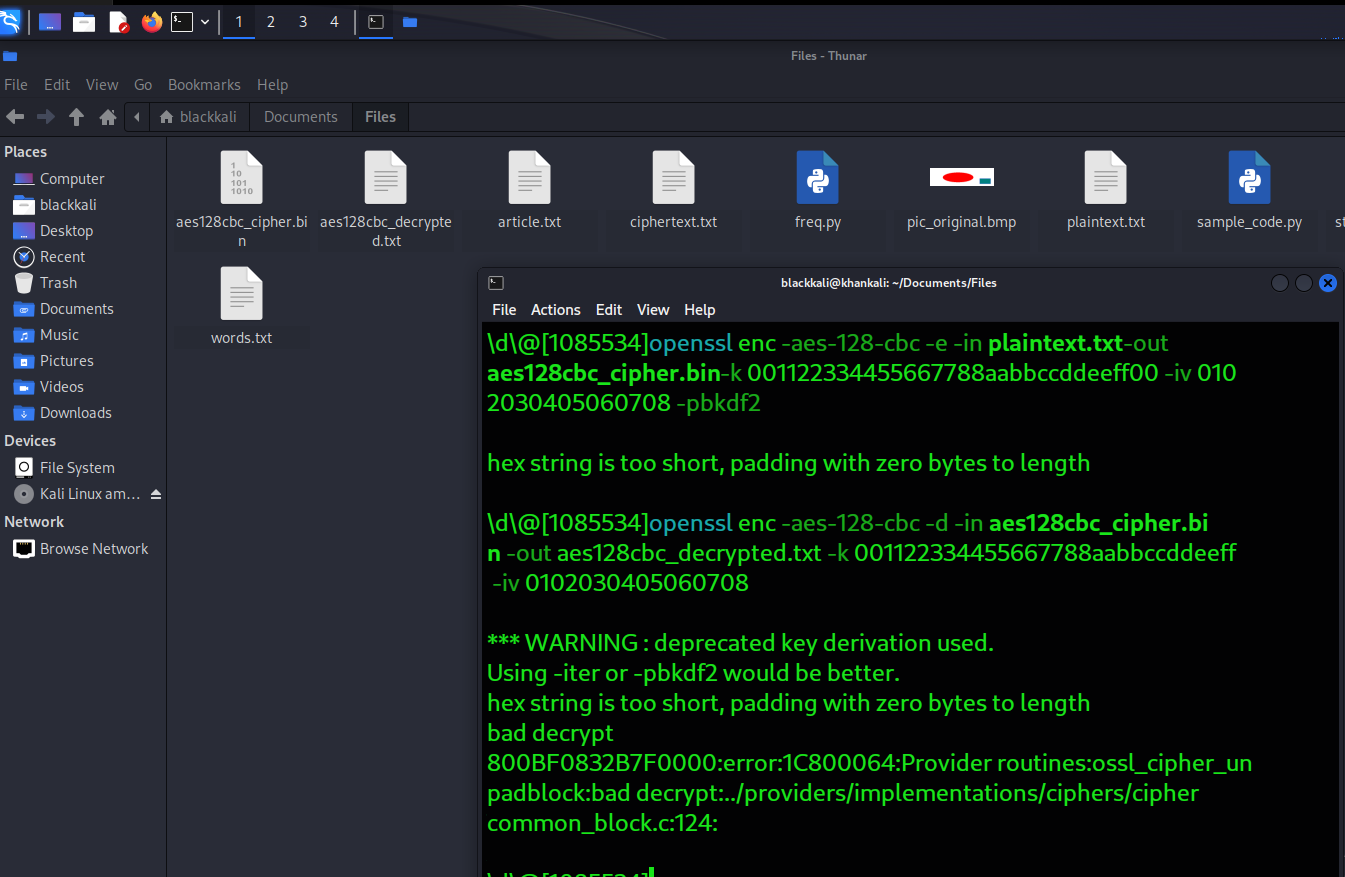




Task3:



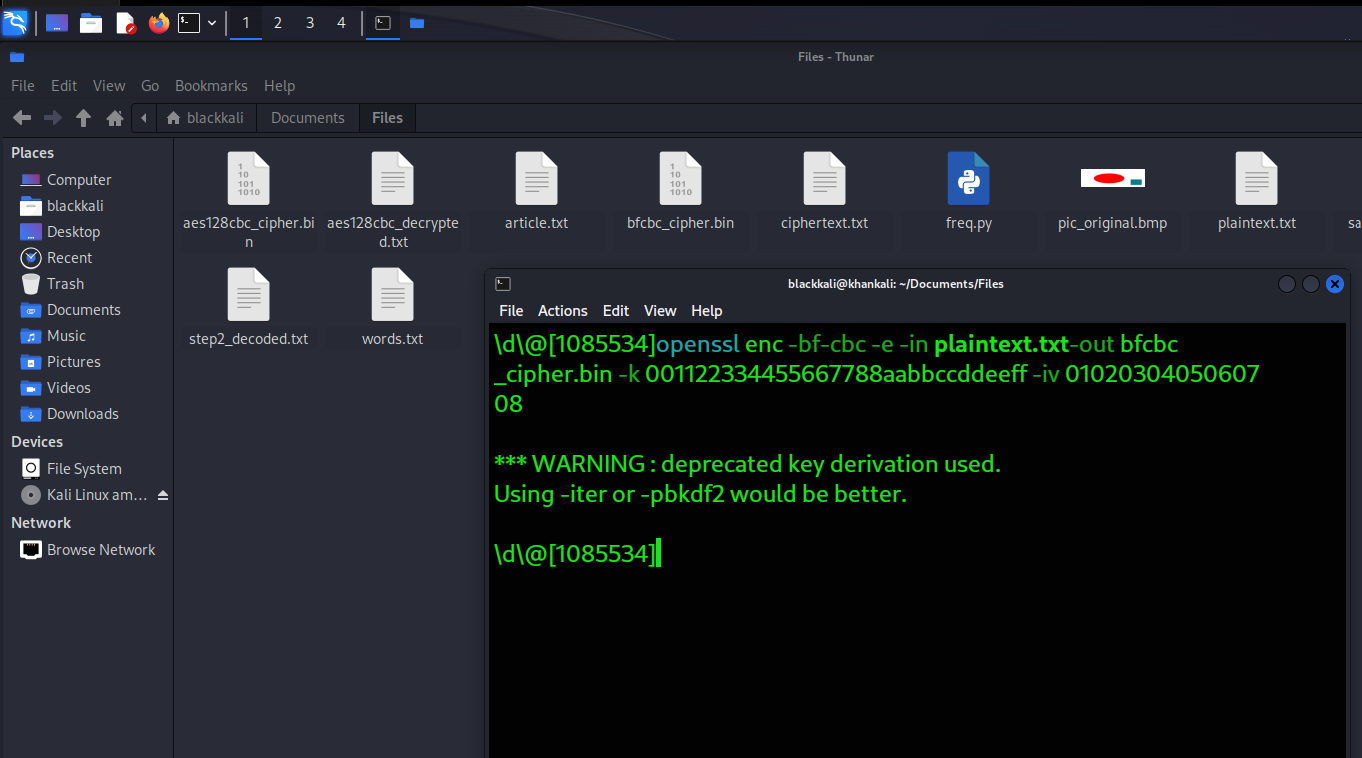
**Decryption:** Now decrypt the file back to plaintext:



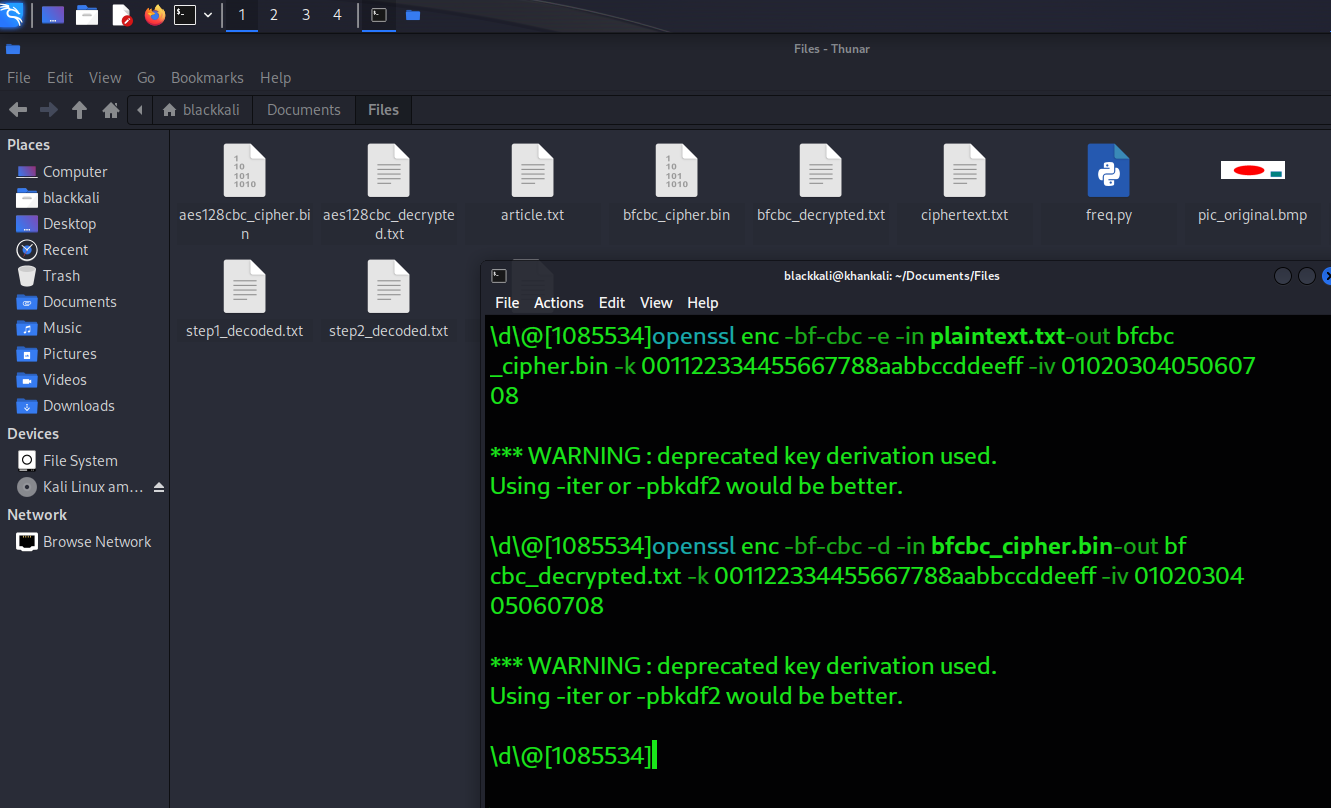
**Step 2: Encrypt with BF-CBC (Blowfish)**

Next, let's use the **Blowfish** cipher in CBC mode.

1. **Encryption:** Encrypt the file using Blowfish:



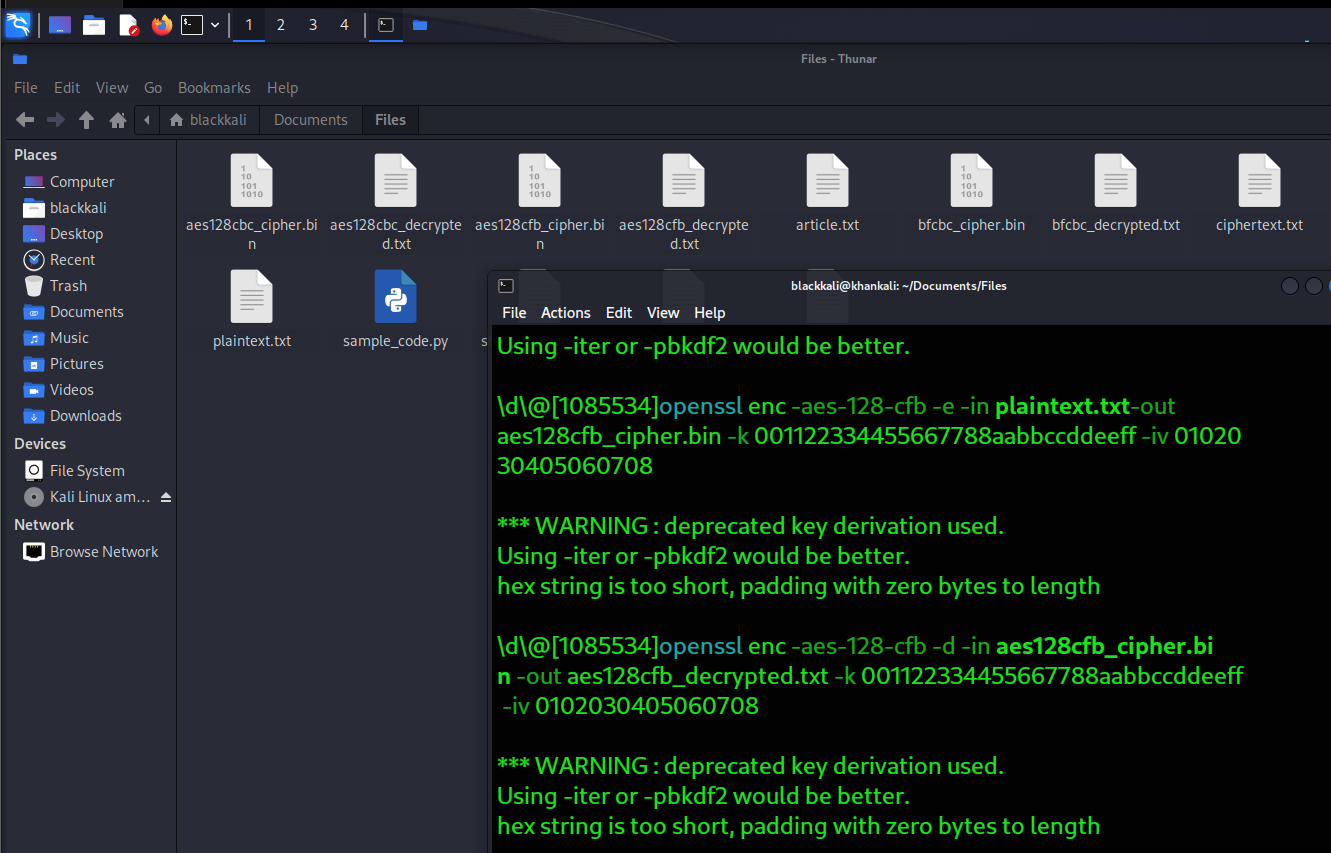
**Decryption:** Now decrypt the file back to plaintext:



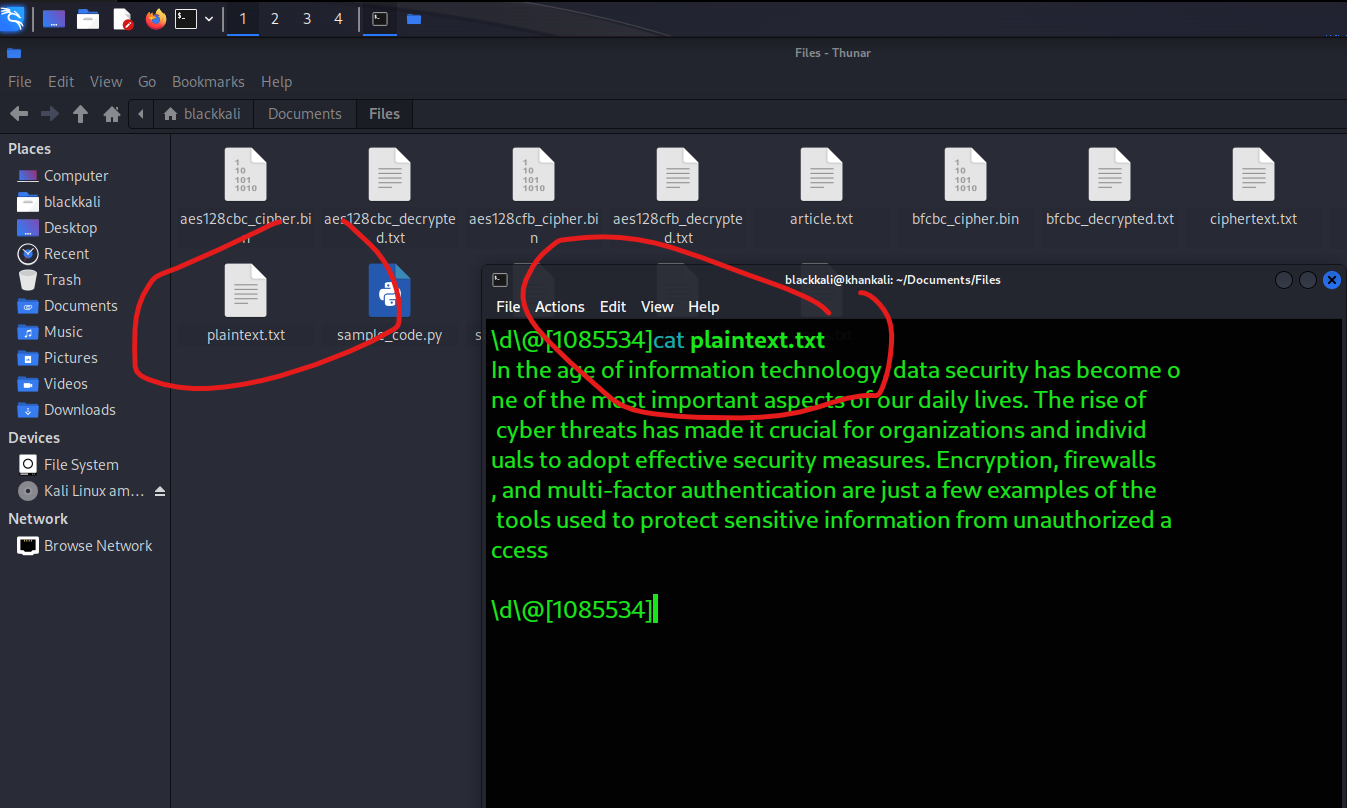
**Step 3: Encrypt with AES-128-CFB**

Finally, let's use **AES-128-CFB** for encryption, which stands for Cipher Feedback mode.

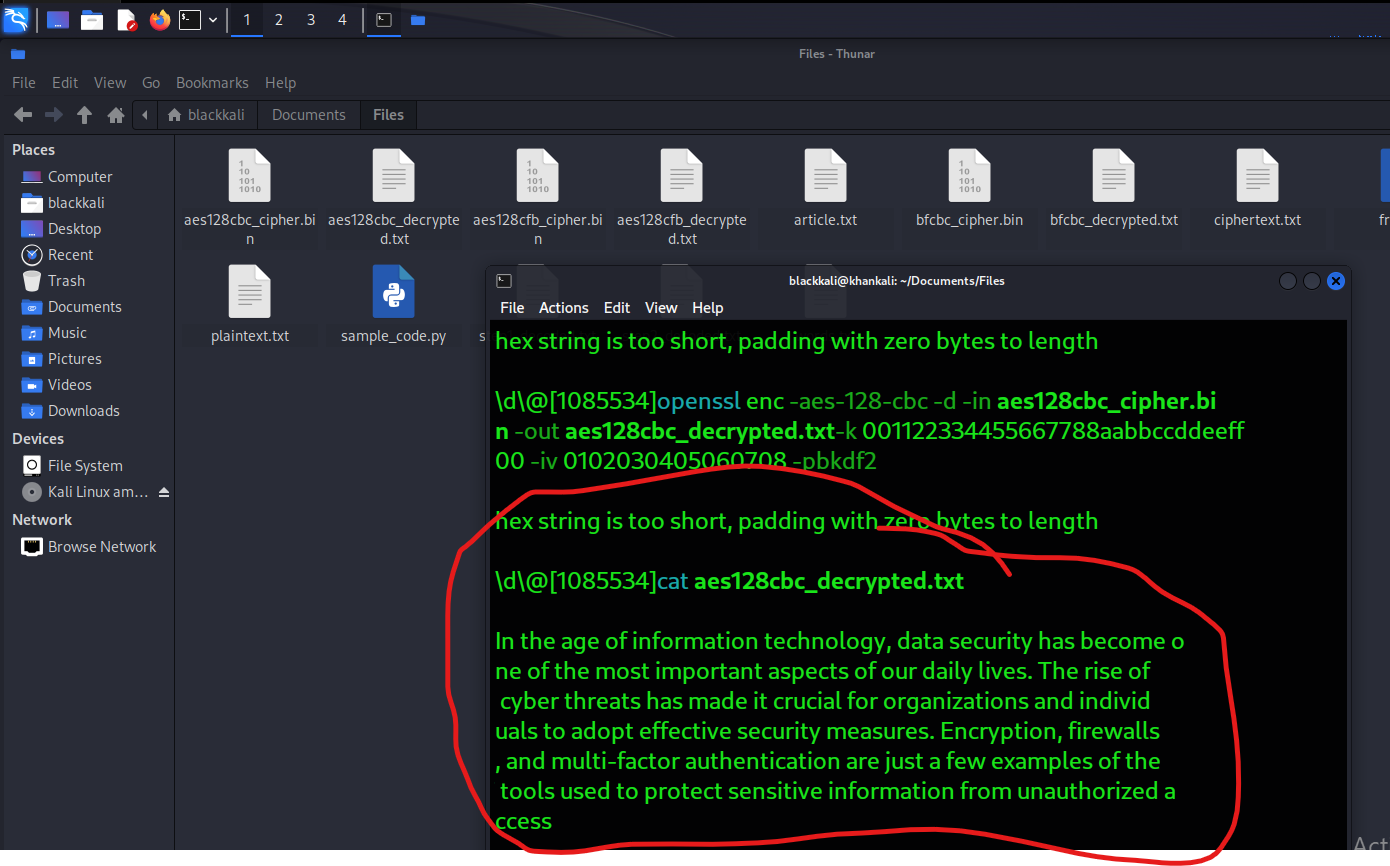
1. **Encryption:** Encrypt the file using AES-128-CFB:
2. **Decryption:** Now decrypt the file back to plaintext:



Original Plaintext:



decrypted files (aes128cbc\_decrypted.txt

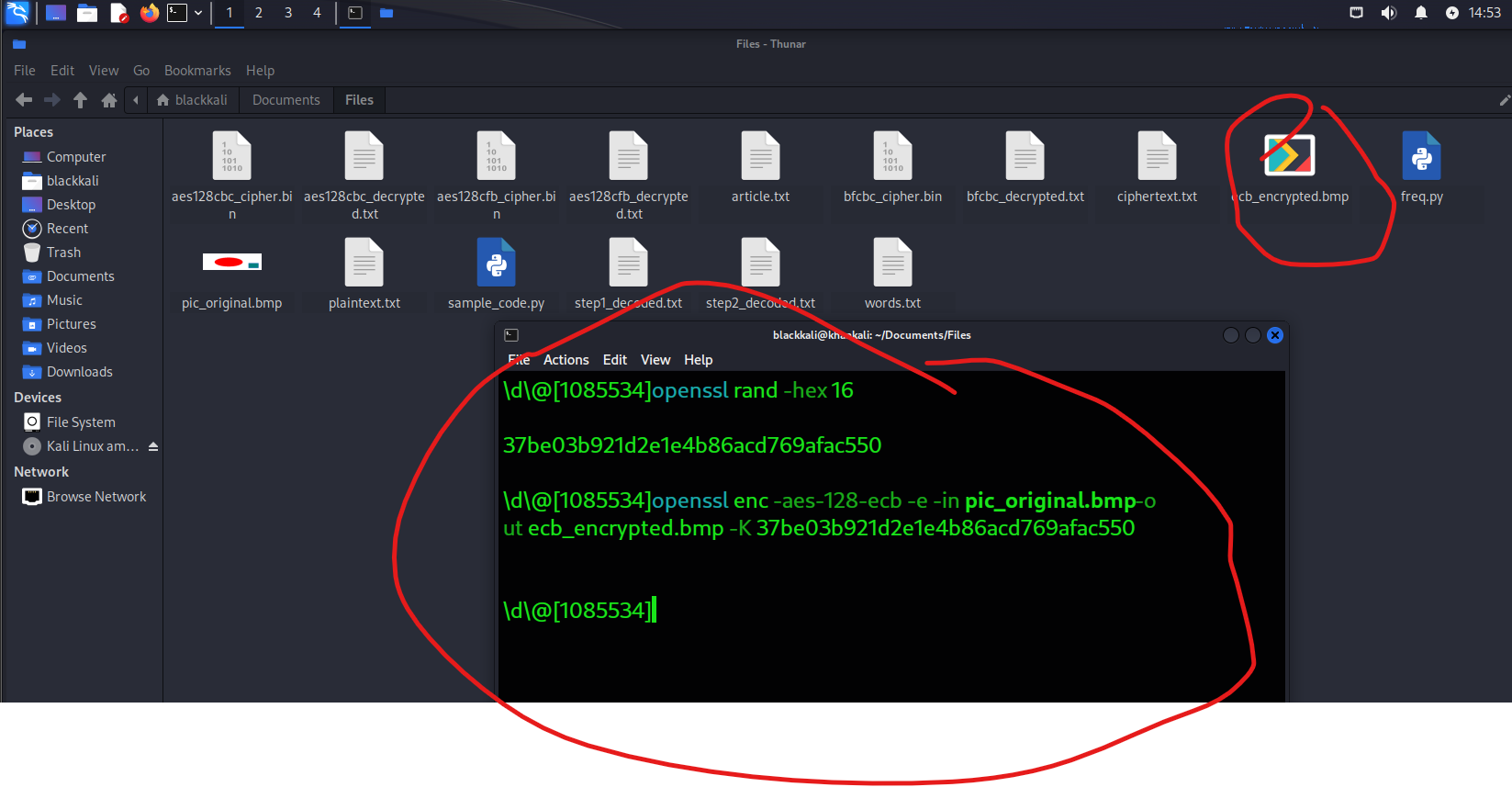


Task4:

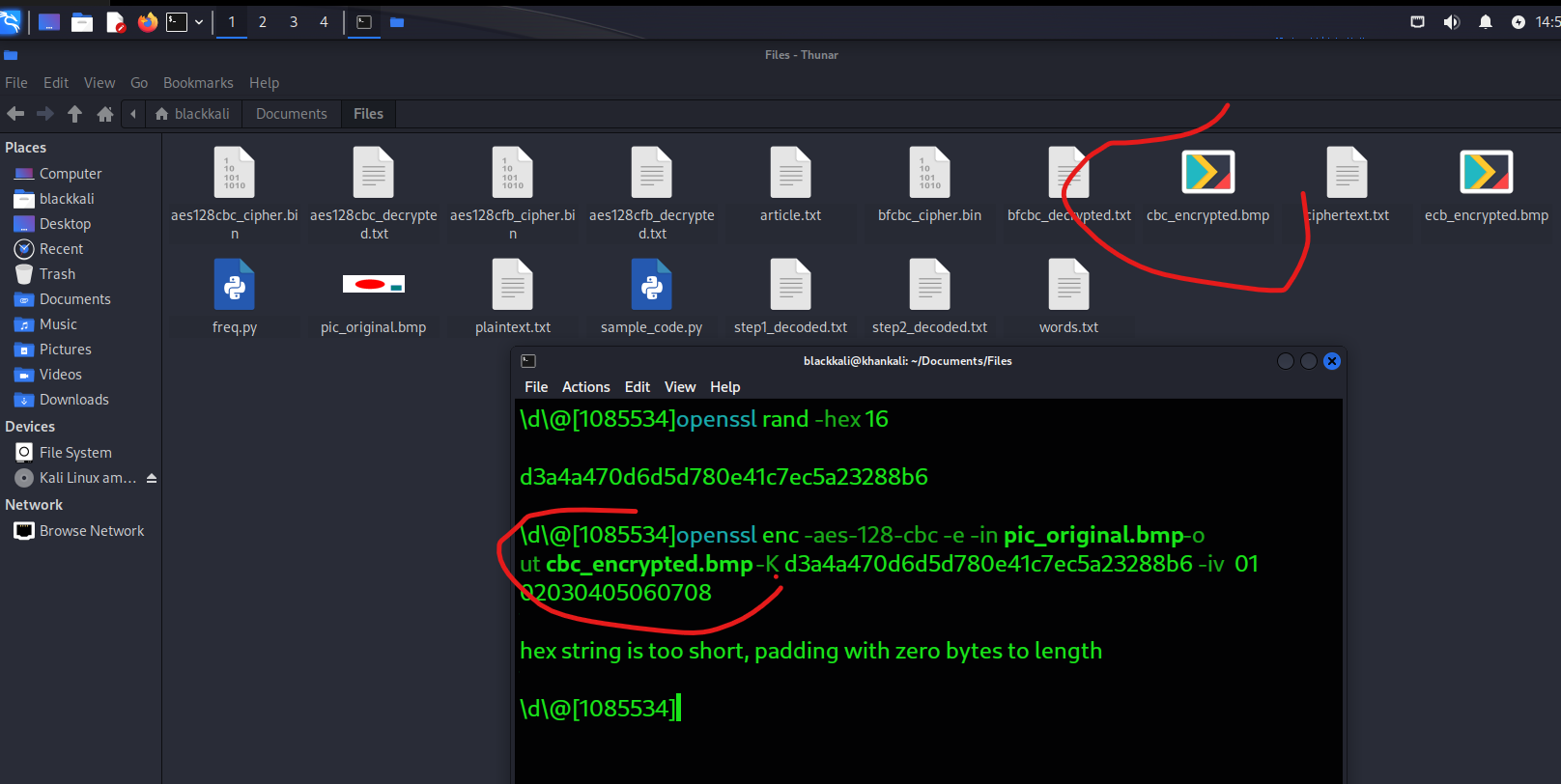
Step 1: Encrypt the Picture

**Encrypt Using ECB Mode**:

Run the following command to encrypt pic\_original.bmp using AES in ECB mode

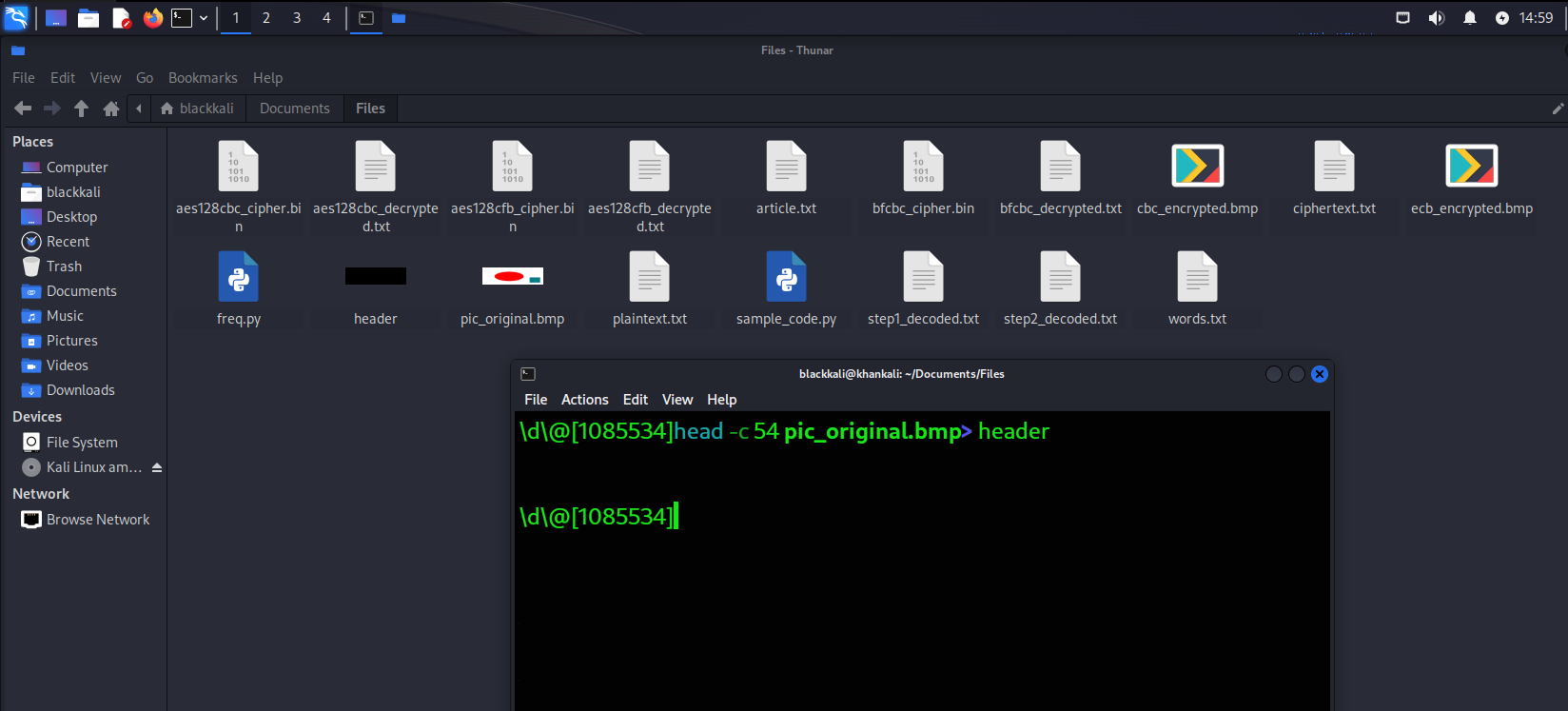


**Encrypt Using CBC Mode**: Next, encrypt the same picture using AES in CBC mode.



Step 2: Modify the Headers

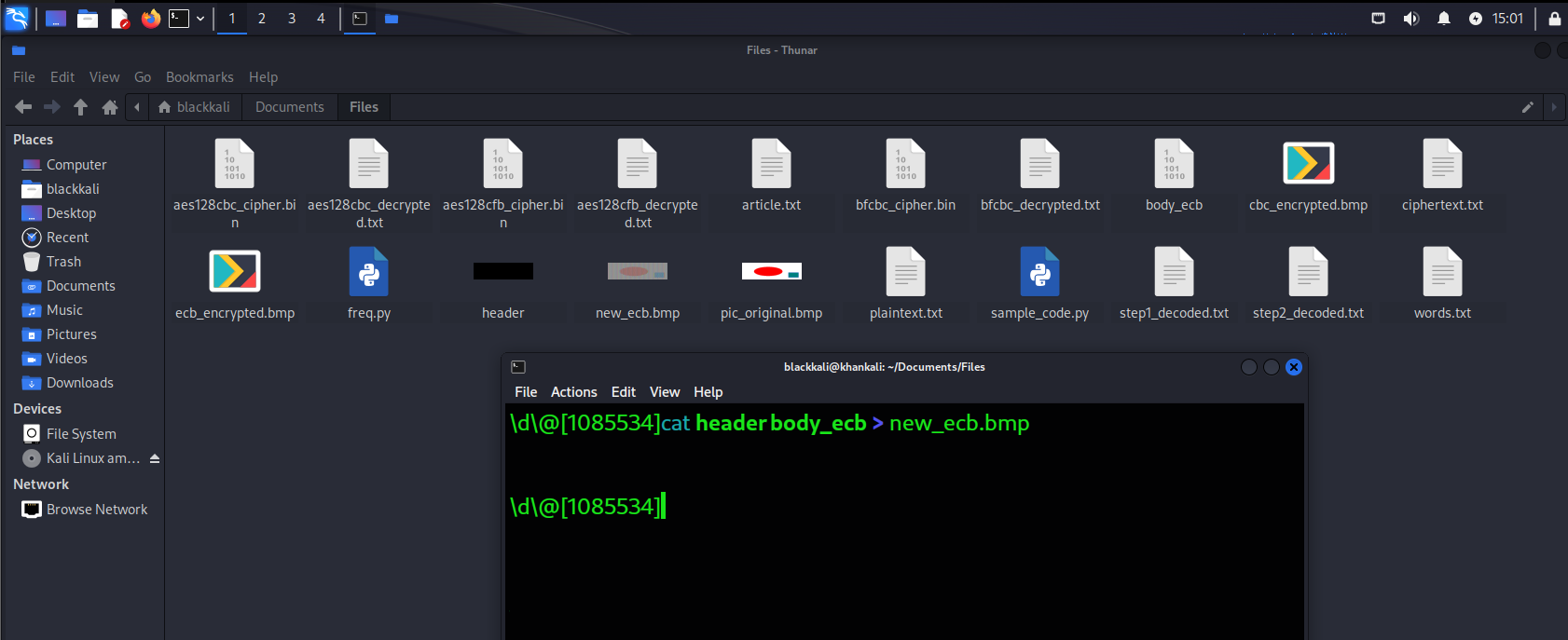
**Extract the Header from the Original BMP**:



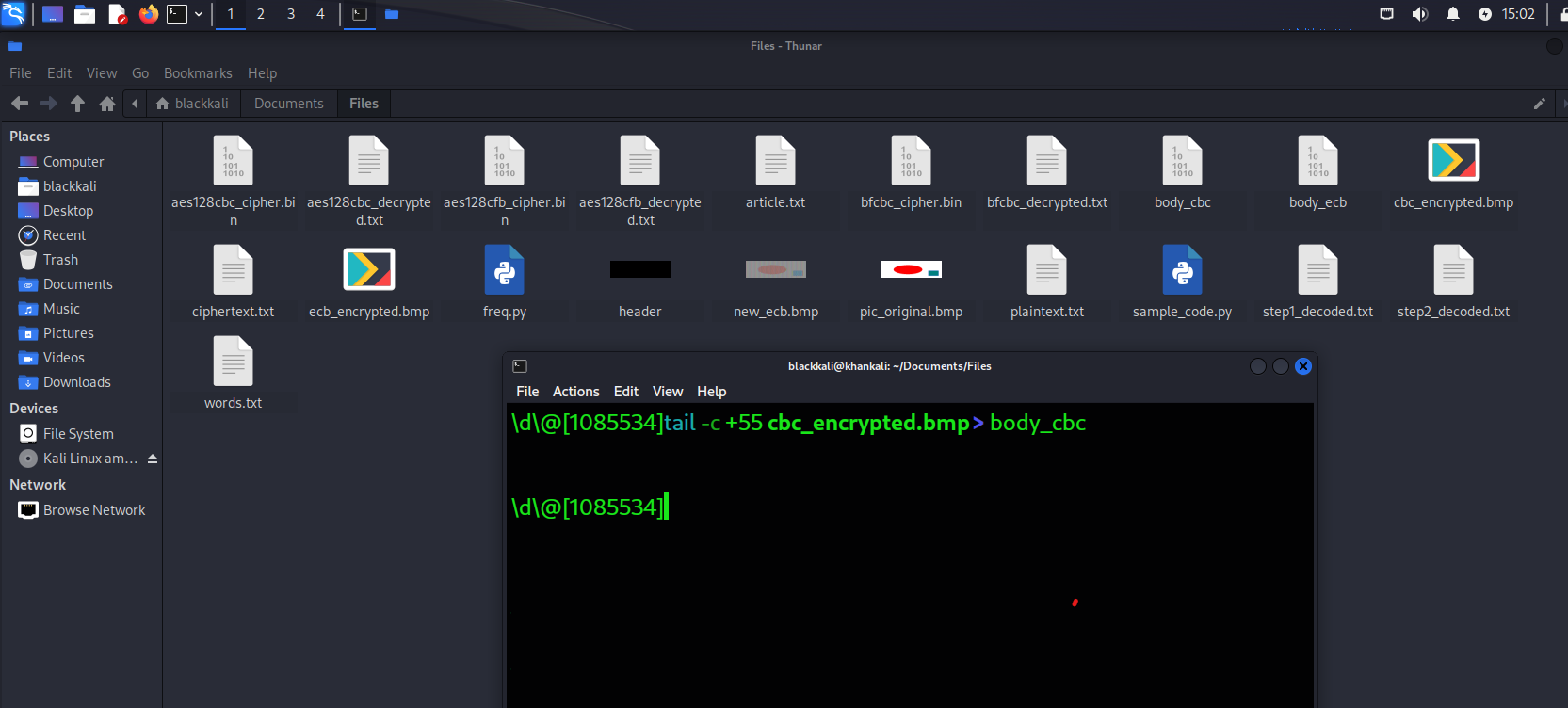
**Extract the Body from the Encrypted BMP (ECB)**:



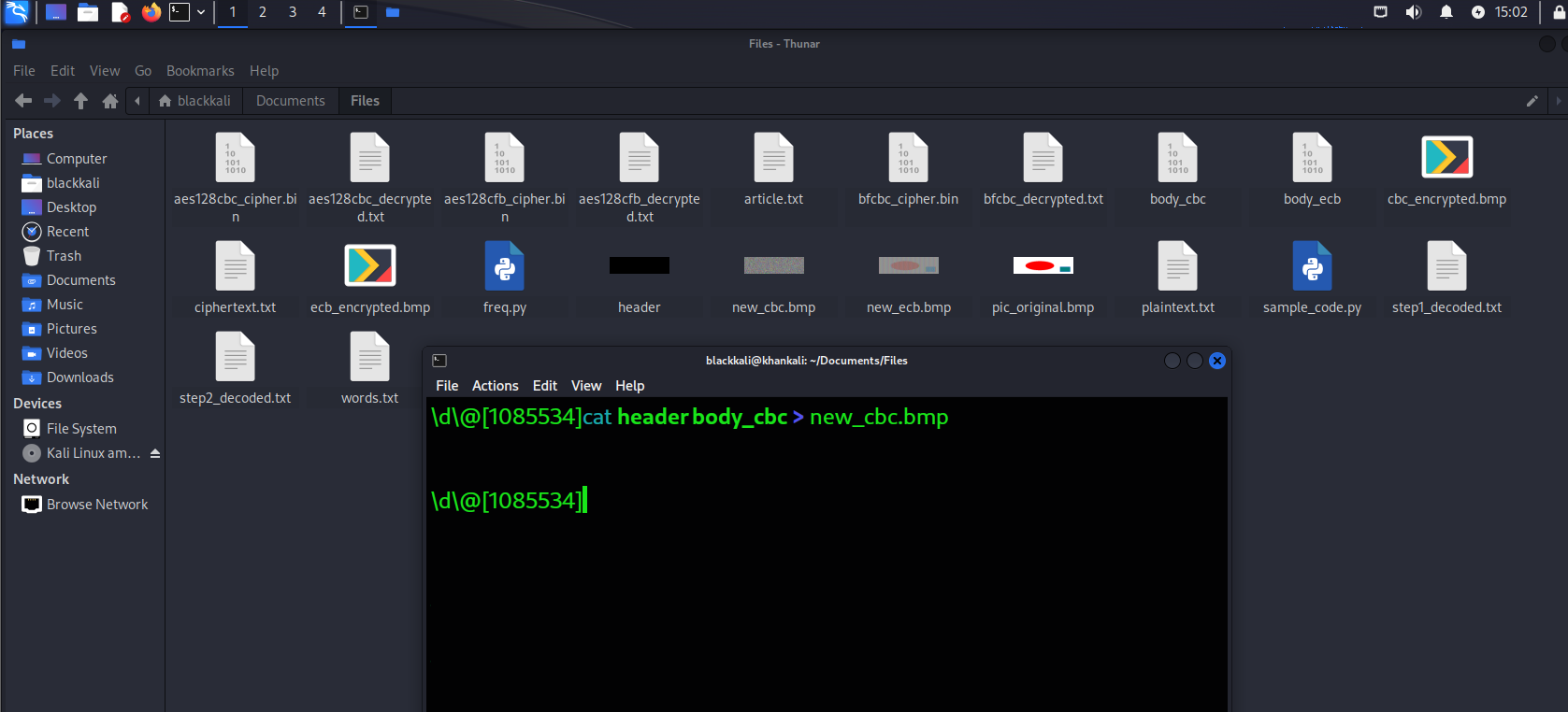
**Combine the Header and Encrypted Body (ECB)**:



Extract the Body from the Encrypted BMP (CBC)

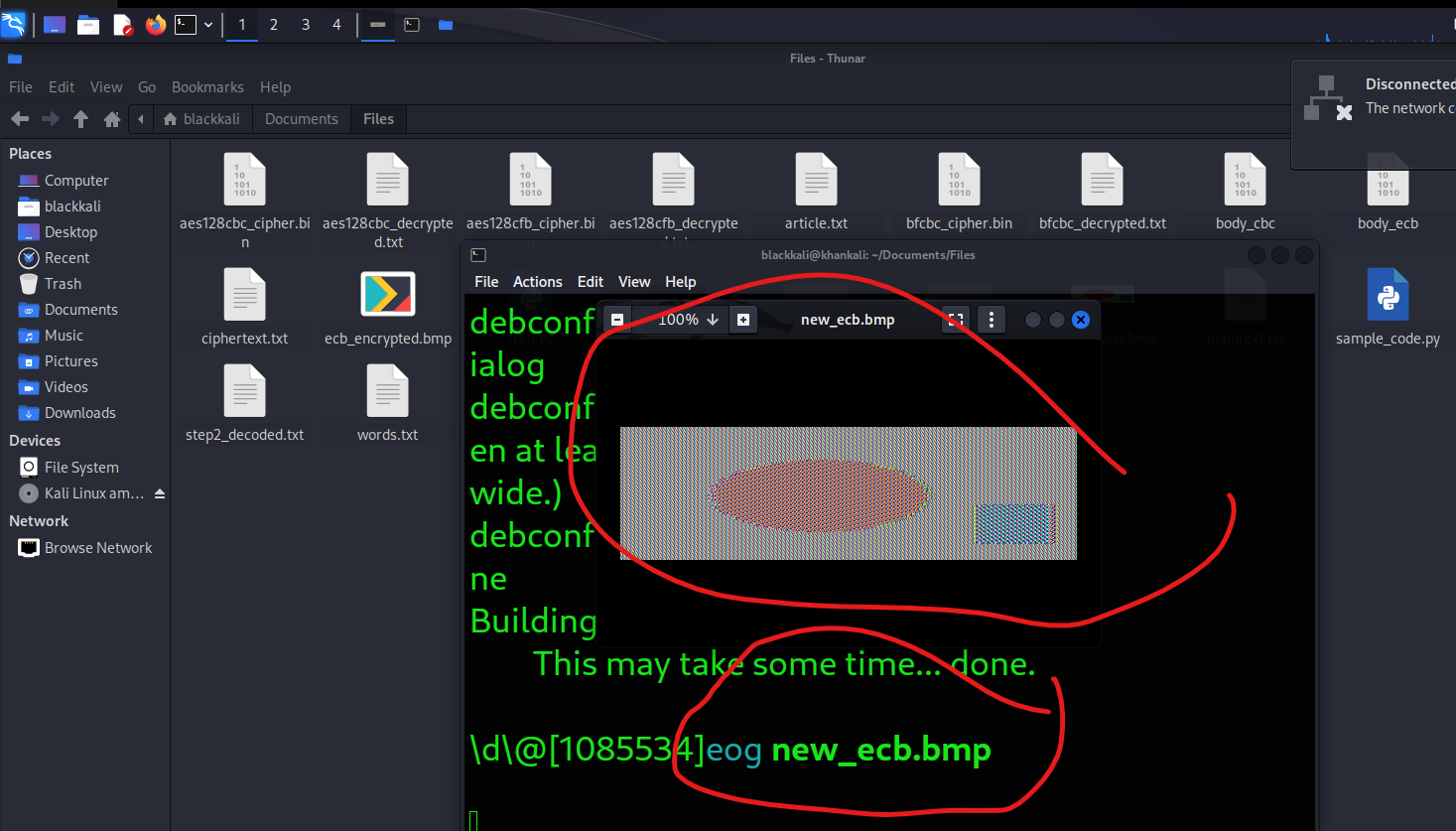


**Combine the Header and Encrypted Body (CBC)**:

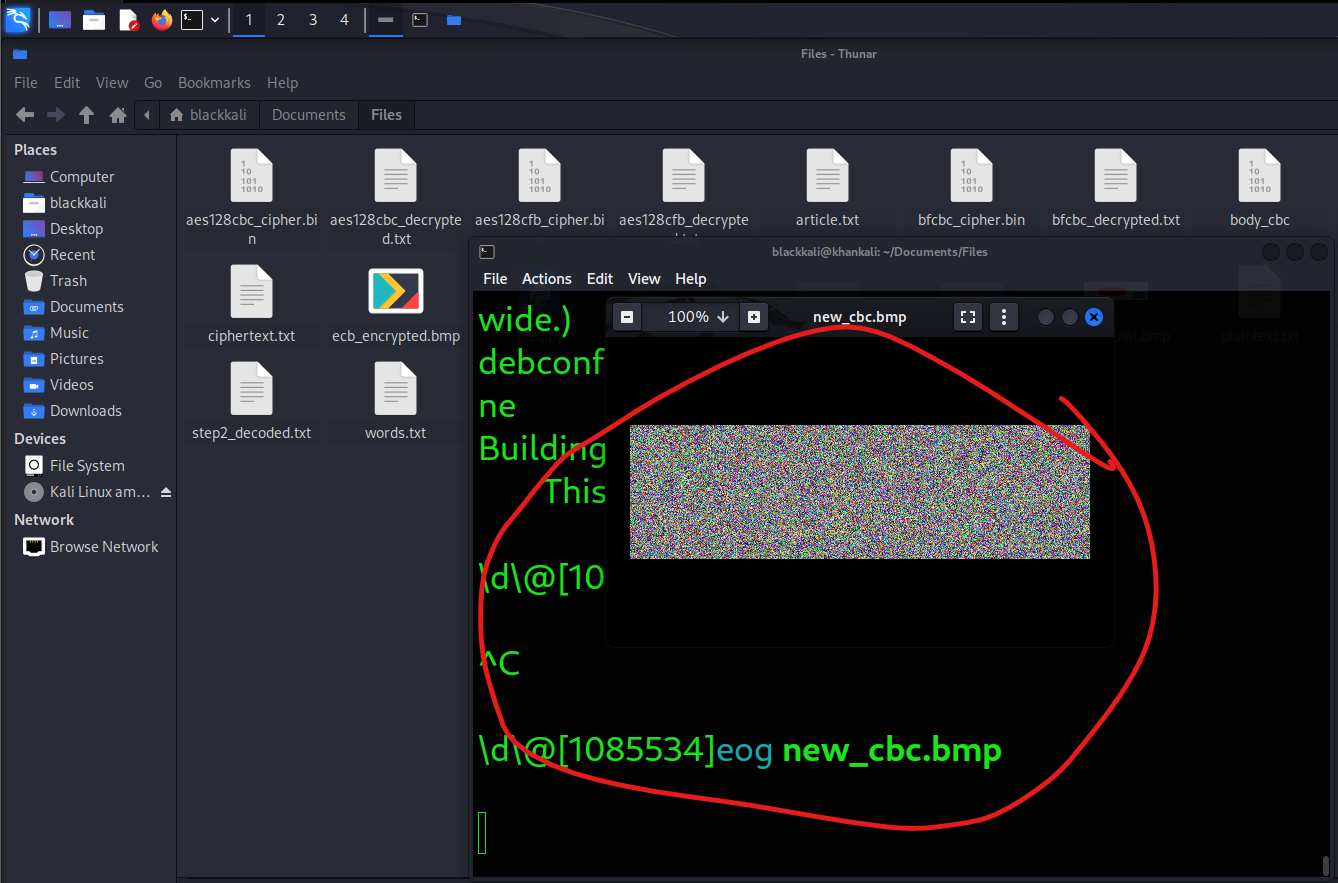


Step 3: View the Encrypted Pictures

**Display the ECB Encrypted Image**:



**Display the CBC Encrypted Image**:



**Step 4: Analyze the Results**

After viewing the encrypted images, consider the following observations:

1. **ECB Mode**:

The ECB mode encrypts identical plaintext blocks into identical ciphertext blocks. This can lead to recognizable patterns in the encrypted image, such as sections that look the same. You may notice parts of the image structure being preserved, which can reveal some information about the original image.

1. **CBC Mode**:

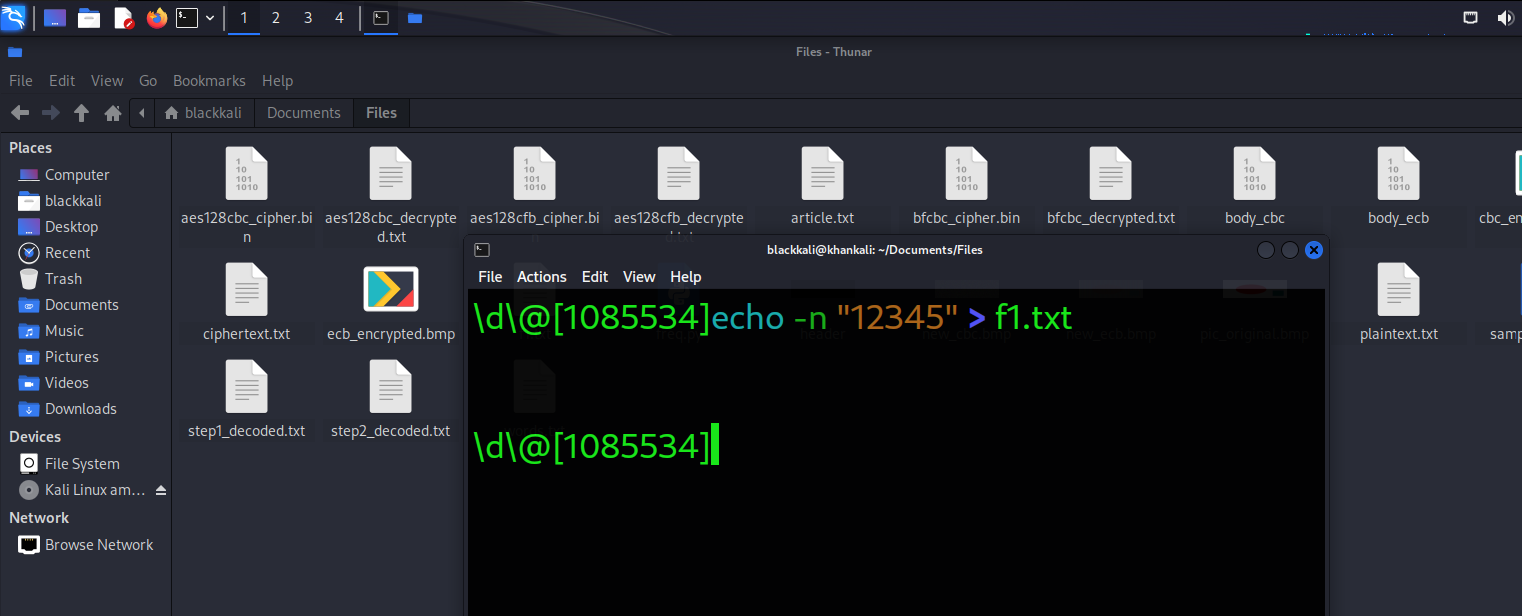
The CBC mode introduces randomness because each block of plaintext is XORed with the previous ciphertext block before encryption. As a result, even identical blocks of plaintext will produce different ciphertexts. Therefore, the encrypted image will look more random compared to the ECB mode, making it less susceptible to analysis.

Task5:

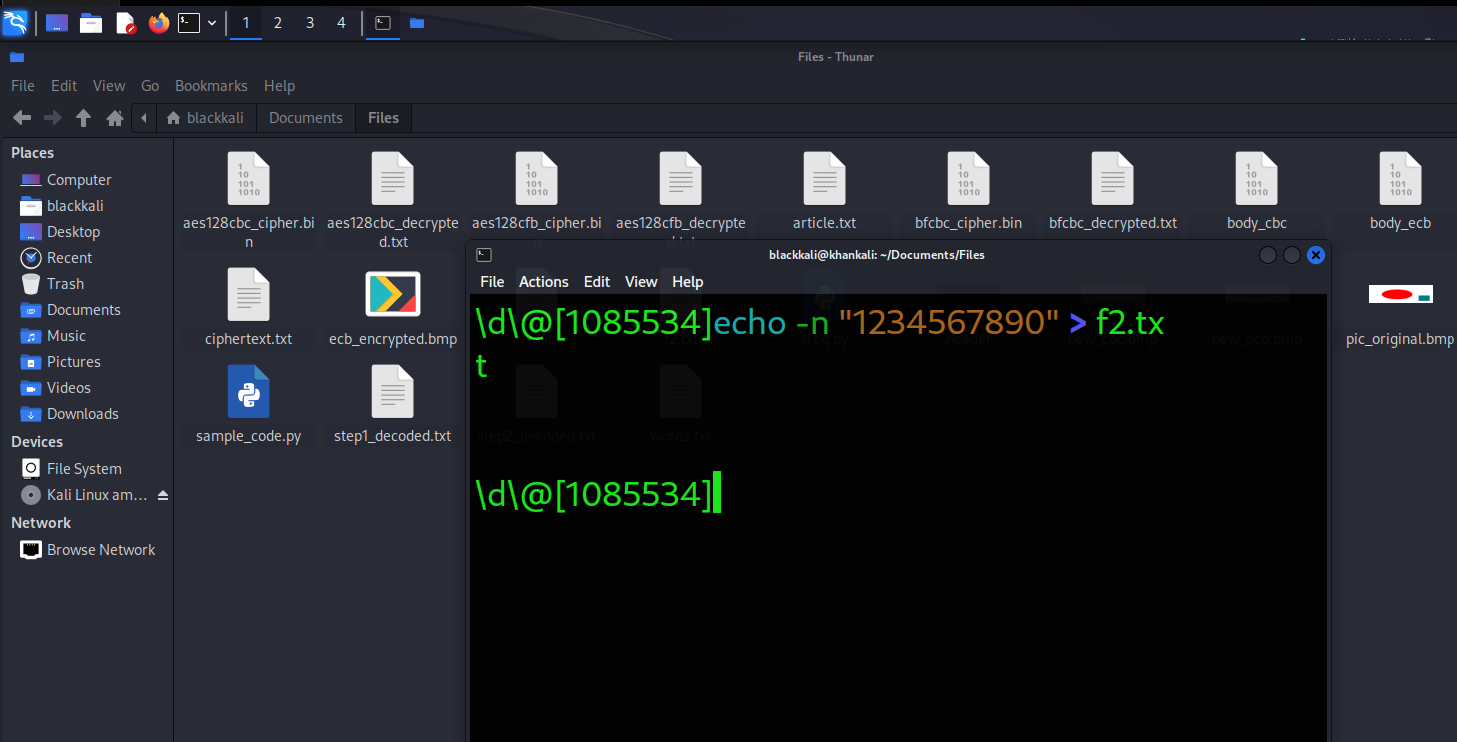
**Step 1: Create Files**

First, create three text files of different sizes (5 bytes, 10 bytes, and 16 bytes):

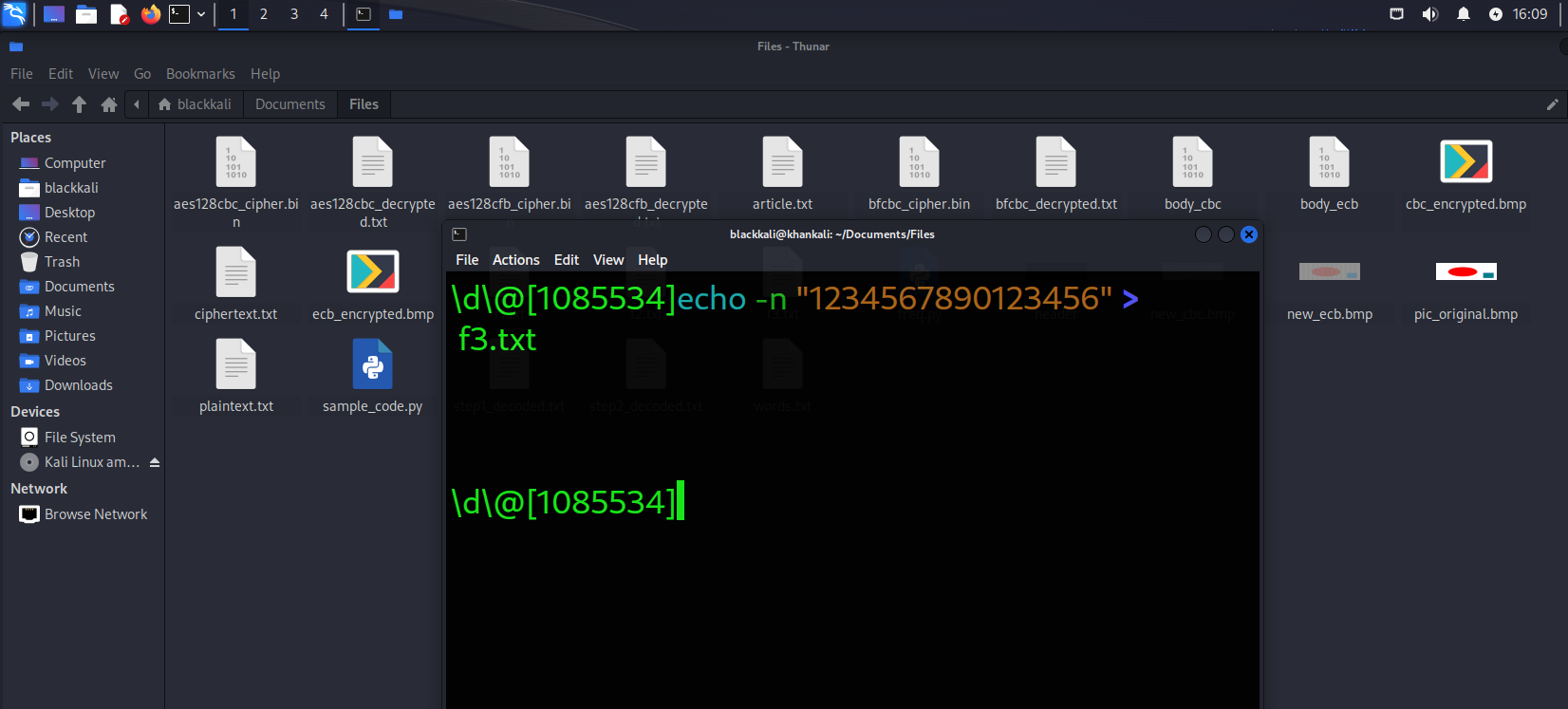
1. **Create the 5-byte file**:



**Create the 10-byte file**:



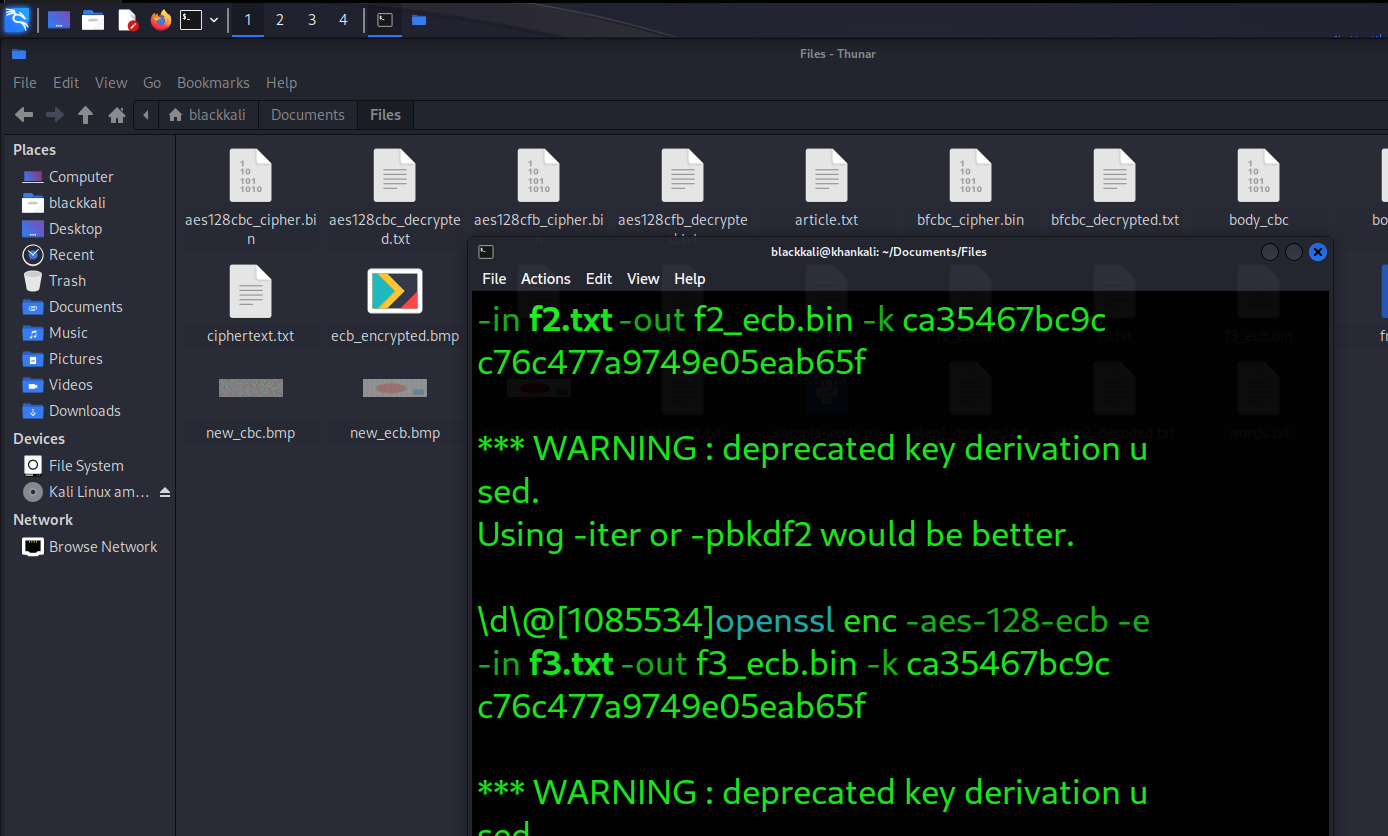
**Create the 16-byte file**:



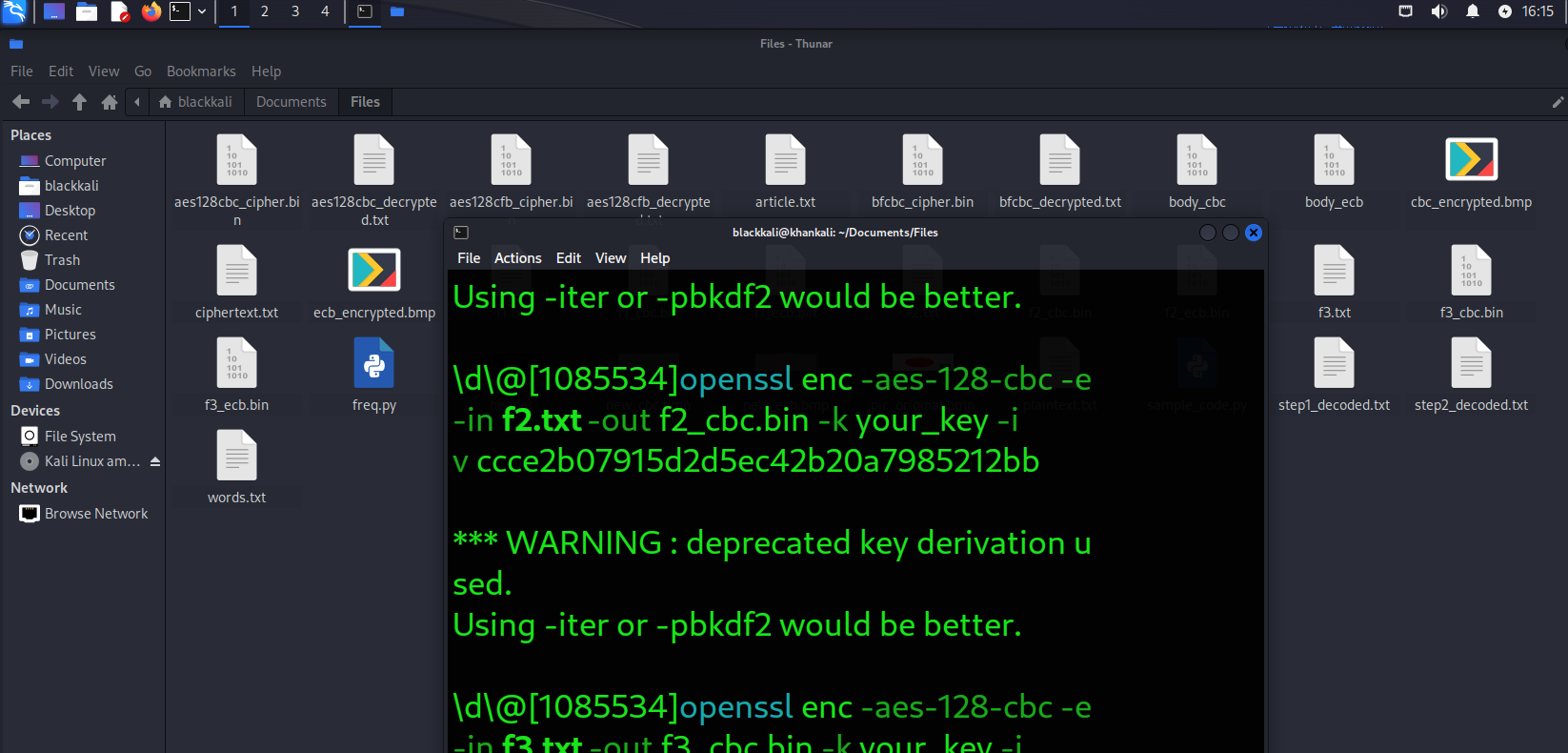
**Step 2: Encrypt the Files**

Now, encrypt each file using AES-128 in ECB, CBC, CFB, and OFB modes. Use a key

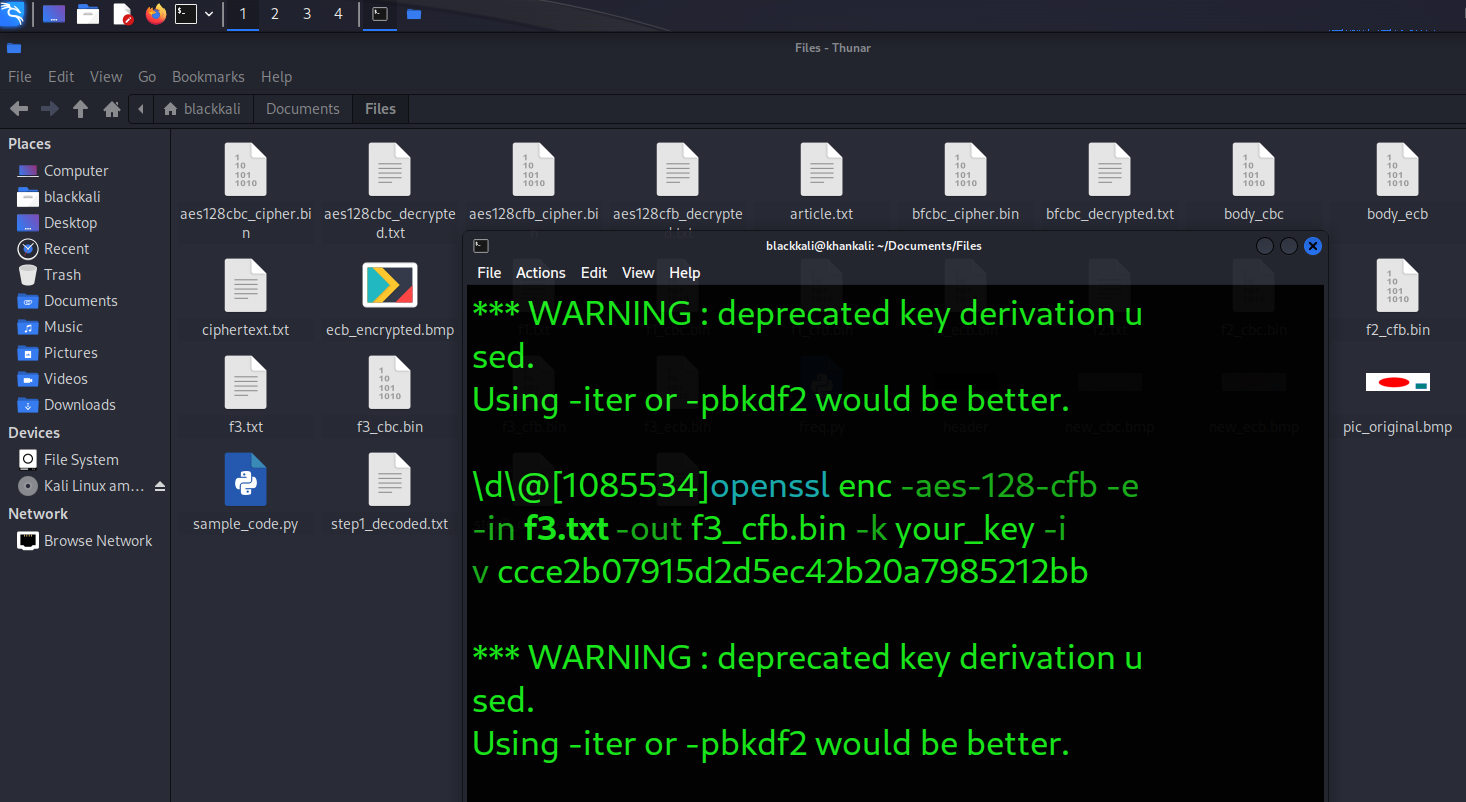
1. **Encrypt with ECB mode**:



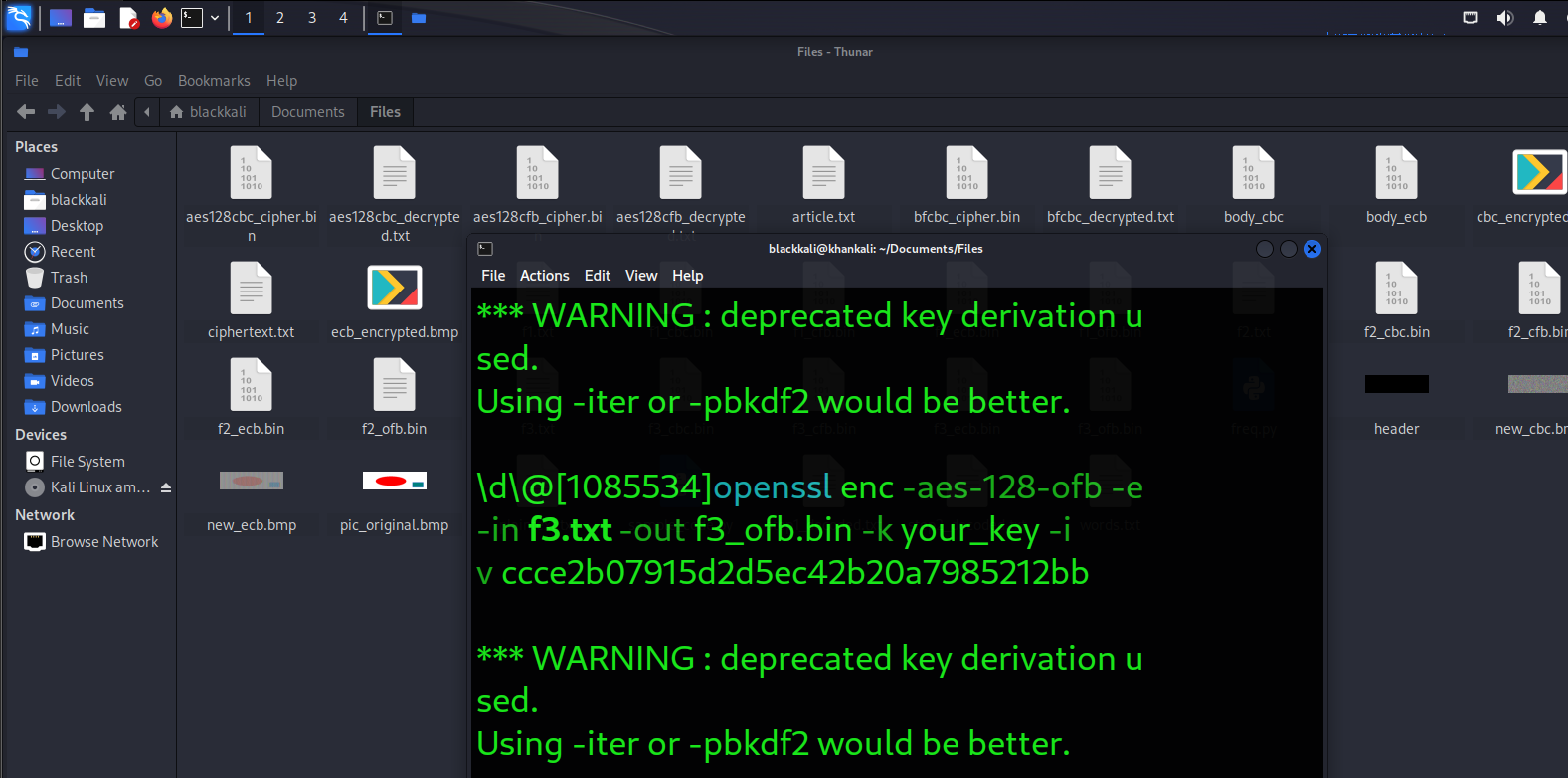
Encrypt with CBC mode



**Encrypt with CFB mode**:

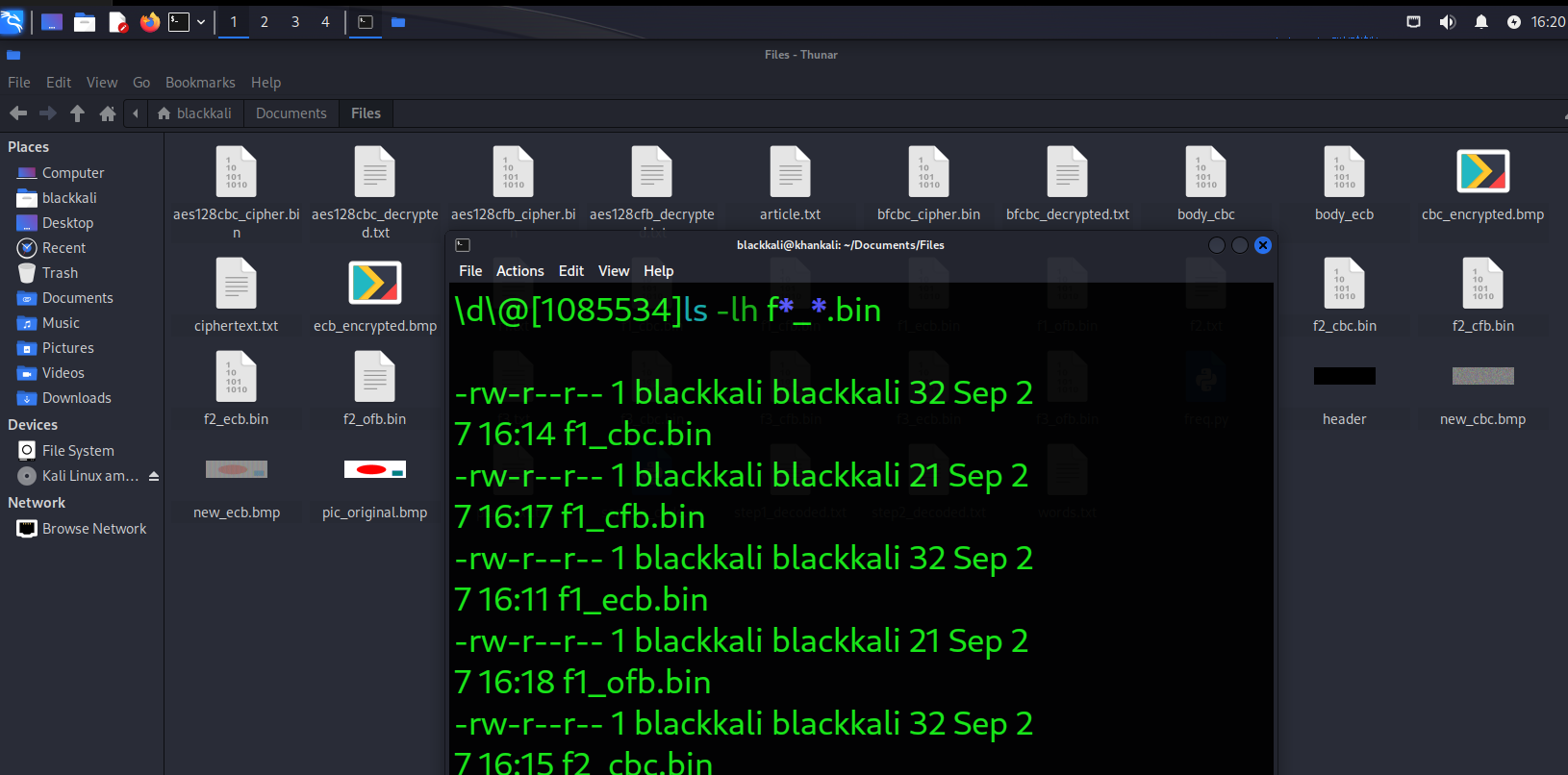


**Encrypt with OFB mode**:



**Step 3: Check the Size of Encrypted Files**

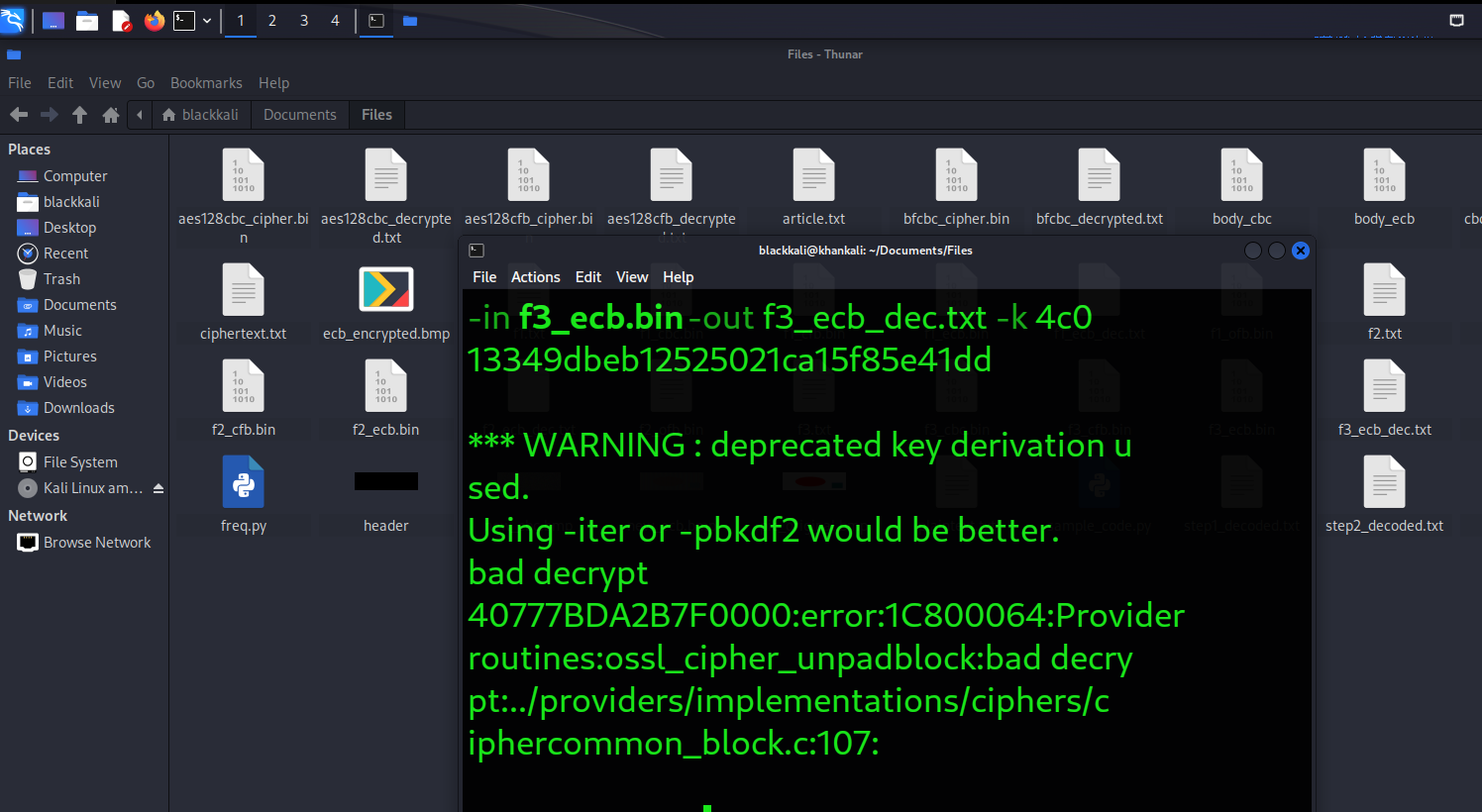
Check the size of each encrypted file to observe how padding affects the output file size:



**Step 4: Decrypt the Files Without Padding**

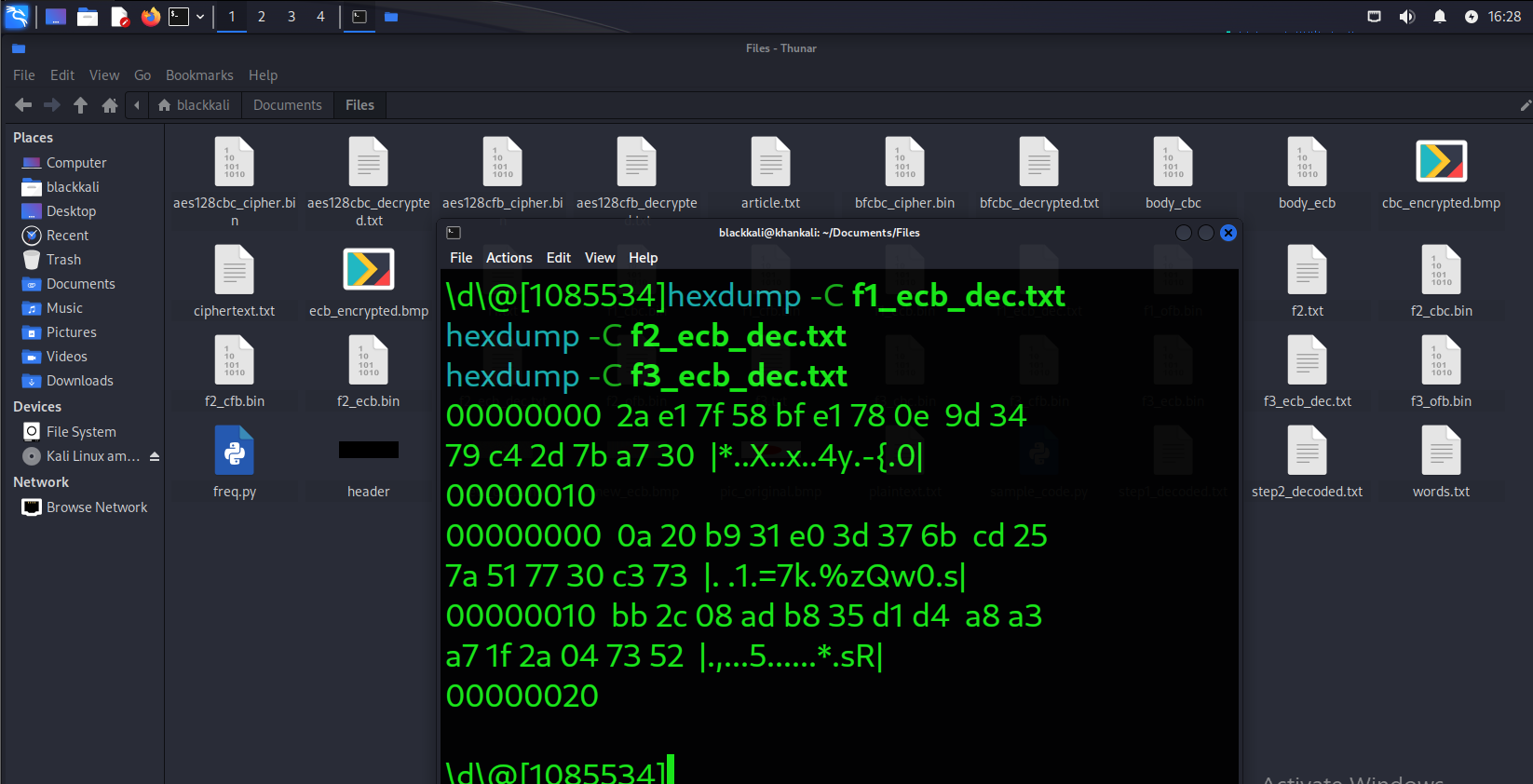
Now decrypt the files without removing the padding to see what has been added.

1. **Decrypt ECB mode files**:



**Step 5: Use Hex Dump to Display Padding**

Use hexdump to examine the decrypted files and observe the padding:



**Step 6: Analyze Padding**

1. **Report Padding for Each Mode**:
   * **ECB**: Yes, padding is required as the input size is not a multiple of the block size.
   * **CBC**: Yes, padding is required for the same reason.
   * **CFB**: No, CFB mode does not require padding as it can handle any length of input data.
   * **OFB**: No, OFB mode does not require padding for the same reason.