New York Institute of Technology



School of Engineering and Computing Sciences

**INCS 741 - Cryptography**

***Exploring Symmetric Key Encryption Modes***

Group 1

Marcelo Magnus Antunes de Oliveira (1282591)

Meet Soni (1279545)

Harshita Grover (1276595)

Professor: Tokunbo Makanju

Date Submitted:

April 1, 2021

Lab Report

Exploring Symmetric Key Encryption Modes

# **Task 1. Warm-up**

1. **What is the difference between size of plaintext and the file after encryption? mention the size of both files in bytes [2 Marks]**

**Answer:**

In block ciphers, the size of plain text and file after encryption are exactly the same.

However, practically, the size of cipher text file must be slightly greater because of the block cipher. AES cipher uses fixed block size and padding, but it is more accurate to say that cipher text is same size rounded up to the nearest block size of the plain text.

Here, during implementation,

Plain text – Group1.txt (4K size)

Cipher text – Group1enc.txt (4K size)

1. **Explain why you see this difference in the size of files and why the ciphertext file has that particular size? [2 Marks]**

**Answer:**

In AES encryption, the block size is done within 16-octet whereas padding is required to maintain 16-octet, which eventually brings the size of cipher text equal or slightly larger compared to plaintext.

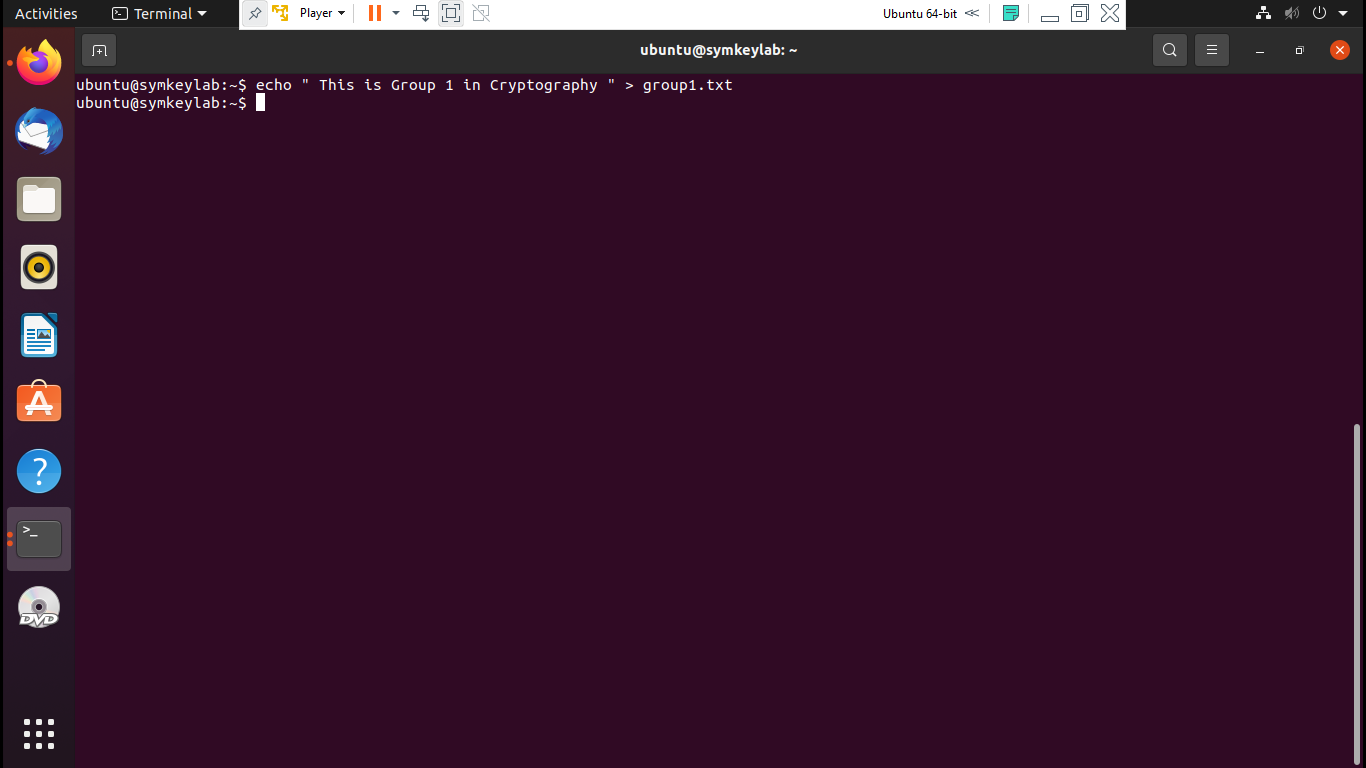
1. **Why is it better to use hexdump instead of cat, more, less or leafpad in this exercise? [1Marks]**

**Answer:**

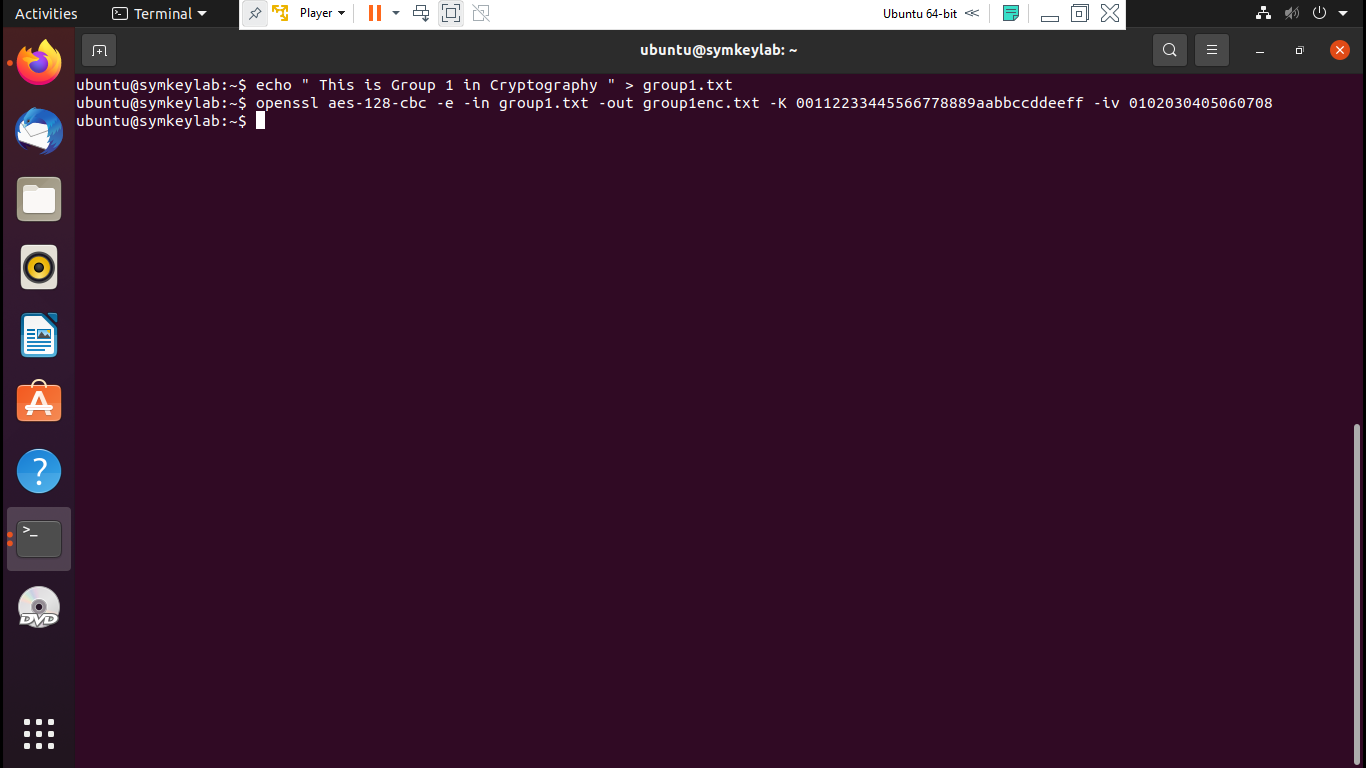
It is better to use the hexdump command in this exercise rather than commands like cat, more, less because it displays the selected files in the human readable format. As it is seen that the cat command displays gibberish text which is not human readable, however, the hexdump output is readable.

**Screenshots:**

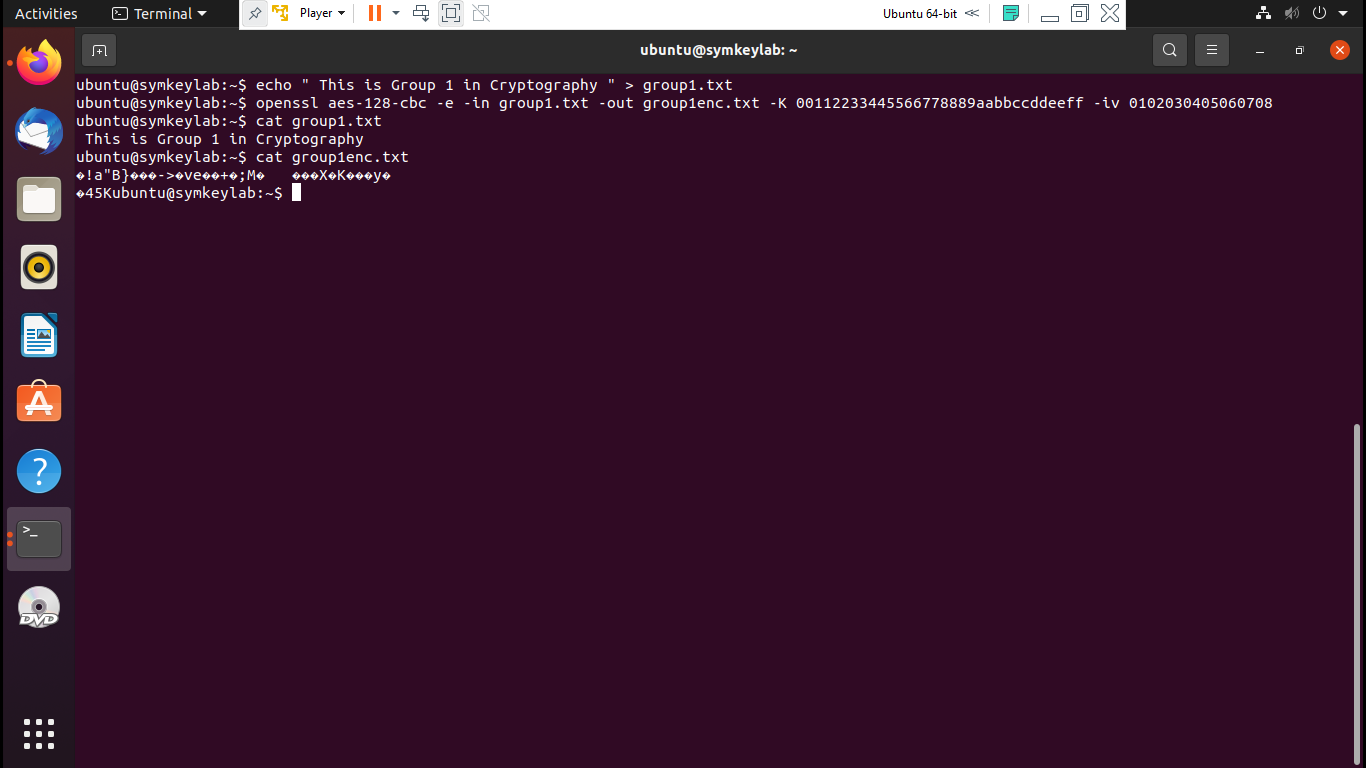
Creating the plain text file named, group1.txt



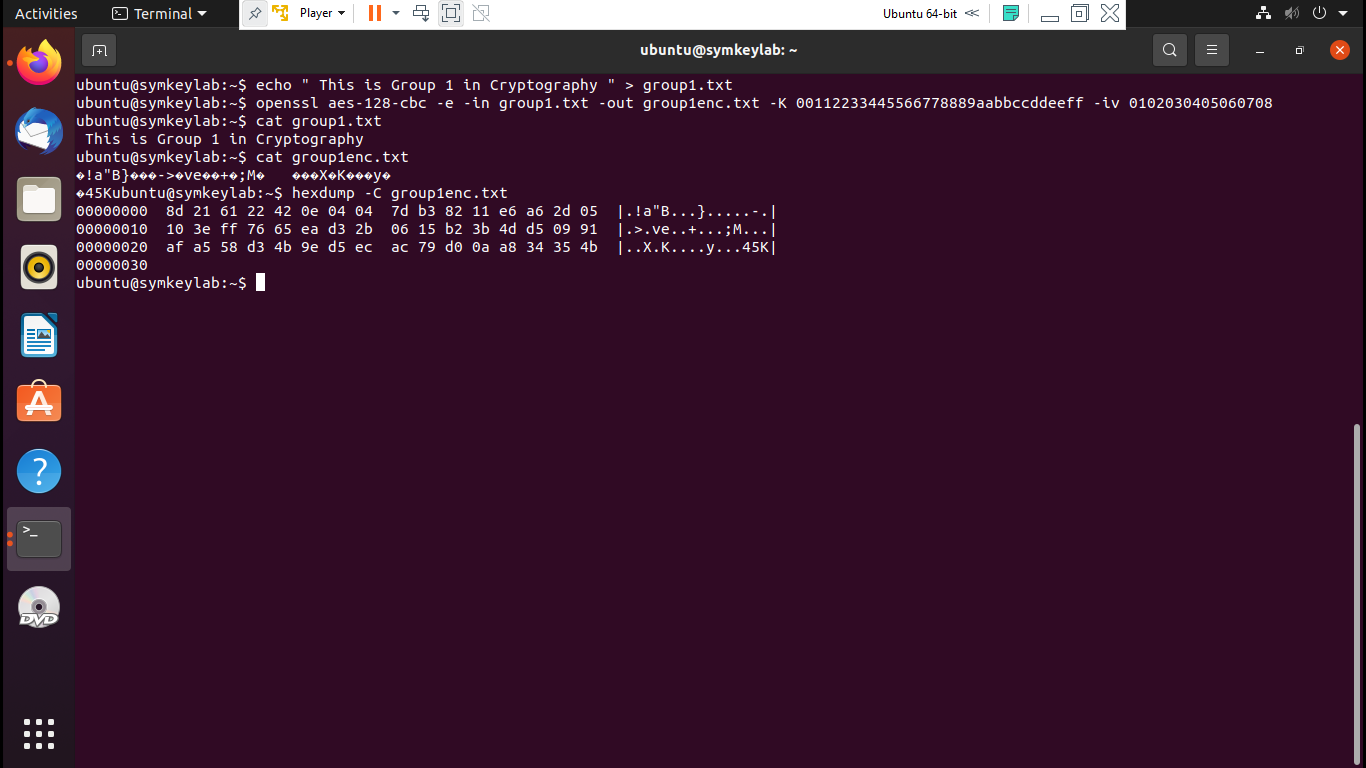
Encrypting the plaintext file (group1.txt) to the cipher text file named, group1enc.txt



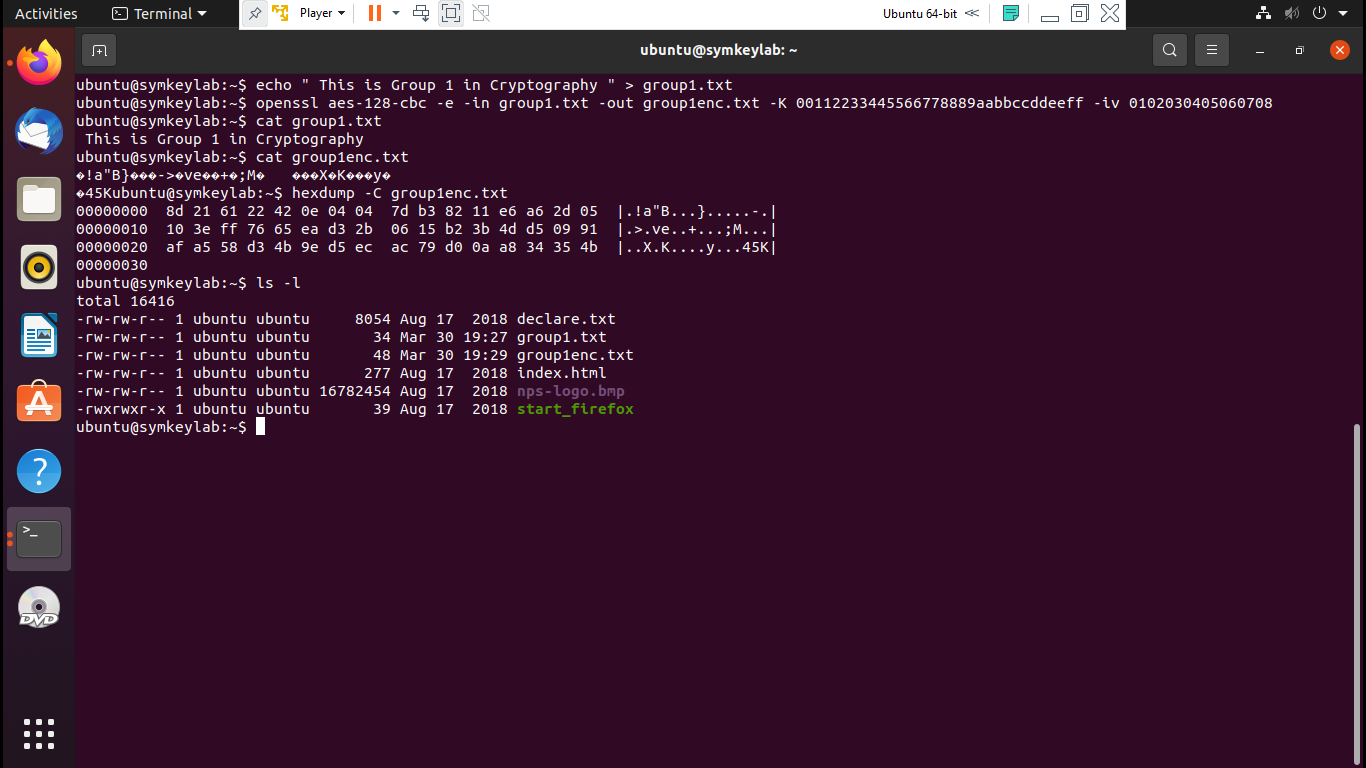
Displaying the cipher text file(group1enc.txt) using cat command. It displays text which is non-readable.



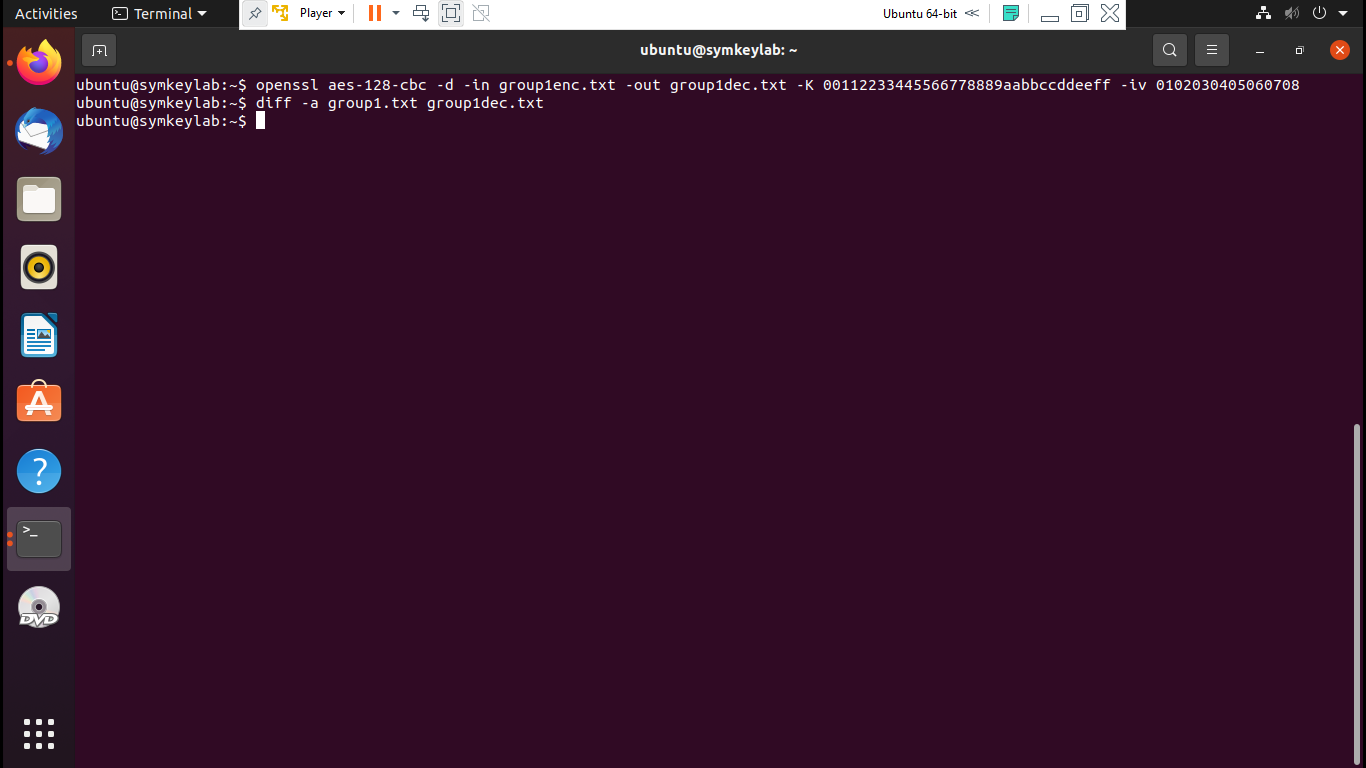
Using hexdump command to display the contents of cipher text file (group1enc.txt). It’s output is human readable.



Using ls –l to display the contents of the directory.



Decrypting the ciphertext file (group1enc.txt) to the new file group1dec.txt.



**Task 2. Encryption Modes**

**ECB Mode**

1. **Describe in your own words, what is the difference between the two images. [2 Marks]**

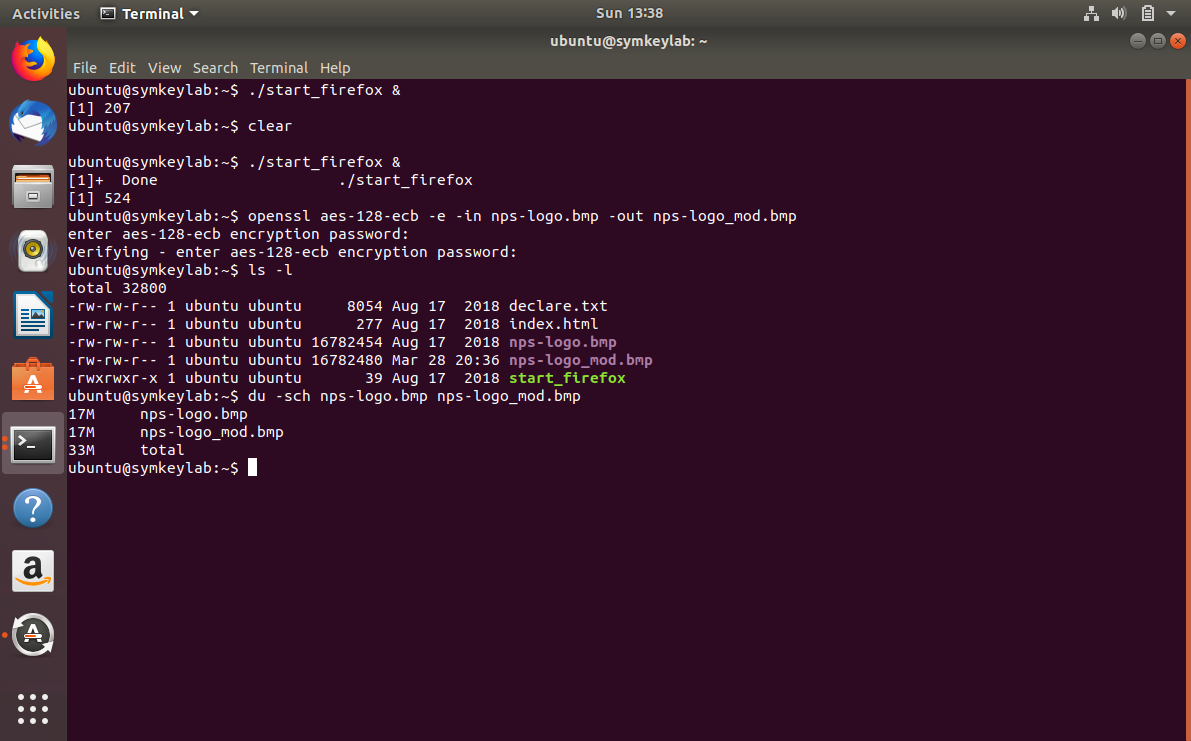
**Answer:**

ECB mode stands for electronic codebook. It is considered to be one of the weakest encryption methods amongst all the groups.

Here in the given two pictures the plaintext is divided into blocks and separated, which means identical blocks of plaintext will have identical block of cipher, because of which it makes the pattern clear. The encrypted data pattern is clearly visible after the encryption, which can be identified.

1. **What is the difference between the size of the logo file and the file after encryption: (attach the screenshot of their size)[3 Marks]**

**Answer:**  
Looking at the size of each file it is clearly visible that each file has almost exactly the same size, which means that plaintext and ciphertext has the same amount of data.



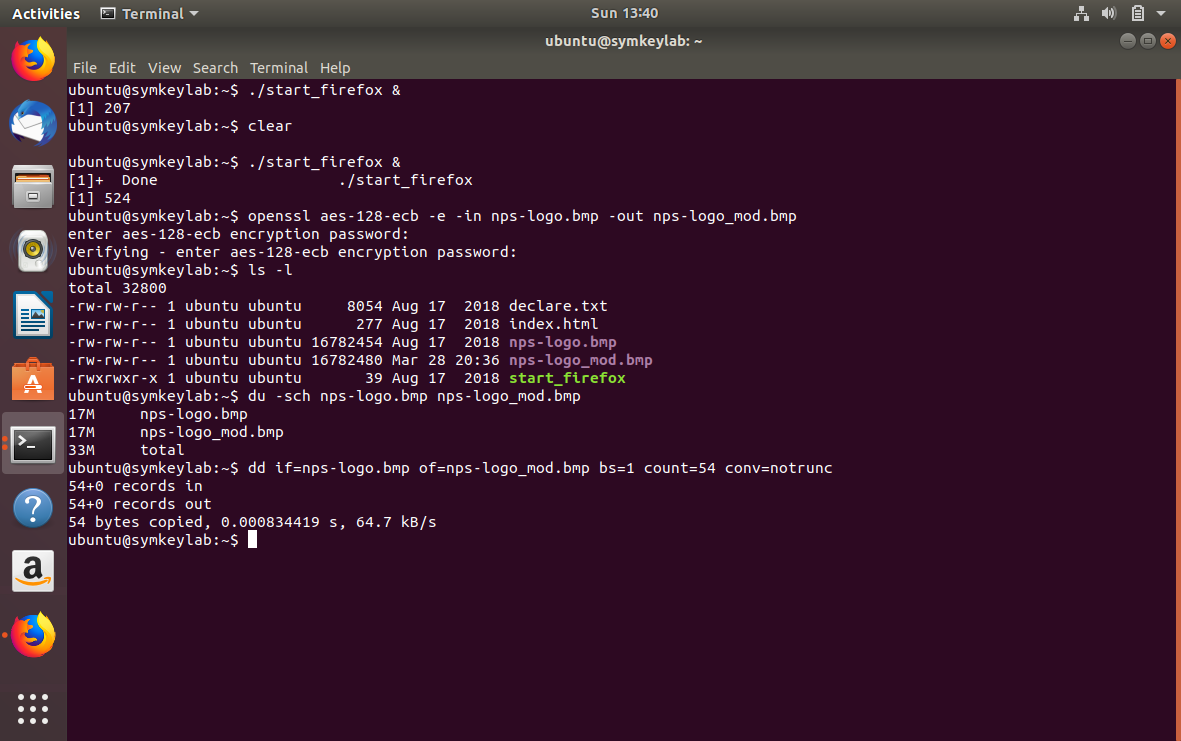
1. **Referring to your observations what characteristic of ECB leads this mode of operation to have such a poor job of encryption? [2 Marks]**

**Answer:**

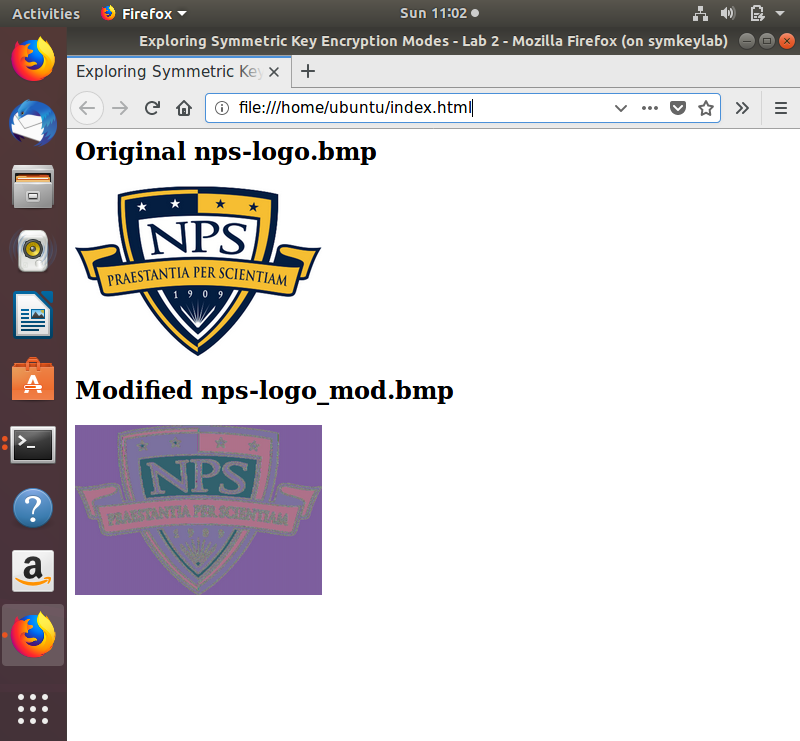
The ECB mode is not semantically secure, where message are divided and encrypted into separately. Our observation related to ECB mode of operation is lack of diffusion as it does not hide the pattern well. The second weak point is ECB uses same key for encryption and decryption.

**Screenshots:**

Starting the firefox browser, encrypting the image in ECB mode, using du –sch command to display the size of the images, and using the dd command.



The logo after encryption in ECB mode.



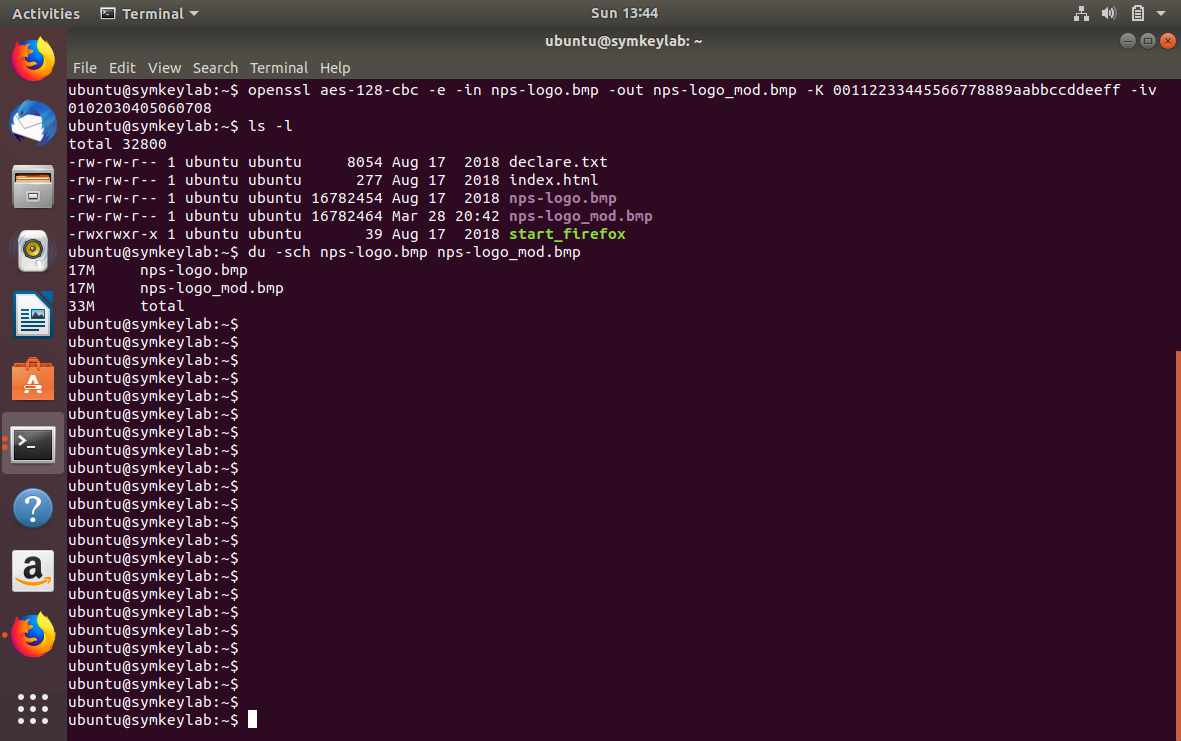
**CBC Mode**

1. **What is the difference between the size of the logo file and the file after encryption: (attach the screenshot of their size) [3 Marks]**

**Answer:**

CBC mode stands for Cipher-block chaining. There are very small differences between two images, which are caused by the vector.

In CBC mode of encryption, each block of ciphertext is used into XORed with the next block plaintext generating a unique cipher due to XOR and initialization vector applied in the first block. The size difference is due to an initialization vector included in the first block.



1. **What is the difference between this logo after encryption and the encrypted logo in ECB mode? [2 Marks]**

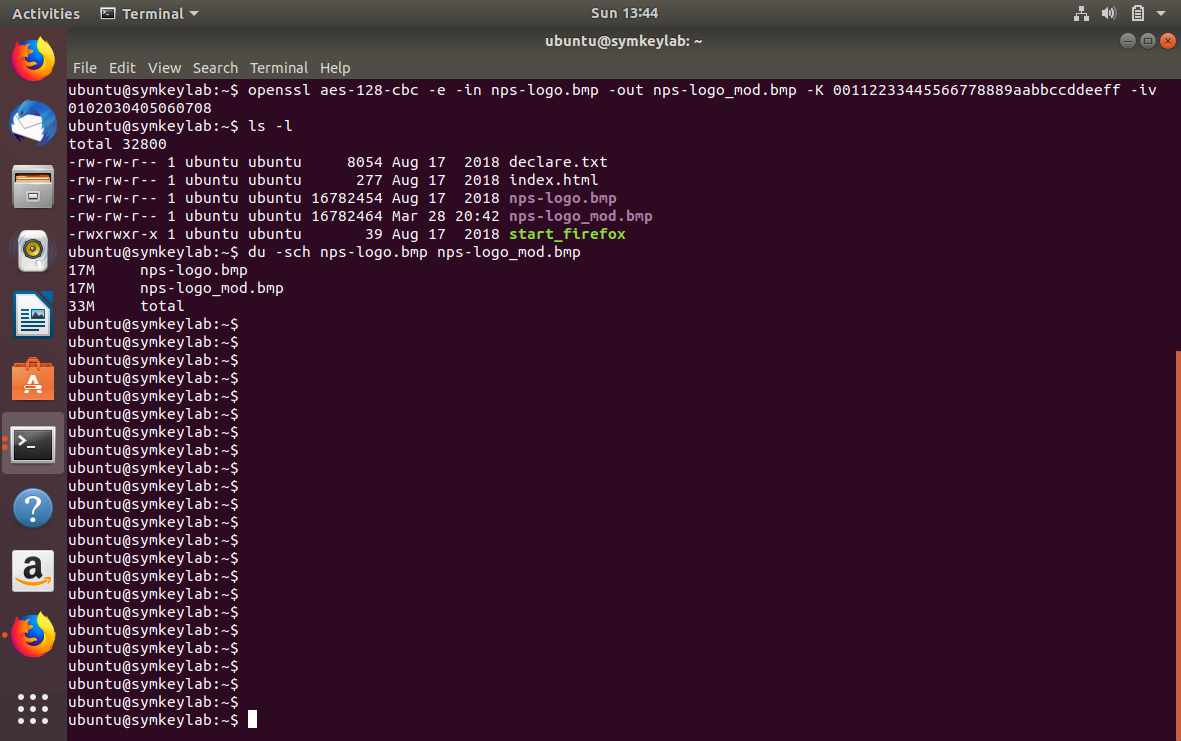
**Answer:**

Differences between the logo after encryption in ECB and CBC modes is as follows:  
1. It is clearly visible that in the ECB mode of encryption, it still possible to identify the image in the cipher text generated, making it weak, while using CBC mode, no pattern is identified.

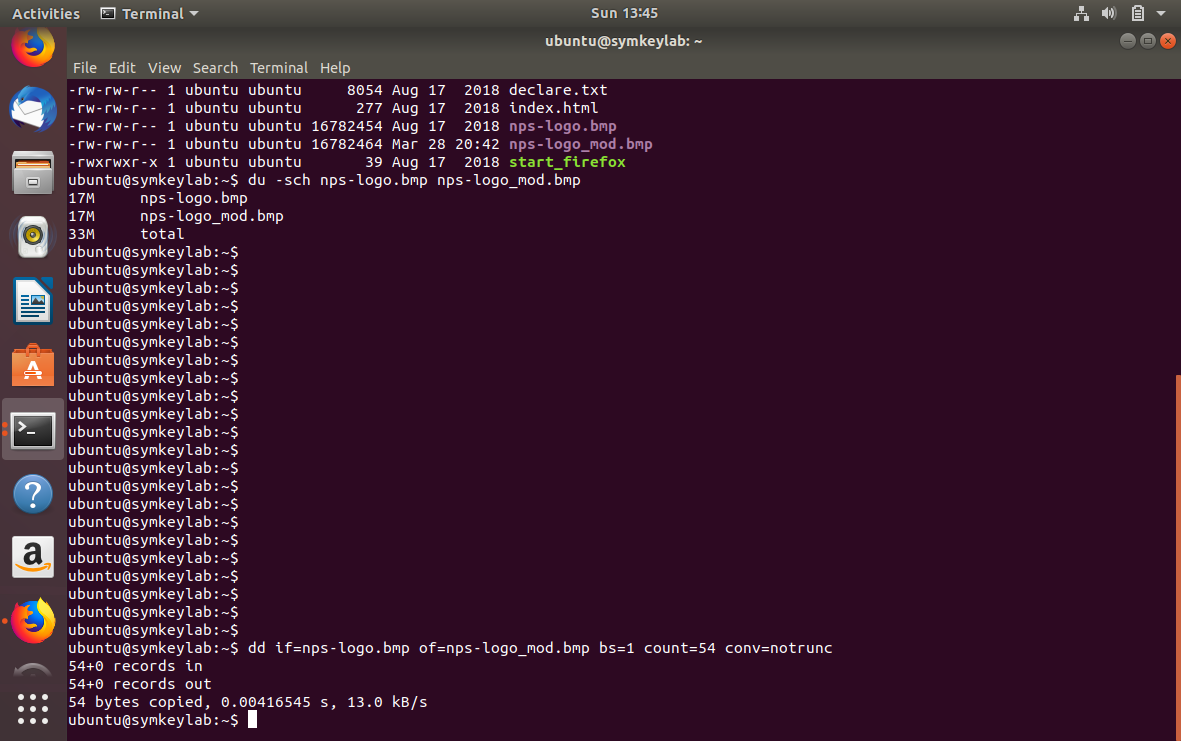
2. Only the color of the logo is changed in the ECB mode, rest we can see the logo clearly. However, in the CBC mode, we can’t identify the image itself.

**Screenshots:**

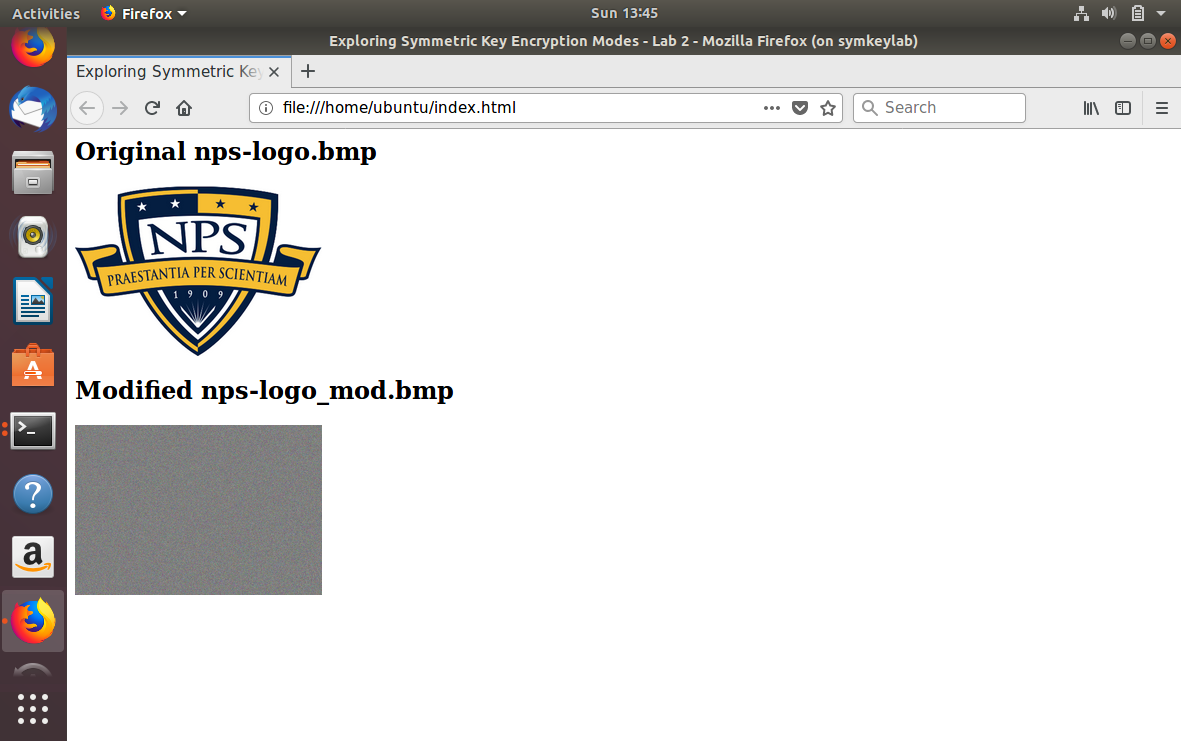
Encrypting the image file using CBC mode, using ls –l to display the contents of the directory, and using du –sch command to display the sizes of the image files.



Using the dd command.



The logo after encryption using CBC mode.

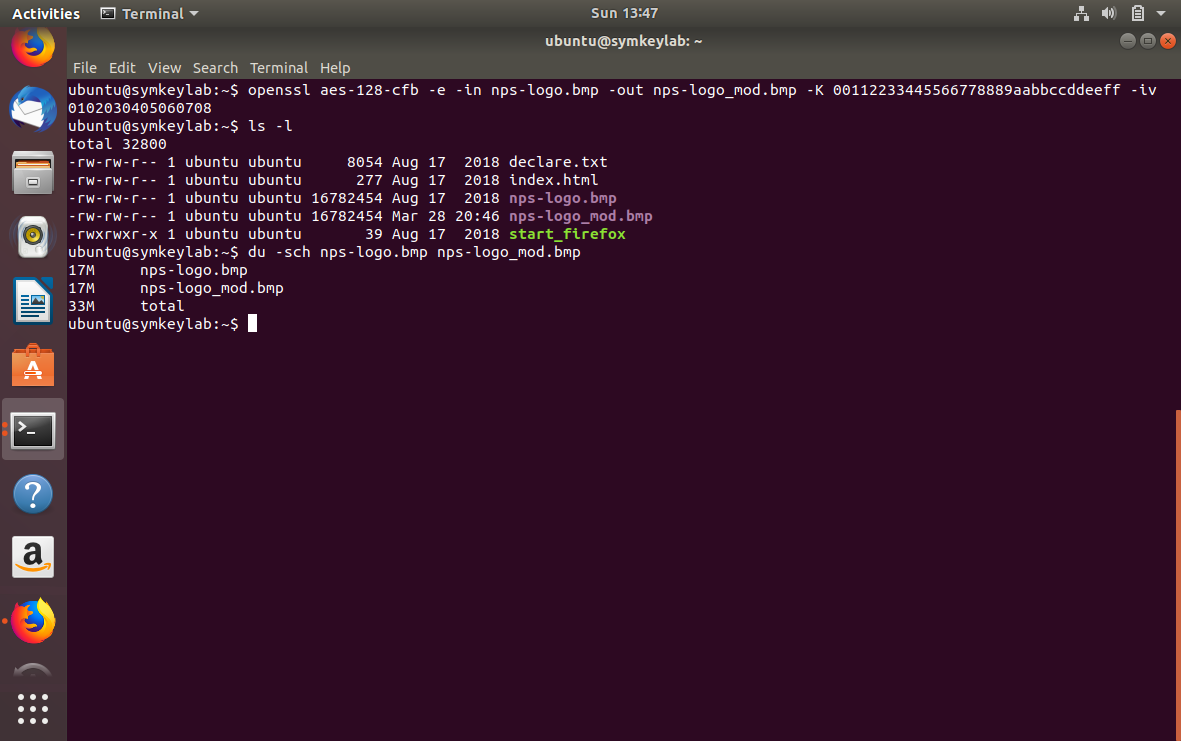


**CFB Mode**

1. **What is the difference between the size of the logo file and the file after encryption: (attach the screenshot of their size) [2 Marks]**

**Answer:**

CFB stands for Cipher Feedback Mode. Here, both the images have the same size in bytes. CFB uses an initialization vector and allows the block encryptor to be used as a stream cipher during the process. It uses the entire block cipher in the process for each round.



1. **What is the difference between logo encrypted with CFB and CBC mode of operation [2 Marks]**

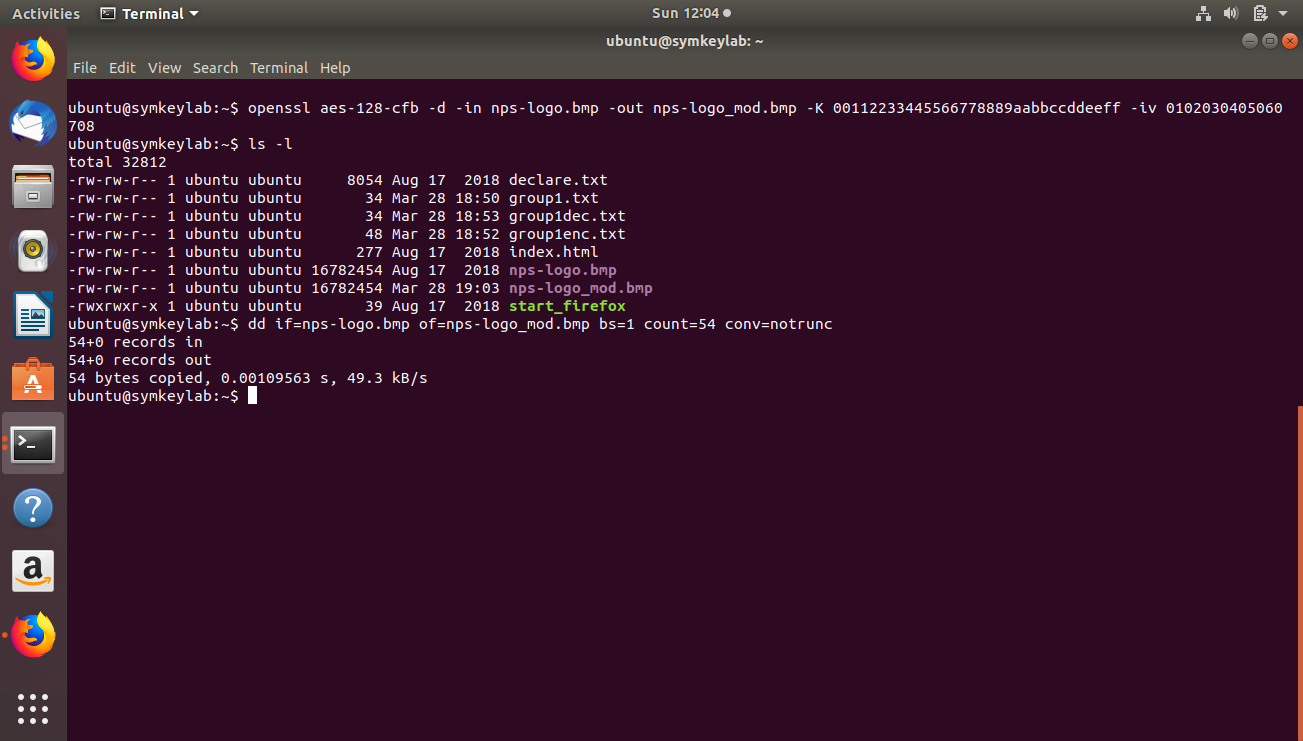
**Answer:**

The CFB is same as EBC generates patterns that facilitate the identification from an attacker. The CFB performs the XORed function between plaintext and ciphertext after block cipher encryption occurs, while after CBC performs XOR function between plaintext and ciphertext before block cipher occurs during each round.

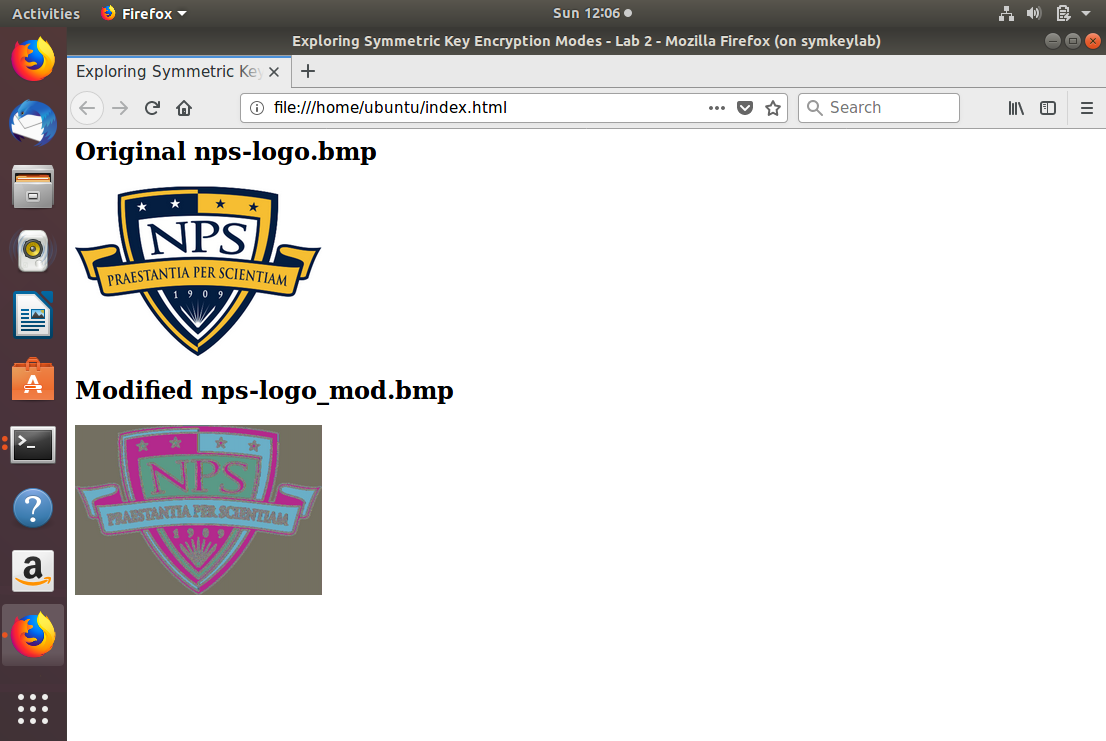
Differences between the logo after encryption in CFB and CBC modes is as follows:  
1. It is clearly visible that in the CFB mode of encryption, it still possible to identify the image in the generated cipher text, while using CBC mode, no pattern is identified.

2. Only the colour of the logo is changed in the ECB mode, rest we can see the pattern of the logo clearly. However, in the CBC mode, we can’t identify the image itself.

**Screenshots:**

Encrypting the image file in CFB mode, using ls –l to display the contents of the directory, and using dd command.

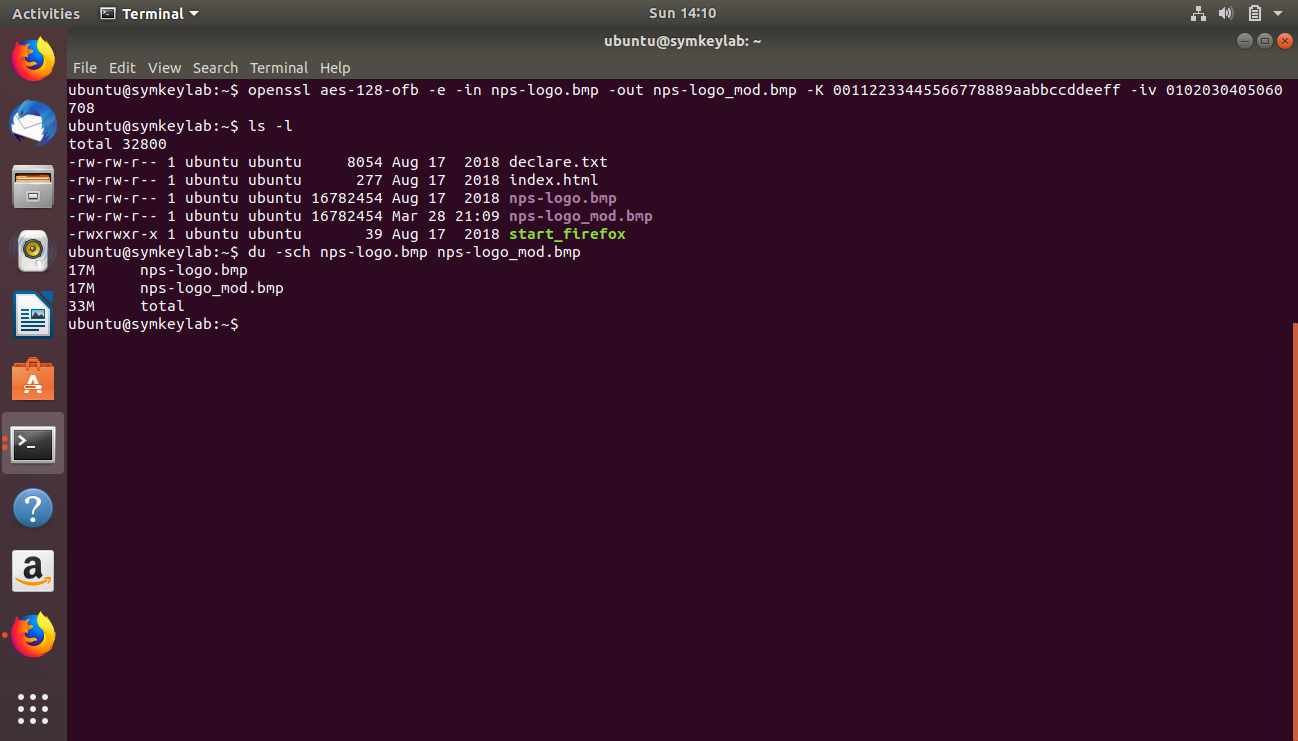
The logo after encryption using CFB mode.



**OFB Mode**

1. **What is the difference between the size of the logo file and the file after encryption: (attach the screenshot of their size) [2 Marks]**

**Answer:**  
OFB stands for Output feedback. Here both the images have exactly the same size. OFB doesn´t use padding data and generates keystream blocks that are XORed with the plaintext portion (block) and then generates the ciphertext. This method uses a block encryptor to be used in stream encryptor.



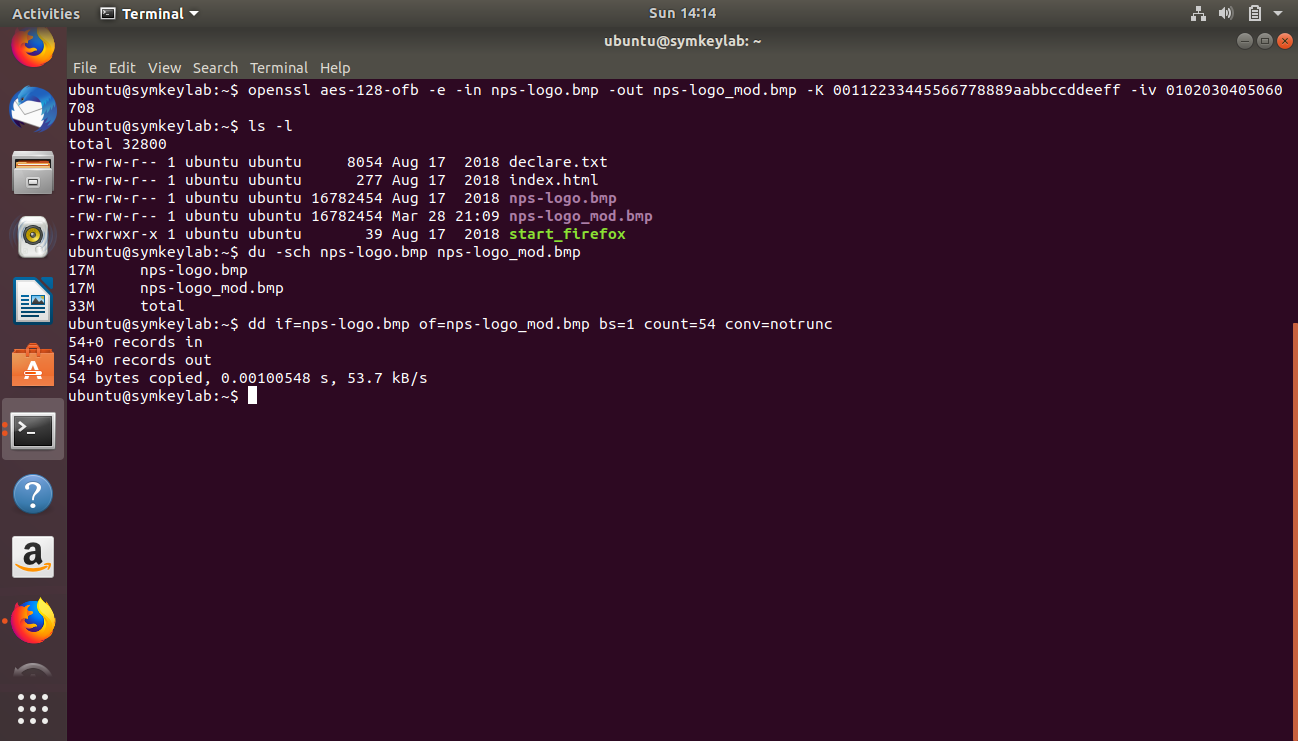
1. **What do you think the advantage of OFB is? (Name at least 1) [2 Marks]**

**Answer:**

The major advantage of OFB is the error-correction once a bit error will not generate an avalanche of propagation. Each block depends on the block generated before (encryption), making it a strong cipher. This method encrypts the initialization vector in the first block and after that XOR with the plaintext, the next round will use the per-result encrypted to XOR with the next block of plaintext. This type of encryption makes it difficult to use packages in a Chosen Plaintext attack.

**Screenshots:**

Encrypting the image file in OFB mode, using ls –l to display the contents of the directory, using du –sch to display the size of the files, and using dd command.



The logo after encryption using OFB mode.



1. **Which mode of operation does a better job of encrypting the logo and why? [3 Marks]**

**Answer:**

OFB is the strongest mode of encryption as it doesn't generate any pattern and each block encrypted will be different even if the plaintext block is identical.

1. **Why do you think the sizes of each encrypted logo are larger than the plaintext logo files? [2 Marks]**

**Answer:**

Adding a initialization vector using a XOR function before or after the encryption could generate more data for each round, which increases file size.

Depending on the encryption method XOR could occur before or after the encryption.

1. **Are there any encrypted logos that have the same size as plain text logo? Which ones? And why? [3 Marks]**

**Answer:**  
Only the logo using the OFB method. This method encrypts the initialization vector in the first block and after that XOR with the plaintext, the next round will use the per-result encrypted to XOR with the next block of plaintext, generating a ciphertext with the same size.

1. **What happens if you do not use IV in modes of operation that require this option? What is the role of IV in these modes? [3 Marks]**

**Answer:**

Using IV helps the encryption/decryption to be unpredictable. Adding a random initialization vector in the first block of encryption will help to prevent an attacker from comparing different messages looking for patterns and statistical values. IV is also considered as a nonce increasing unpredictability.

# 

# 

# **Task 3. Error Propagation during Decryption**

**Preparation**

1. **What does Error propagation mean? [1 Marks]**

**Answer:**

Let us suppose if in a cipher text, there is a one bit error, that means, if a single bit say 1 is erroneously converted to 0, then that incorrect cipher text produces a plaintext output (during decryption) that may have more than one erroneous bit. This phenomenon is known as Error propagation.

1. **What is the number of AES blocks needed to encrypt the Declaration of Independence? Show your work. [3 Marks]**

**Answer:**

Number of bytes used by the file declare.txt = 8054 bytes

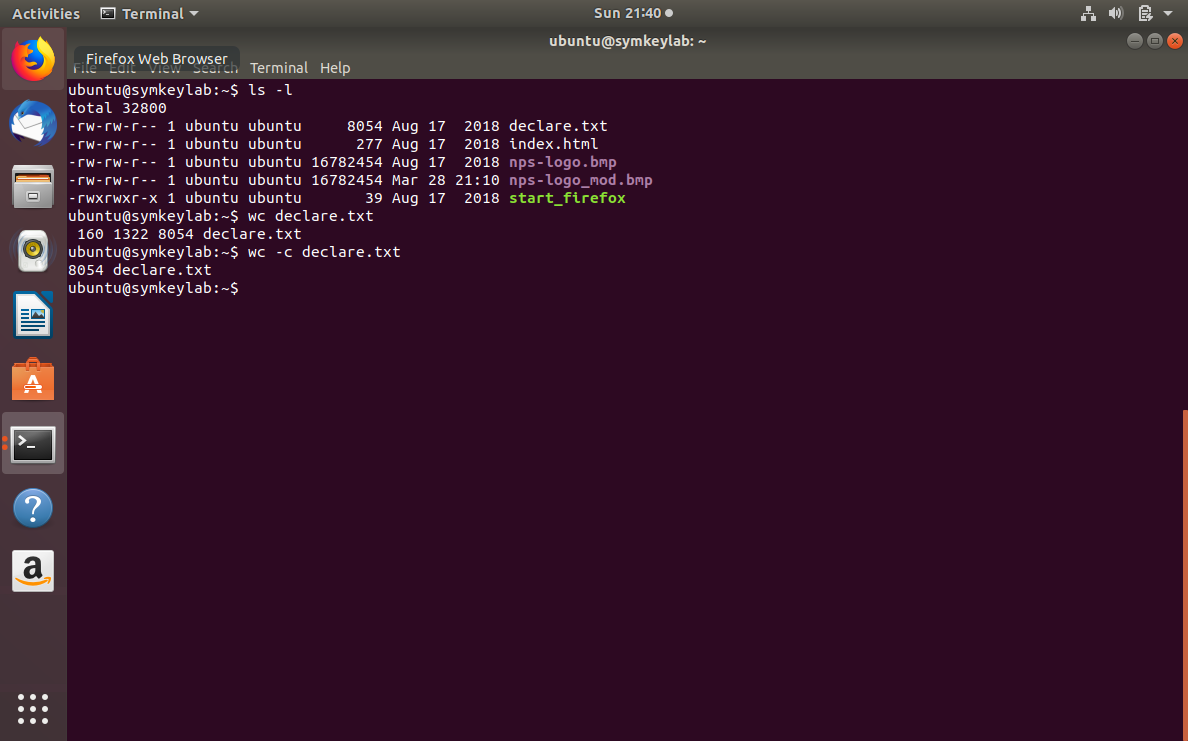
Number of bits used by the file declare.txt = 8054\*8 bits

Size of one AES block = 128 bits

Therefore, 128 bits makes => 1 AES block, then

8054\*8 bits make => (8054\*8)/128 => 503.375 => 504 AES blocks

Hence, 504 AES blocks will be required to encrypt the Declaration of Independence (declare.txt) file.



1. **How many characters can AES encrypt in one block? (Hint: it takes one byte to encode a character). Show your work. [2 Marks]**

**Answer:**

Block size of AES = 128 bits

It takes 8 bits(one byte) to encode 1 character.

Therefore, 128 bits (in one AES block) will encode 128/8 characters,

that is, 16 characters.

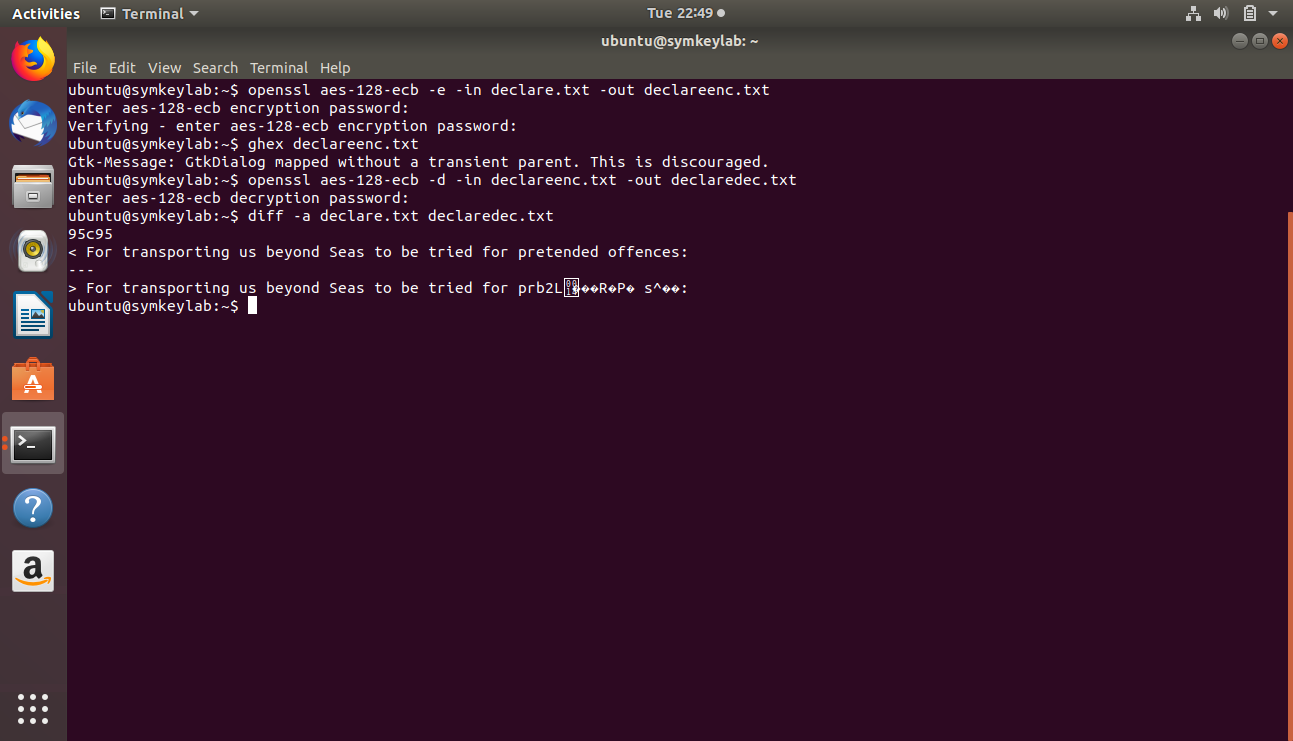
Hence, AES can encrypt 16 characters in one block.

**ECB Mode**

1. **Describe the corruption that occurred when a cipher text (with one-bit of corruption) was decrypted using ECB mode. [2 Marks]**

**Answer:**

In ECB mode, the error in the cipher text block only propagates to the corresponding plain text block. When the cipher text was decrypted using ECB mode, the corruption that occurred with one-bit of corruption is found using the *diff* command. The output of the *diff* command is 95c95, which means there is a difference between the 95th line of first file (plain text file) and the 95th line of the second file (cipher text file), and to make the files same, change the 95th line of the first file to the 95th line of the second file or vice versa. It also displays the text where the corruption occurred. That is, the text “For transporting us beyond Seas to be tried for pretended offences:” corresponds to the 95th line of the first file that is the plain text file (declare.txt) and the other text as shown in the image below corresponds to the 95th line of the second file (declaredec.txt).



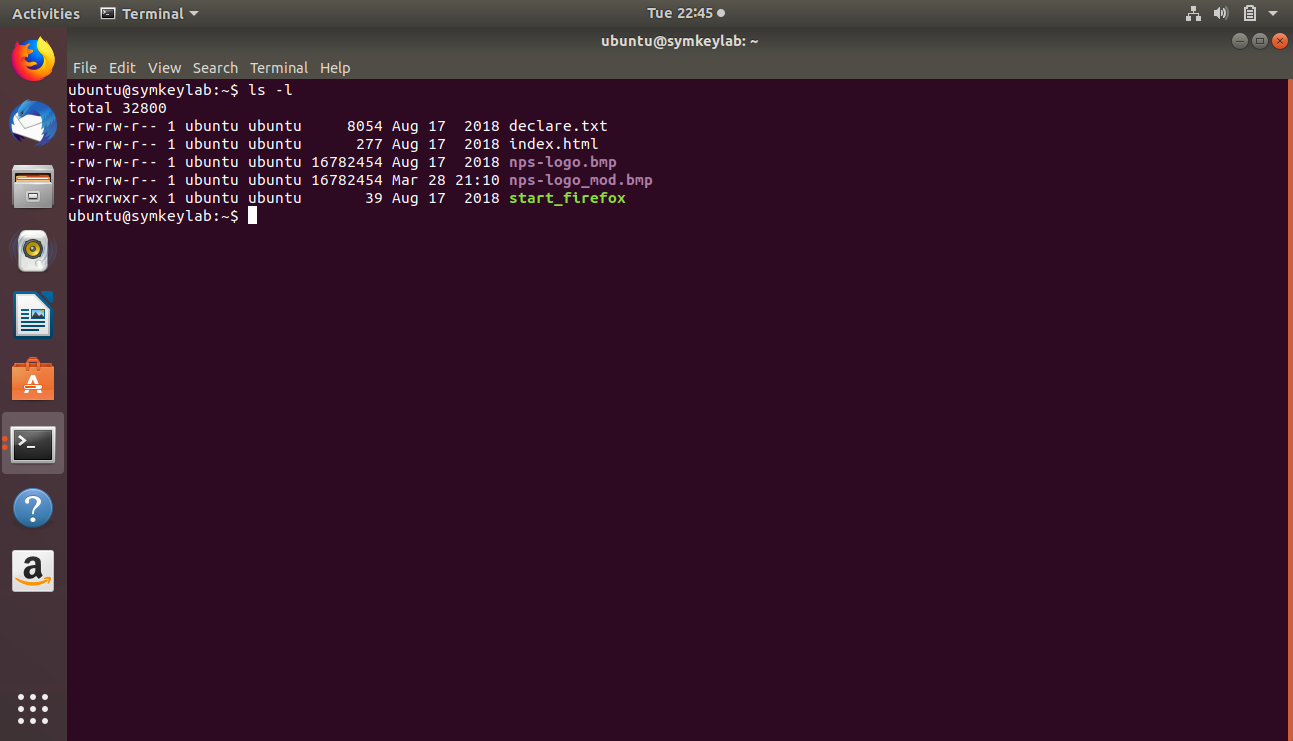
1. **Why do you think ECB mode corrupted the plaintext file? [3 Marks]**

**Answer:**

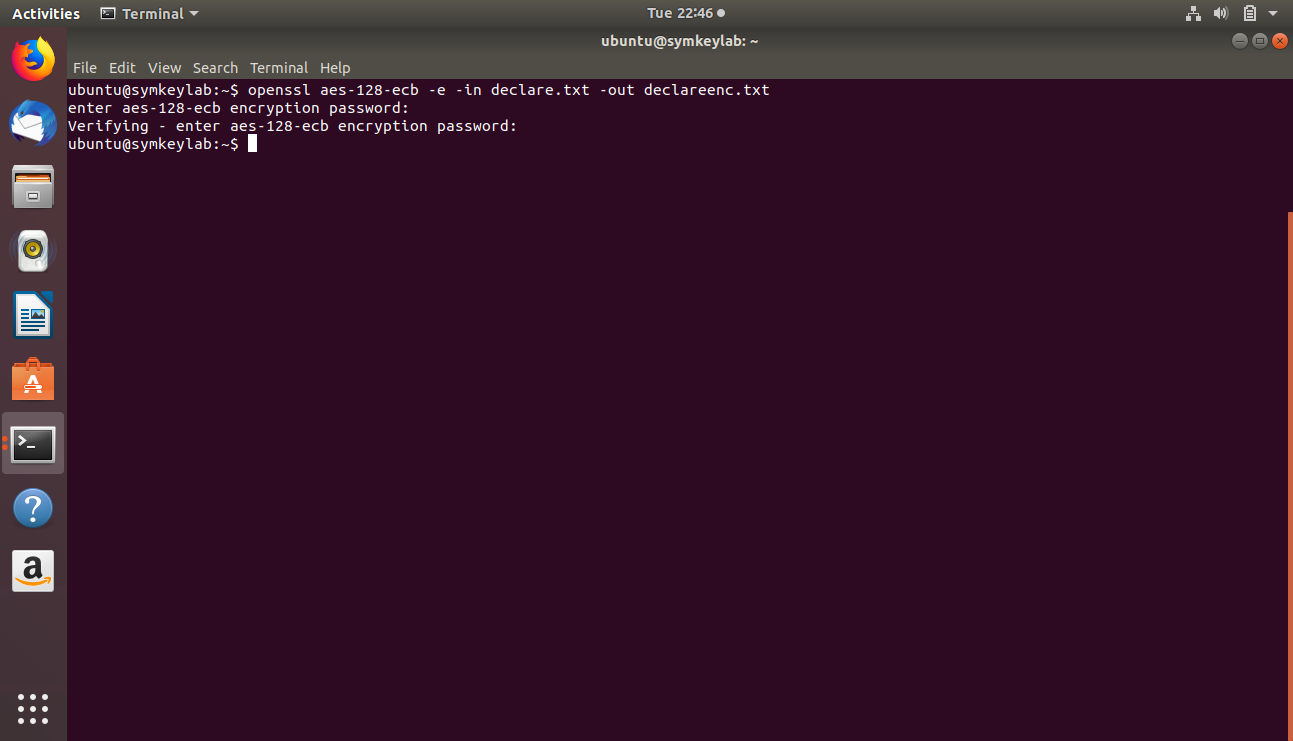
In the ECB mode, the error in the cipher text block only causes the deciphering error in the corresponding plain text block. Each block in the ECB mode is independently encrypted and the decryption takes place in the same way but in reverse order. Since an error in a single bit affects the other bits in the same block but will not affect the other blocks because of the independency in the encryption of one plain text block and the other plain text block. However, the block corresponding to where the error occurs will be wholly affected due to the dependence of the bits on each other for encryption.

**Screenshots:**

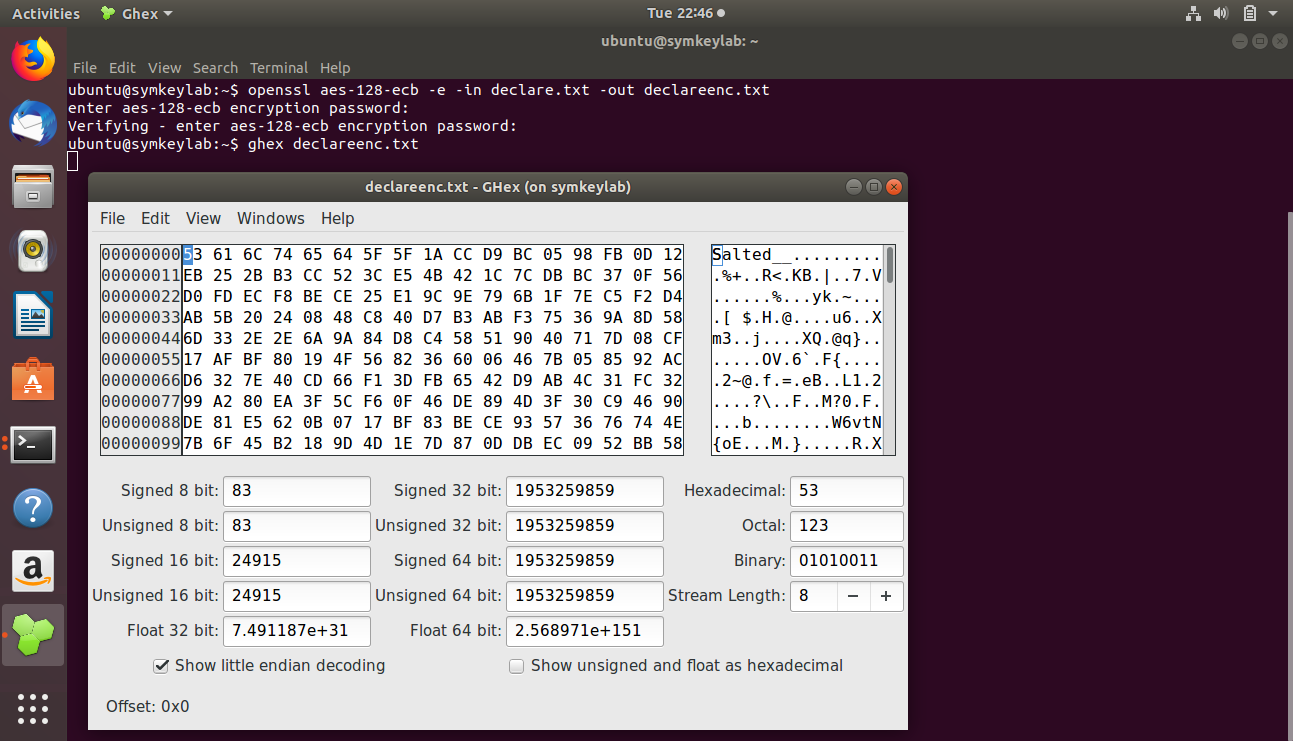
Displaying the files using ls –l command.



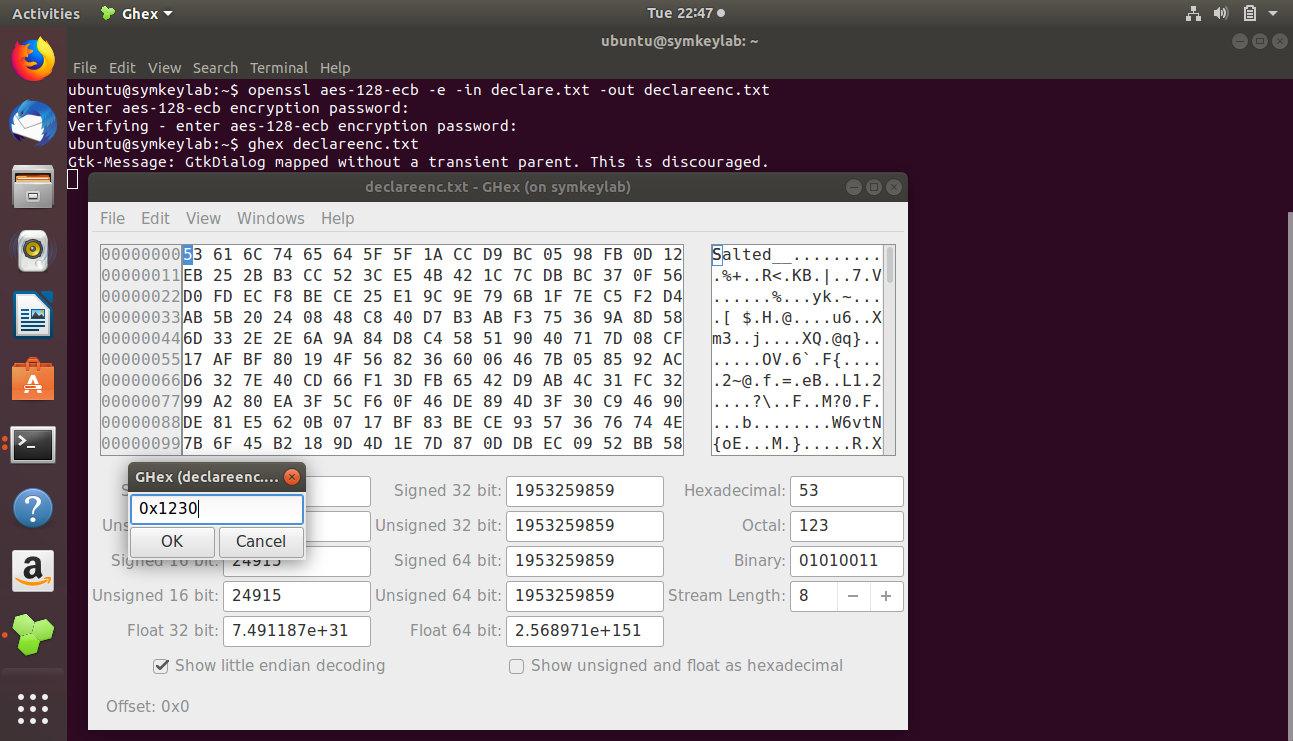
Encrypting the plaintext file (declare.txt) using the ECB mode. The ciphertext file generated is declareenc.txt.



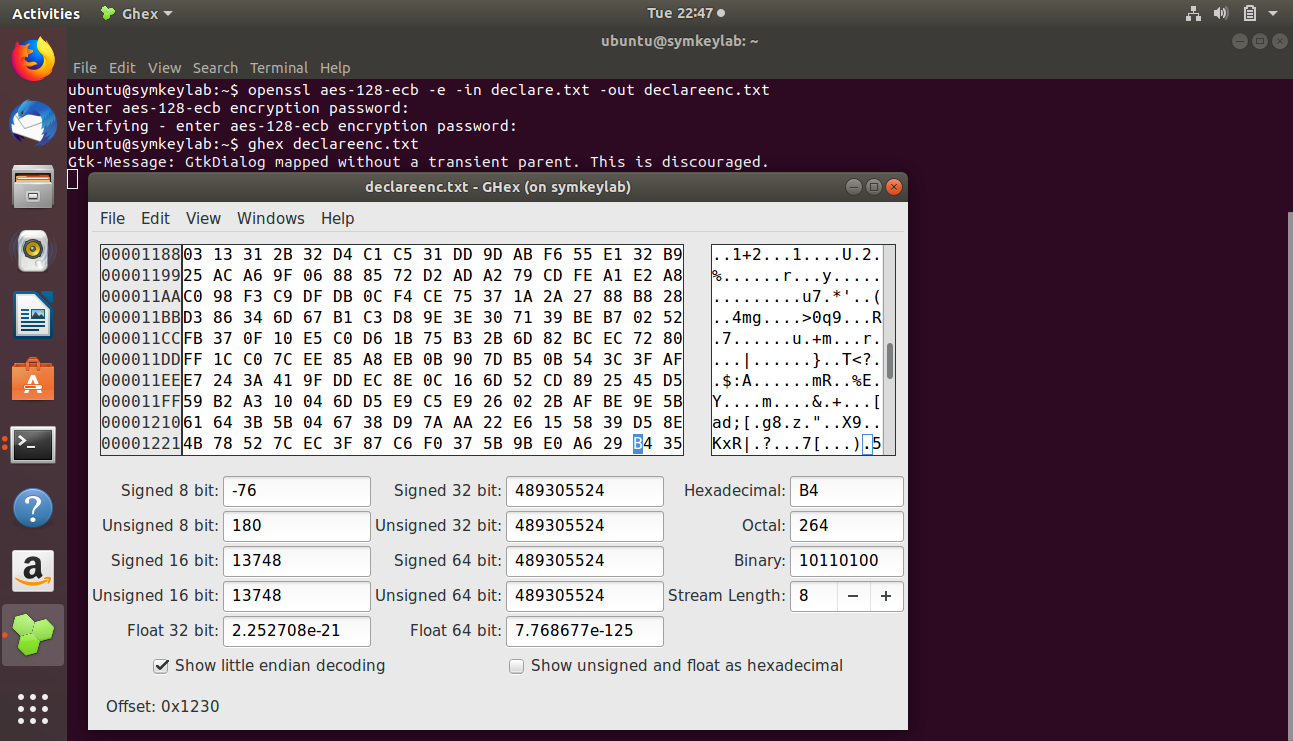
Using the ghex command.



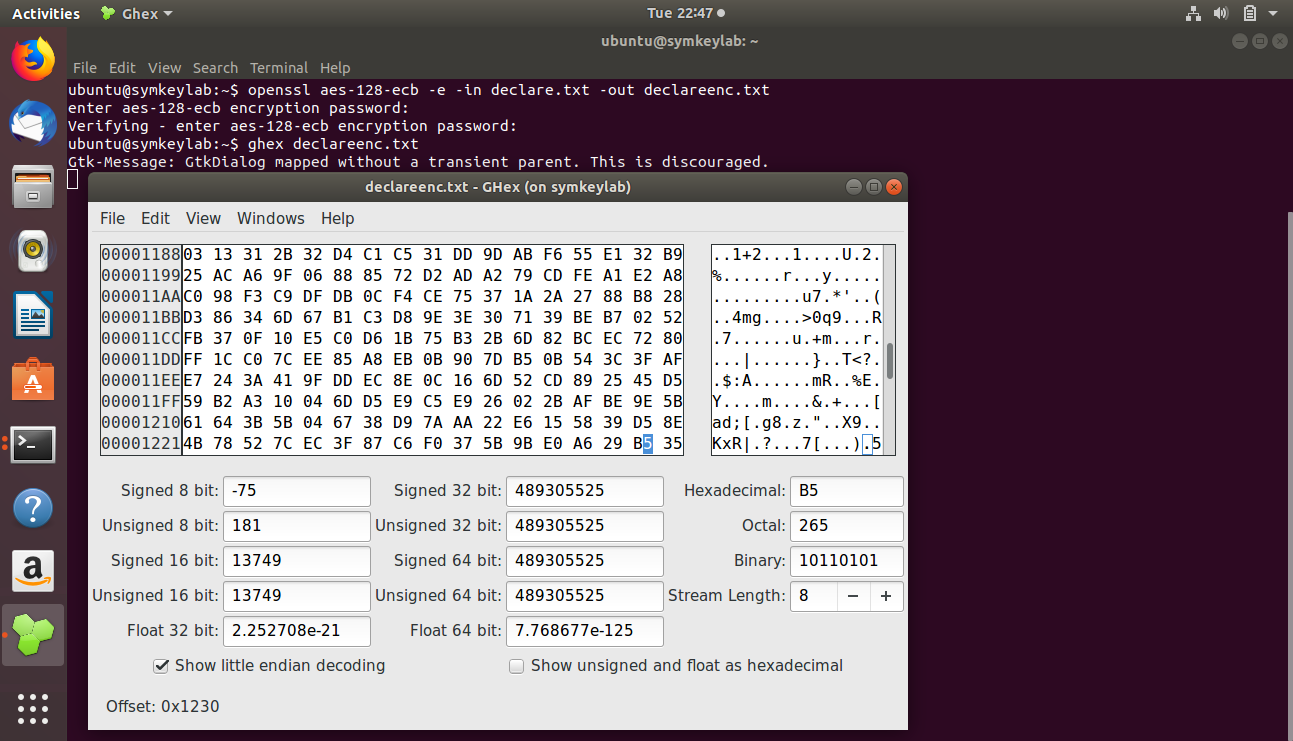
Setting the offset to 0x1230.



It displays the bit *prior* to changing the bit in the file.



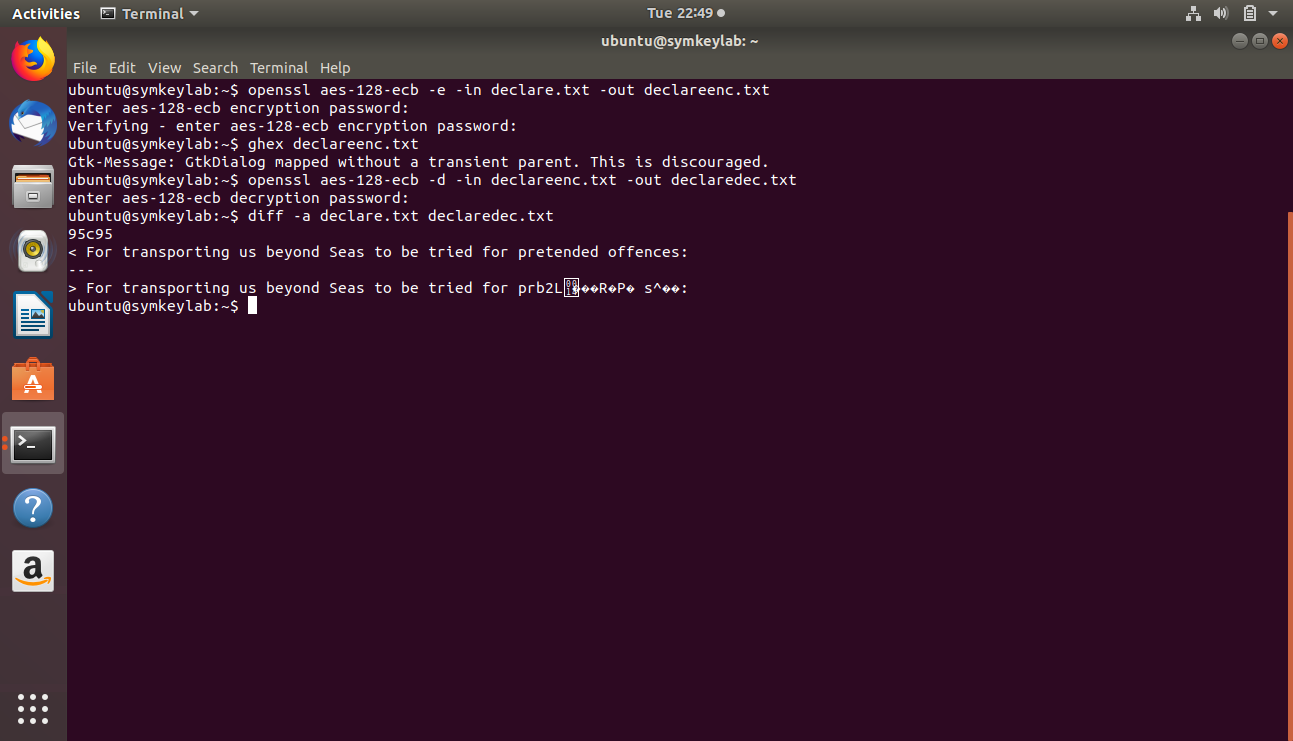
The bit change is done by converting 4 to 5(as highlighted in the image below).



Decrypting the cipher text file (declareenc.txt) to the new file declaredec.txt.



Using diff command to find the difference between the plain text file(declare.txt) and the new file after decryption(declaredec.txt).



**CBC Mode**

1. **Describe the corruption that occurred when a ciphertext (with one-bit of corruption) was decrypted using CBC mode. [2 Marks]**

**Answer:** When the cipher text was decrypted using CBC mode, the corruption that occurred with one-bit of corruption is found using the *diff* command. The output of the *diff* command is 95,97c95, which means there is a difference between the 95th line and 97th line of first file (plain text file) and the 95th line of the second file (cipher text file), and to make the files same, change the 95th line and the 97th line of the first file to the 95th line of the second file or vice versa. It also displays the text where the corruption occurred as shown in the image. The lines starting with “<” sign correspond to the first file (plain text file) and the line starting with “>” sign correspond to the second file (cipher text file).



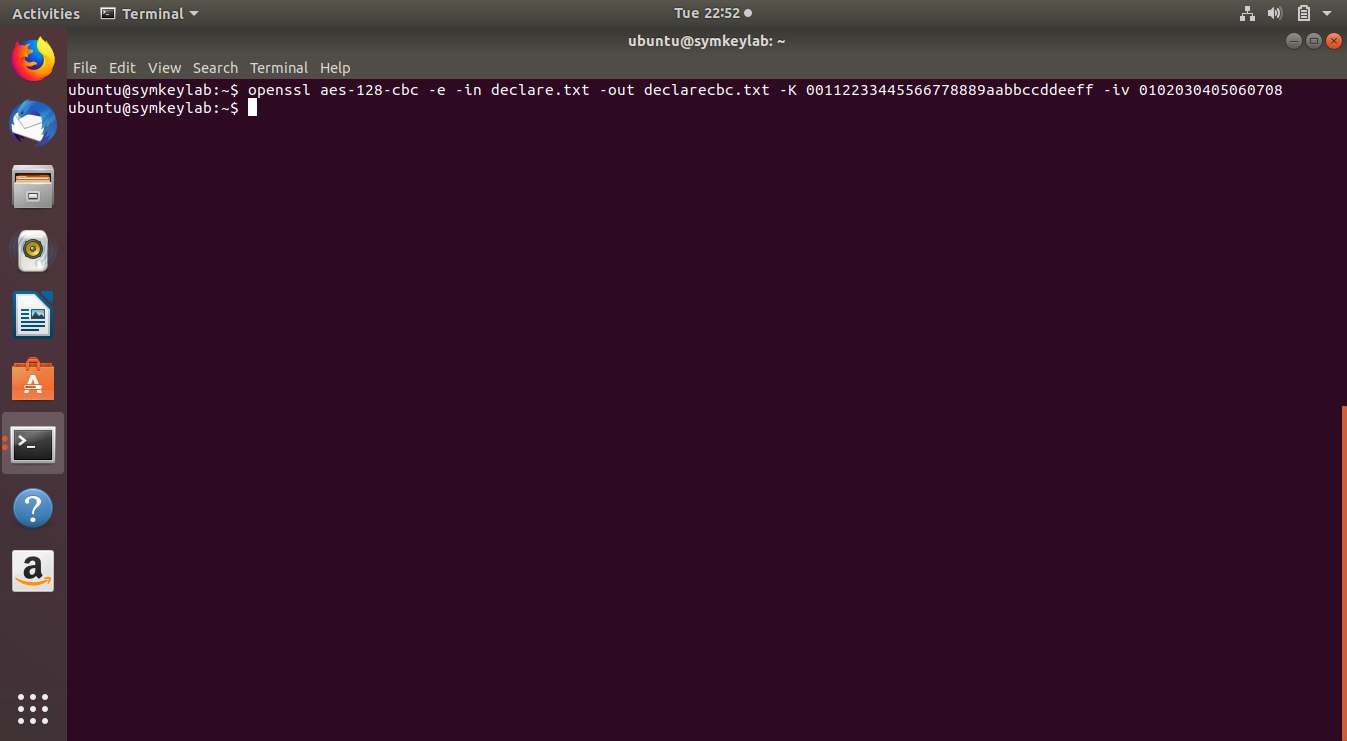
1. **Why do you think CBC mode corrupted the plaintext file? [2 Marks]**

**Answer:**

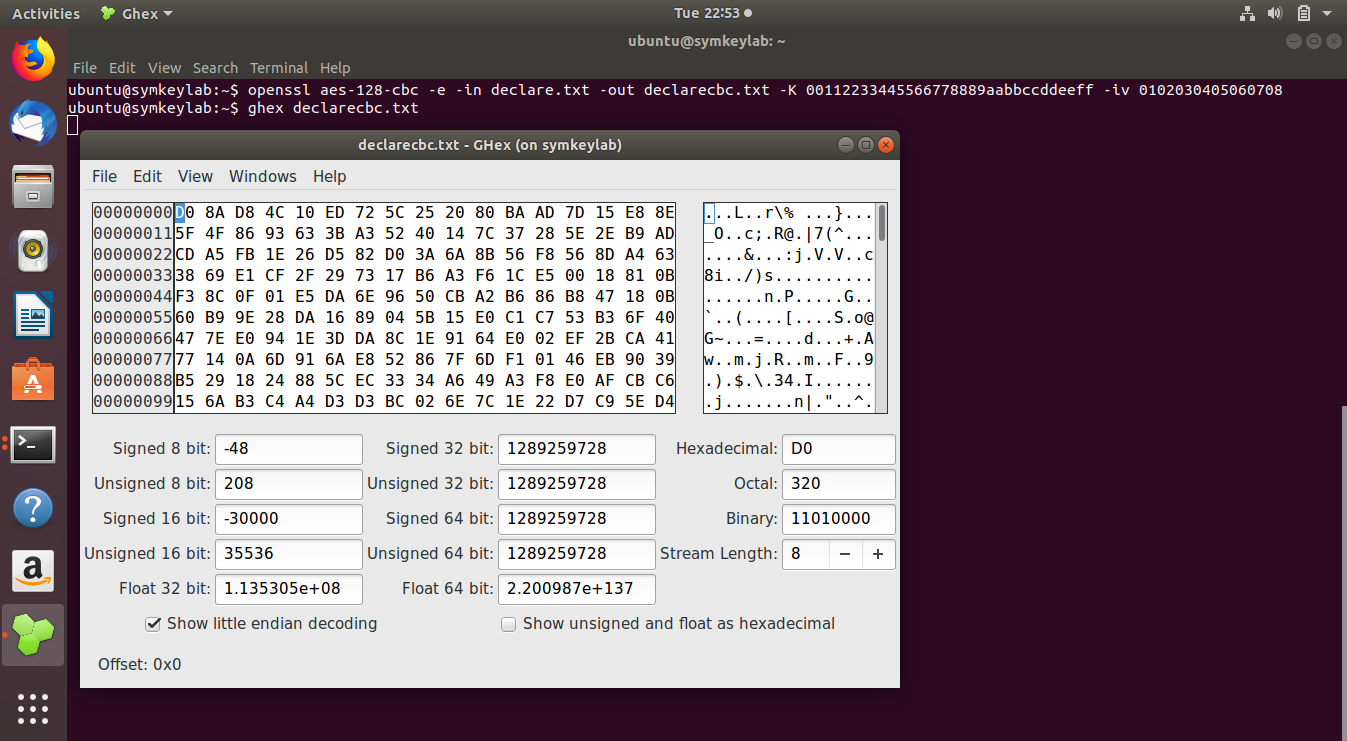
In the CBC mode, a single bit error in the cipher text block say cx may change the corresponding bit in the next plain text block say px+1 but has significant changes in the corresponding plain text block px. This is because the encryption of a plain text block depends on the cipher text of the previous plain text block.

**Screenshots:**

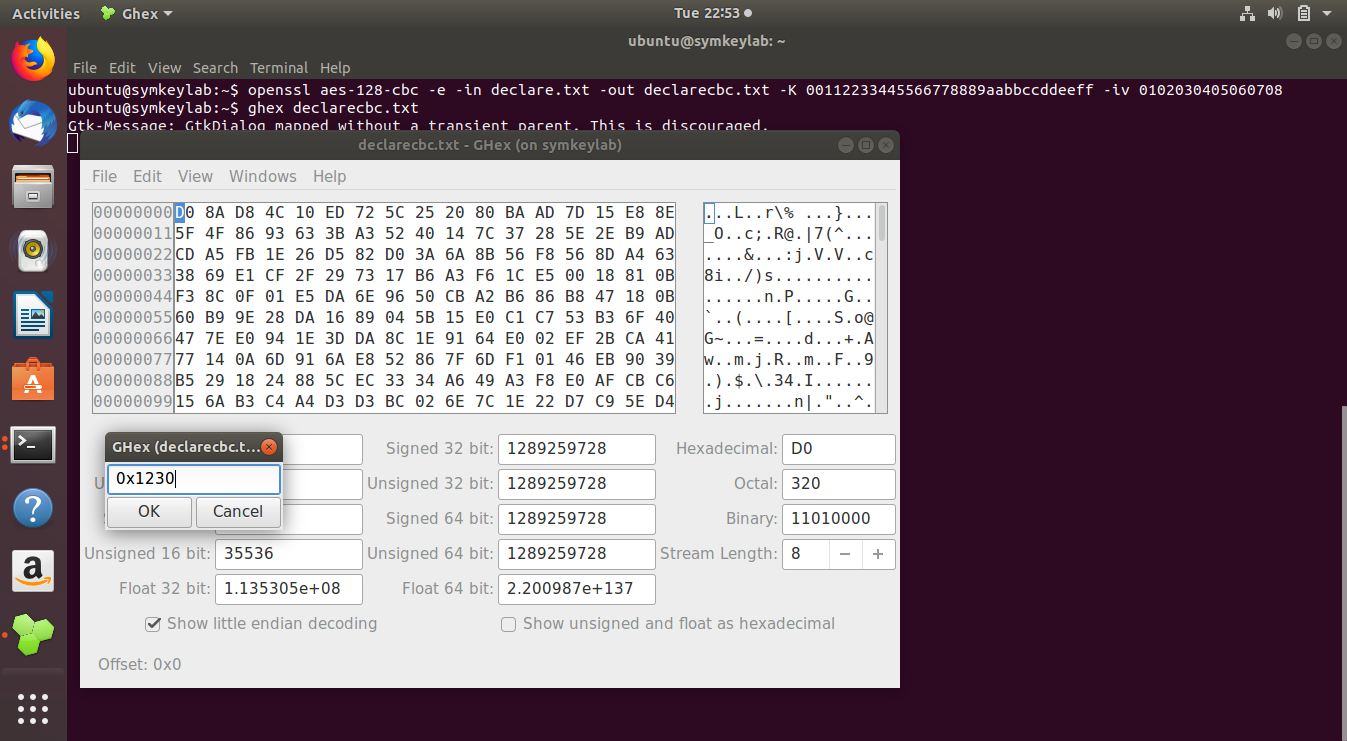
Encrypting the plaintext file (declare.txt) using the CBC mode. The cipher text file generated is declarecbc.txt.



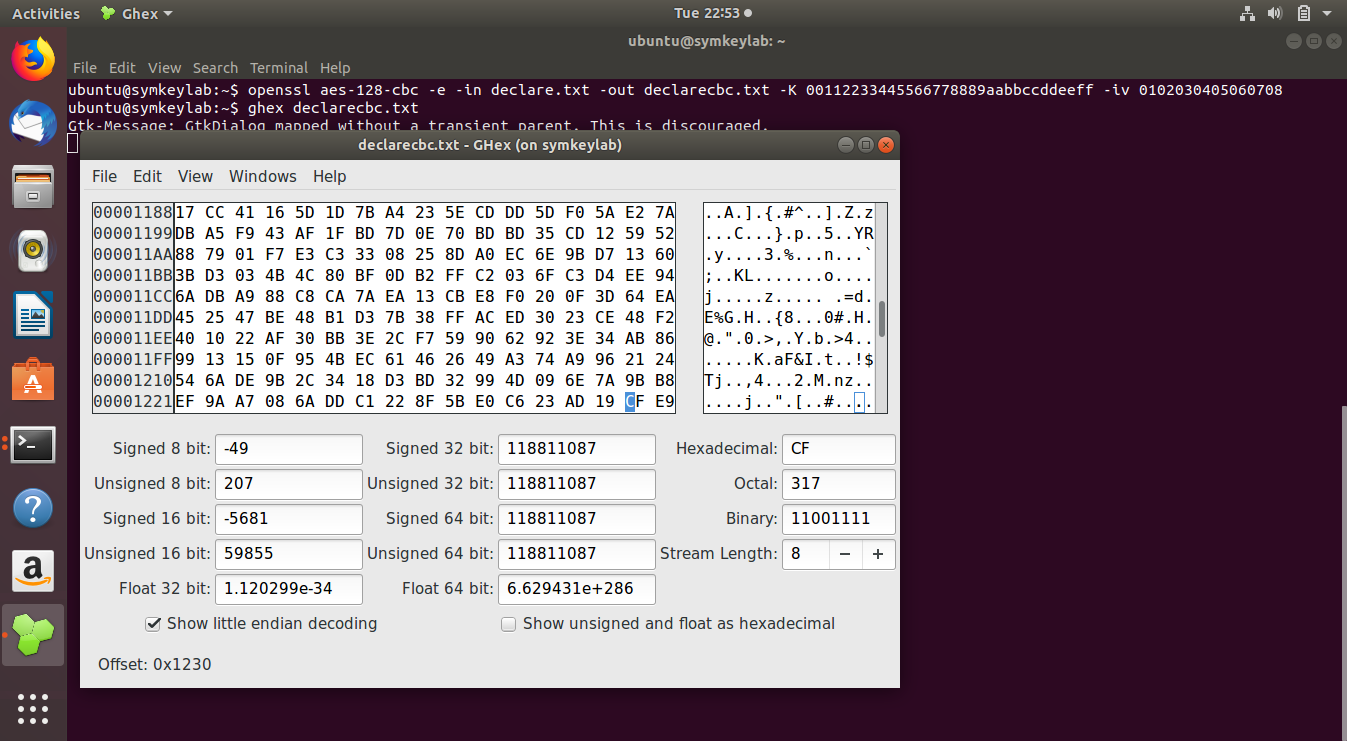
Using the ghex command.



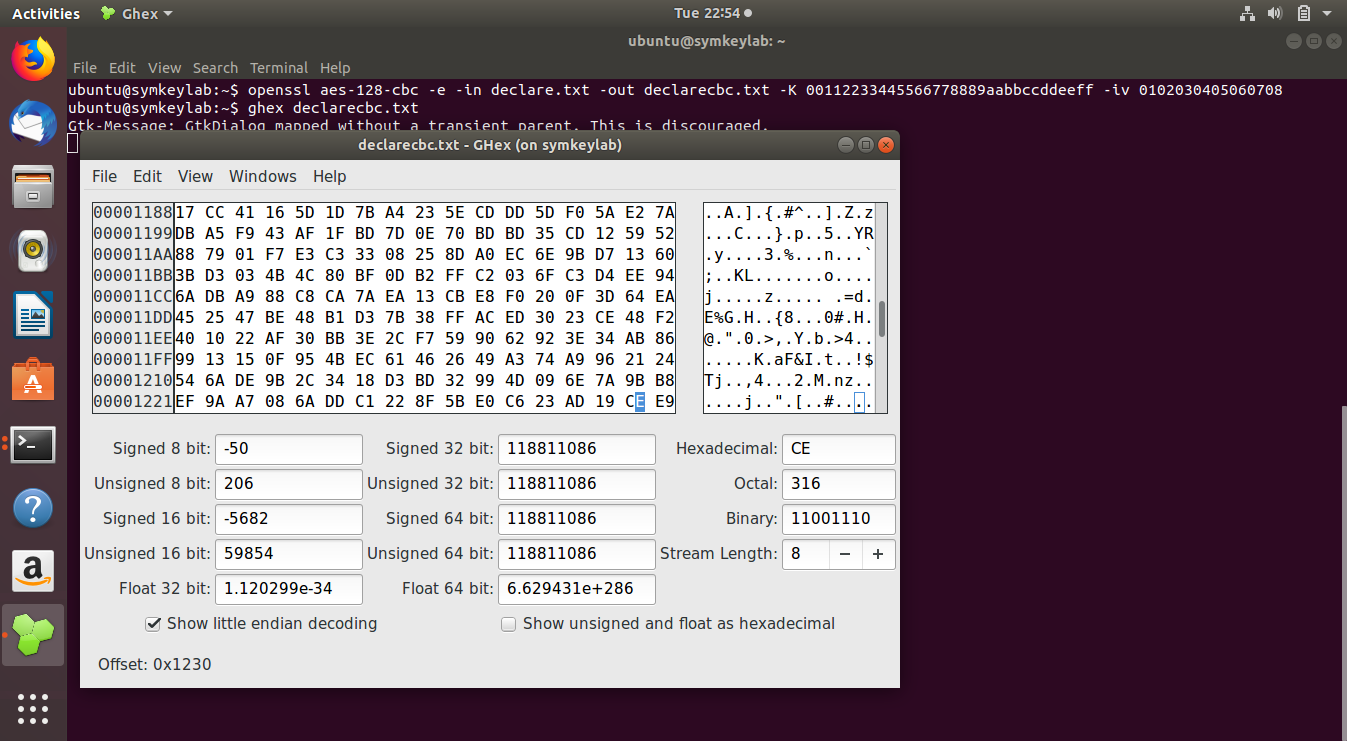
Setting the offset to 0x1230.



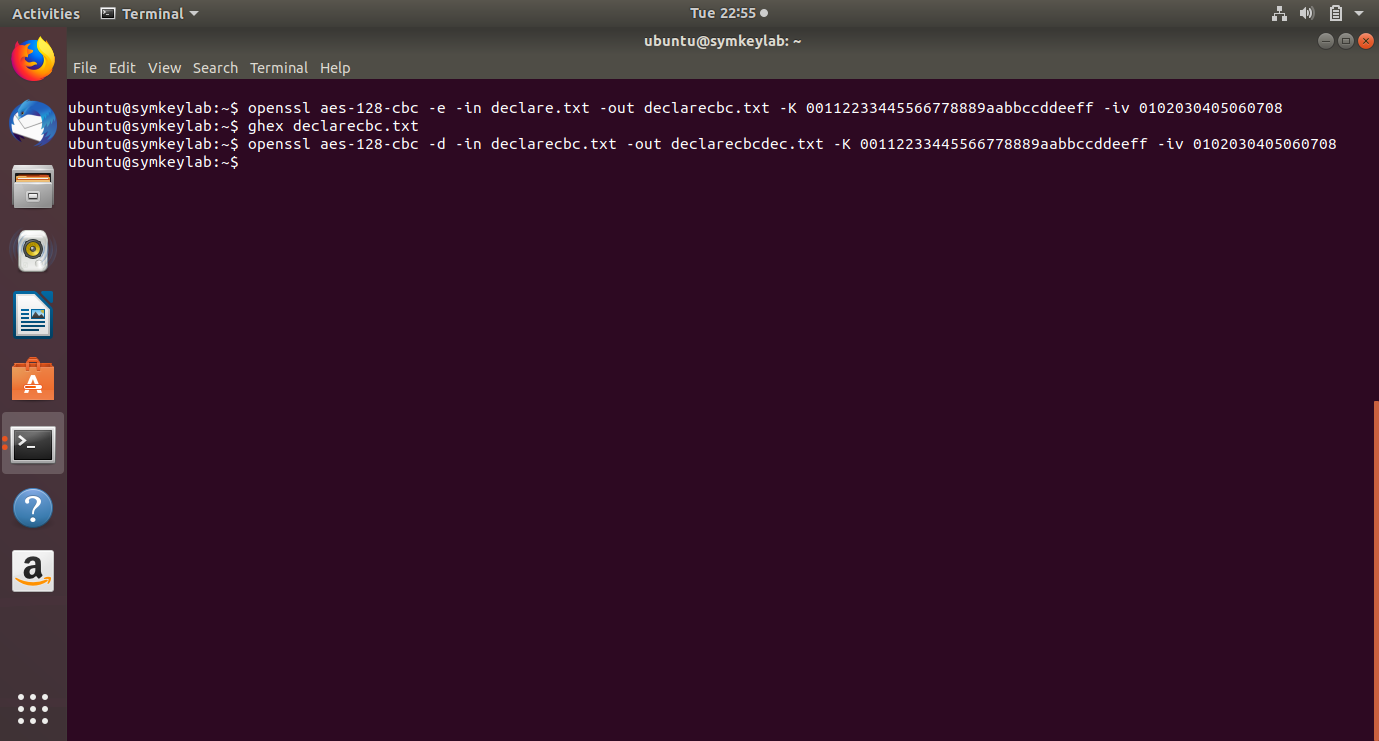
It displays the bit *prior* to changing the bit in the file.



The bit change is done by converting F to E(as highlighted in the image below).



Decrypting the cipher text file (declarecbc.txt) to the new file declarecbcdec.txt.



Using diff command to find the difference between the plain text file(declare.txt) and the new file after decryption(declarecbcdec.txt).



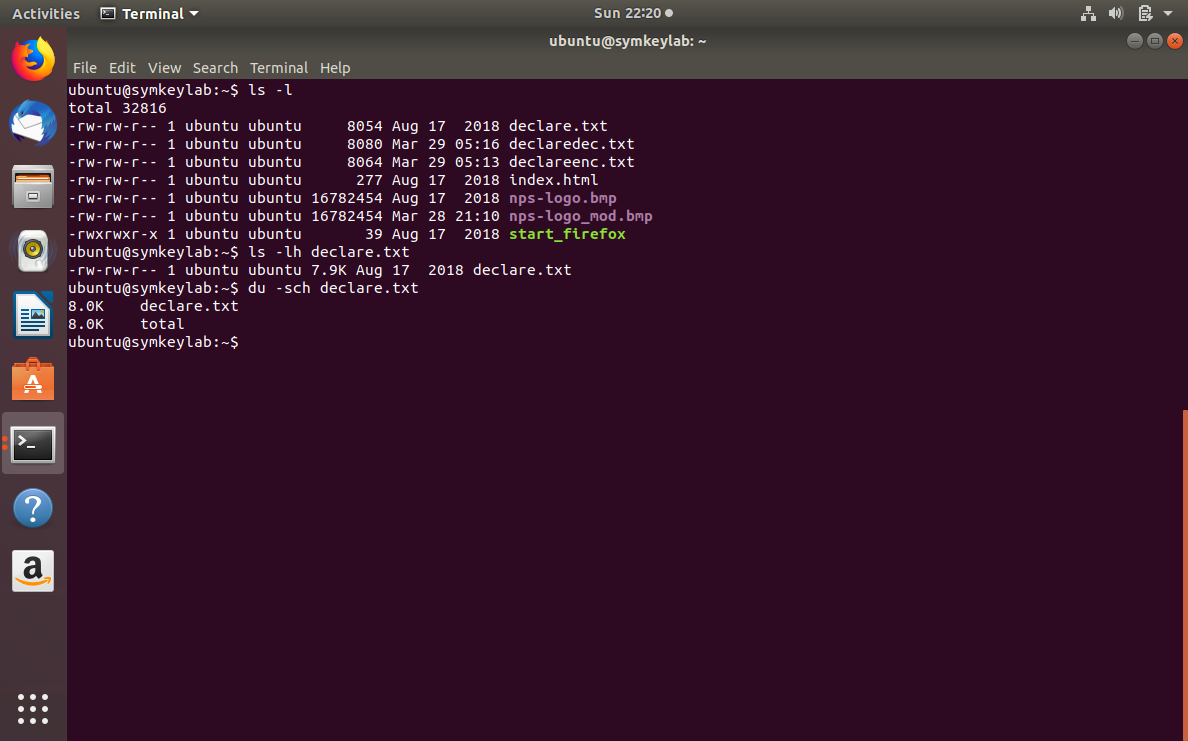
**In Conclusion**

1. **Describe any extra experimentation you performed.**

**Answer:**

In the assignment, we used the following extra commands:

1. du –sch : We used this command to find out the size of the files.
2. Ls –lh : We also explored this command to find out the detailed information about the files.
3. We explored the *enc* command using the command *man enc*.
4. We also explored the *diff* command using the *man diff* to find out the options available in the command.



1. **What did you learn from this exercise (if anything)? Please point out the observations or results that were surprising to you (if any). Also include those concepts that were made clearer (if any) by doing the assignment.**

**Answer:** By doing the assignment, we got a clearer understanding of different modes of operation that is ECB, CBC, CFB and OFB modes. The assignment gives us a hands-on experience on symmetric key encryption, which helped us in understanding the concepts in a more practical manner. We observed the differences in the different modes and the error propagation using these modes. We found the results of the image encryption using different modes quite surprising and interesting. All in all, we got a better understanding of the concepts now.

1. **How could this exercise be improved?**

**Answer:** This exercise is apt for learning symmetric key encryption and different modes of operation.