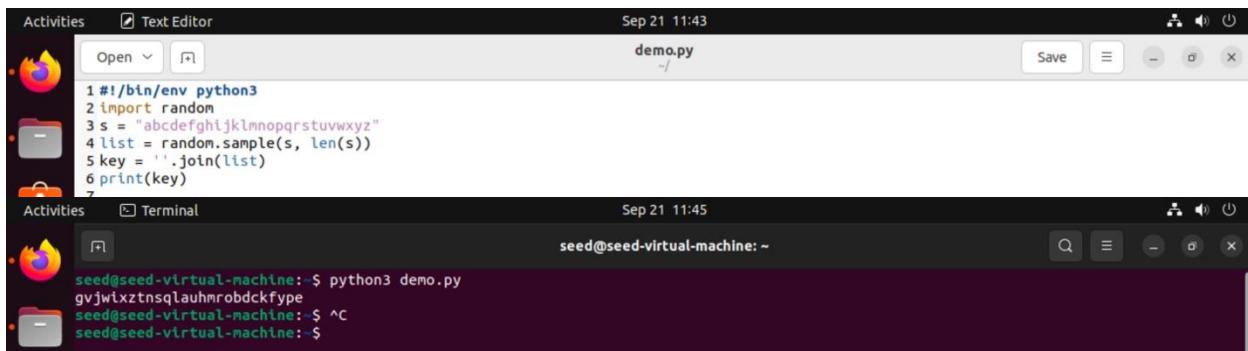


Mini Project 1: Secret-Key Encryption

Project Topic: Experiments on Symmetric-Key Cryptography Principles

Task 1: Frequency Analysis

Step 1: We generate an encryption key; we use the given python program to produce an encryption key. An encryption key is just shuffled alphabet, which is later used to for monoalphabetic substitution.



The screenshot shows a desktop environment with two windows. The top window is a 'Text Editor' titled 'demo.py' containing Python code to generate a shuffled alphabet key. The bottom window is a terminal window titled 'seed@seed-virtual-machine: ~' showing the execution of the script and the resulting shuffled key.

```
#!/bin/env python3
import random
s = "abcdefghijklmnopqrstuvwxyz"
list = random.sample(s, len(s))
key = ''.join(list)
print(key)
```

```
seed@seed-virtual-machine: $ python3 demo.py
gvjwixztnsqlauhmr0bdckfype
seed@seed-virtual-machine: $ ^C
seed@seed-virtual-machine: $
```

Fig 1.1

Fig 1.1 shows the code and the encryption key produced after executing the code.

Step 2: In this step we create a text file called as “article.txt” which contains an example sentence. The purpose of this step is to demonstrate simplification. This is achieved by converting all upper case characters to lower case characters, then removing all punctuations and numbers. In the real monoalphabetic cipher the spaces are removed but to keep the demonstration simple we keep the spaces.

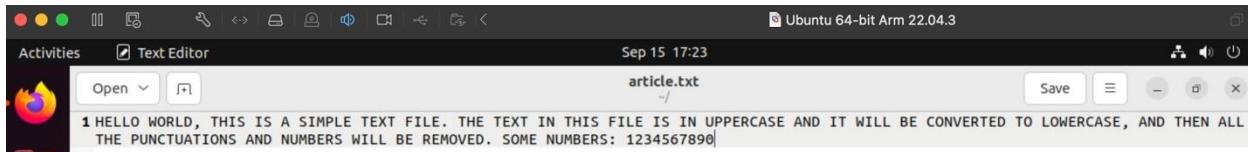
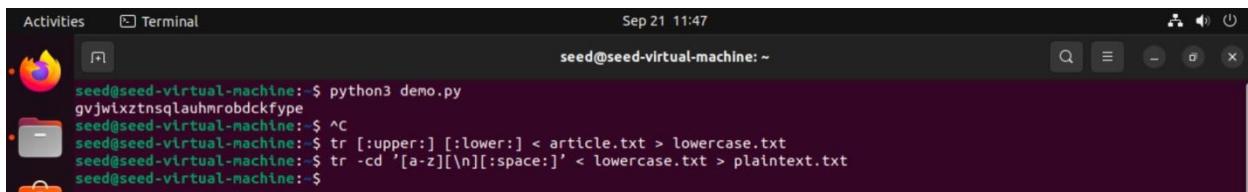


Fig 1.2.1

In Fig 1.2.1, we can see that the text file named “article.txt” has a sentence in all upper case letters for easy demonstration and it also contains number at the end.



The screenshot shows a terminal window with two commands executed. The first command uses 'tr' to convert uppercase to lowercase. The second command uses 'tr -cd' to remove punctuation and numbers, resulting in a plain text file.

```
seed@seed-virtual-machine: $ python3 demo.py
gvjwixztnsqlauhmr0bdckfype
seed@seed-virtual-machine: $ ^C
seed@seed-virtual-machine: $ tr [:upper:] [:lower:] < article.txt > lowercase.txt
seed@seed-virtual-machine: $ tr -cd '[a-z][\n[:space:]]' < lowercase.txt > plaintext.txt
seed@seed-virtual-machine: $
```

Fig 1.2.2

In Fig 1.2.2, we have executed two commands for simplification of the text present in “article.txt” file.

Command 1: `tr [:upper:] [:lower:] < article.txt > lowercase.txt`

With this command, all uppercase letters in the "article.txt" file are changed to their lowercase equivalents, and the transformed content is saved in a new file called "lowercase.txt." Numbers and punctuation are unaffected by this alteration of other characters. This can be seen in the Fig 1.2.3 given below.

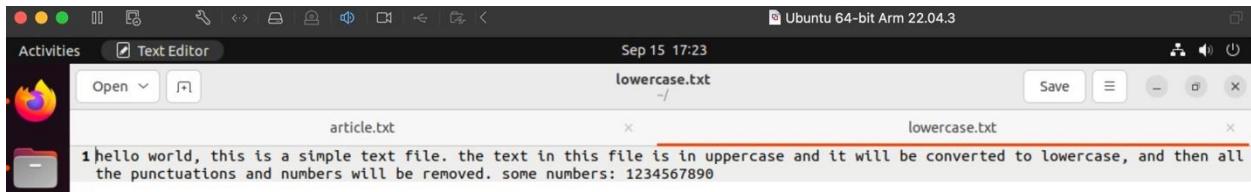


Fig 1.2.3

Command 2: `tr -cd '[a-z][\n][:space:]' < lowercase.txt > plaintext.txt`

This command processes the text from the "lowercase.txt" file. It removes all characters that are not lowercase letters, newline characters, or whitespace characters. The resulting filtered text, which does not contain any number, is then saved in a new file called "plaintext.txt." This can be seen in the Fig 1.2.4 given below.



Fig 1.2.4

Step 3: In this step we encrypt the “plaintext.txt” file.

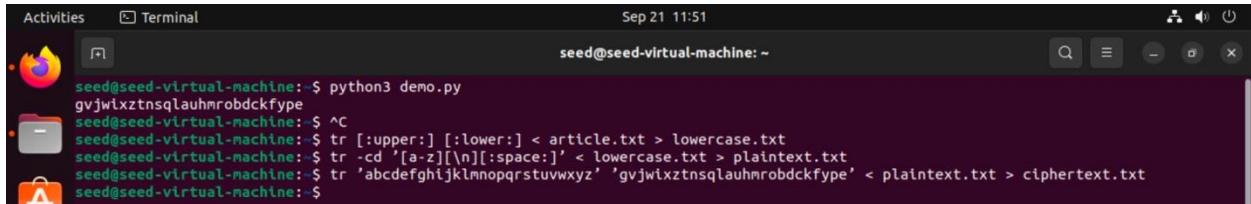


Fig 1.3.1

Fig 1.3.2 shows the command used to perform the encryption.

Command: `tr 'abcdefghijklmnopqrstuvwxyz' 'gvjwixztnsqlauhmrobdckfype' < plaintext.txt > ciphertext.txt`

This is the encryption key we got in the step 1: “`gvjwixztnsqlauhmrobdckfype`”.

This command substitutes characters in the "plaintext.txt" file based on a specified mapping. It replaces each letter from the alphabet 'a' to 'z' with the corresponding letter from the encryption key. The resulting transformed text is then saved in the "ciphertext.txt" file. The Fig 1.3.2 given below show the transformed text in the file “ciphertext.txt”.

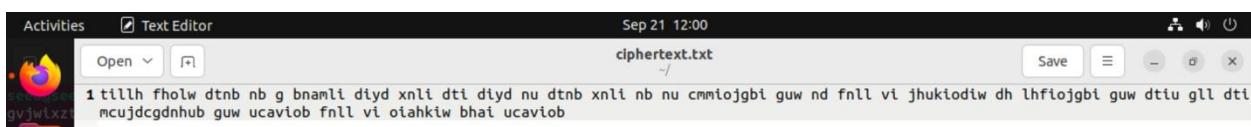


Fig 1.3.2

Now, to finish the Task 1, it is required to use Frequency Analysis to figure out the encryption key and the original plain text. The Python code provided in the file “freq.py” inside the Labsetup/Files folder.

- By running the provided “freq.py” file we generated statistics for n-grams in the ciphertext. This helped us identify common letter sequences in English.
- We started with single-letter frequencies (1-gram) to identify the most common letters in the ciphertext. Match these with their English counterparts which we found here ([single-letter frequency graph](#)) (e.g., if 'n' is the most common letter, assume it corresponds to 'e').
- Then we used the identified substitutions to update the ciphertext using the “tr” command to get more clues about the key and plaintext.
- Then we used the bigram (2-gram) and trigram (3-gram) frequencies to find common letter pairs and triplets. This helped us refine our substitutions. ([Bigram frequencies](#) and [Trigram frequencies](#))
- We continued this process iteratively, making educated guesses based on frequency analysis, updating the ciphertext, and refining our substitutions.
- Eventually, you were able to decode most of the text, which will give us more clues about the key and made it easier to decrypt the remaining portions.
- We also kept track of your substitutions and progress and used capital letters for the plaintext to distinguish it from the ciphertext.

The encryption key is “vgapnbrtmosicuxehqyzflkdw”.

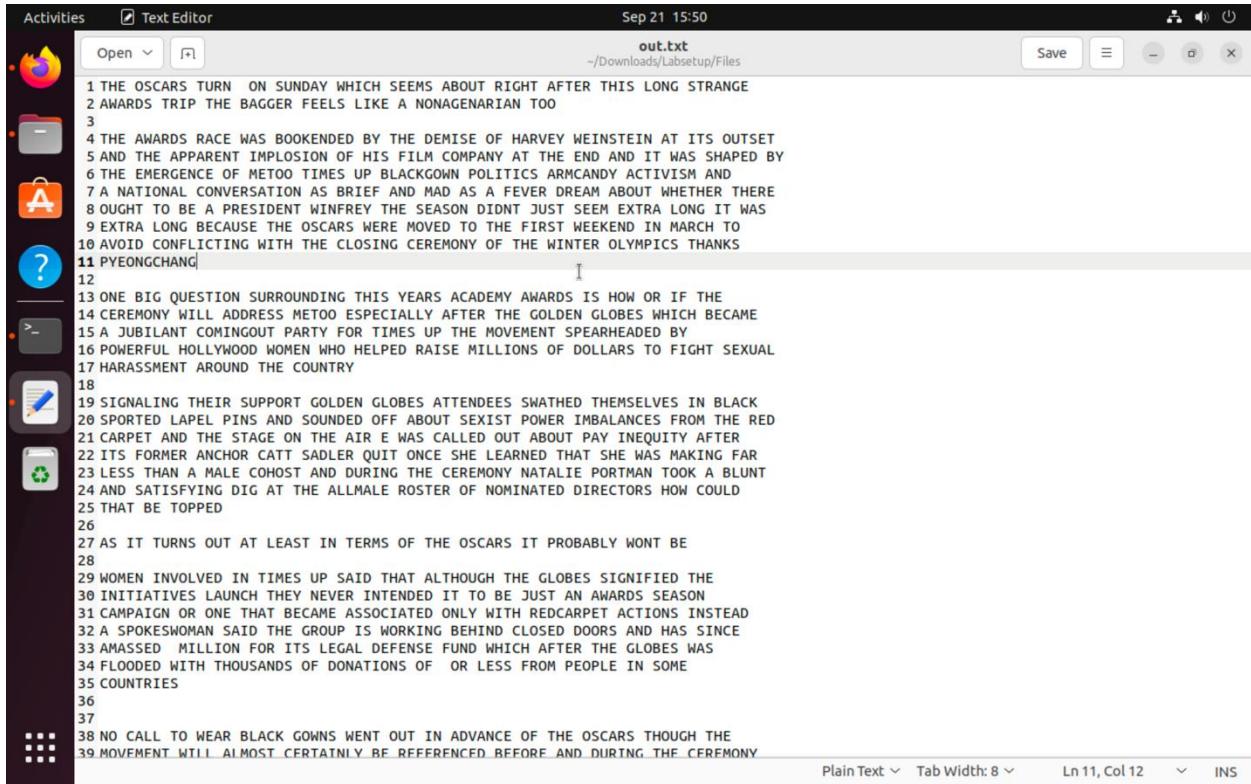
Below are the screenshots of the frequency statistics for 1-gram, 2-gram, and 3-gram along with the commands used to replace the characters and automatically store them in a new file called “out.txt”.

```
Activities Terminal Sep 21 14:46
seed@seed-virtual-machine: ~/Downloads/Labsetup/Files$ python3 freq.py
1-gram (top 20):
n: 488
y: 373
v: 348
x: 291
u: 280
q: 276
m: 264
h: 235
t: 183
i: 166
p: 156
a: 116
c: 104
z: 95
l: 90
g: 83
b: 83
r: 82
e: 76
d: 59
2-gram (top 20):
yt: 115
tn: 89
mu: 74
nh: 58
vh: 57
hn: 57
vu: 56
nq: 53
xu: 52
up: 46
xh: 45
yn: 44
np: 44
vy: 44
nu: 42
seed@seed-virtual-machine: ~/Downloads/Labsetup/Files$ cat > out.txt
```

Activities Terminal Sep 21 14:47 seed@seed-virtual-machine: ~/Downloads/Labsetup/Files

```
yn: 44
np: 44
vy: 44
nu: 42
qy: 39
vq: 33
vi: 32
gn: 32
av: 31
-----
3-gram (top 20):
ytn: 78
vup: 30
mur: 20
ynh: 18
xzy: 16
mxu: 14
gnq: 14
ytv: 13
nqy: 13
vii: 13
bxh: 13
lvq: 12
nuy: 12
vyn: 12
uvy: 11
lmu: 11
nvh: 11
cmu: 11
tmq: 10
vhp: 10
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'n' 'E' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'ny' 'ET' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyv' 'ETA' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvx' 'ETAO' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxu' 'ETAON' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuq' 'ETAONS' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqm' 'ETAONSI' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmh' 'ETAONSIR' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmht' 'ETAONSIRH' < ciphertext.txt > out.txt
Activities Terminal Sep 21 14:47 seed@seed-virtual-machine: ~/Downloads/Labsetup/Files
nuy: 12
vyn: 12
uvy: 11
lmu: 11
nvh: 11
cmu: 11
tmq: 10
vhp: 10
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'n' 'E' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'ny' 'ET' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyv' 'ETA' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvx' 'ETAO' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxu' 'ETAON' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuq' 'ETAONS' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqm' 'ETAONSI' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmh' 'ETAONSIR' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmht' 'ETAONSIRH' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtip' 'ETAONSIRHL' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDC' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCM' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMU' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUM' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMB' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMBFG' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMBFGP' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMBFGPV' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMBFGPVV' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMBFGPYVK' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMBFGPYVKQ' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMBFGPYVKQX' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMBFGPYVKQJ' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tr 'nyvxuqmhtipac' 'ETAONSIRHLDCMUMBFGPYVKQJZ' < ciphertext.txt > out.txt
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$
```

The screenshot below shows the original plain text that is stored on a text file called “out.txt”. Every time the “tr” command is executed the out.txt file gets updated with the substituted alphabet.



A screenshot of a Mac OS X Text Editor window. The title bar says "Text Editor" and "out.txt". The status bar shows "Sep 21 15:50" and the file path "~/Downloads/LabSetup/Files". The window contains a large block of text with many numbered lines (1 through 38) describing the Oscars and the #OscarsSoWhite controversy. The text discusses Harvey Weinstein, the awards race, and the BlackGown movement. It also mentions the Pyeongchang Winter Olympics and the #MeToo movement. The text is in a monospaced font and is mostly in uppercase. The window has a dark theme with light-colored text. The sidebar on the left shows various application icons like Finder, Mail, and Safari.

```
1 THE OSCARS TURN ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG STRANGE
2 AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO
3
4 THE AWARDS RACE WAS BOOKENDED BY THE DEMISE OF HARVEY WEINSTEIN AT ITS OUTSET
5 AND THE APPARENT IMPLOSION OF HIS FILM COMPANY AT THE END AND IT WAS SHAPED BY
6 THE EMERGENCE OF METOO TIMES UP BLACKGOWN POLITICS ARMCANDY ACTIVISM AND
7 A NATIONAL CONVERSATION AS BRIEF AND MAD AS A FEVER DREAM ABOUT WHETHER THERE
8 OUGHT TO BE A PRESIDENT WINFREY THE SEASON DIDNT JUST SEEM EXTRA LONG IT WAS
9 EXTRA LONG BECAUSE THE OSCARS WERE MOVED TO THE FIRST WEEKEND IN MARCH TO
10 AVOID CONFLICTING WITH THE CLOSING CEREMONY OF THE WINTER OLYMPICS THANKS
11 PYEONGCHANG|
12
13 ONE BIG QUESTION SURROUNDING THIS YEARS ACADEMY AWARDS IS HOW OR IF THE
14 CEREMONY WILL ADDRESS METOO ESPECIALLY AFTER THE GOLDEN GLOBES WHICH BECAME
15 A JUBILANT COMINGOUT PARTY FOR TIMES UP THE MOVEMENT SPEARHEADED BY
16 POWERFUL HOLLYWOOD WOMEN WHO HELPED RAISE MILLIONS OF DOLLARS TO FIGHT SEXUAL
17 HARASSMENT AROUND THE COUNTRY
18
19 SIGNALING THEIR SUPPORT GOLDEN GLOBES ATTENDEES SWATHED THEMSELVES IN BLACK
20 SPORDED LAPEL PINS AND SOUNDED OFF ABOUT SEXIST POWER IMBALANCES FROM THE RED
21 CARPET AND THE STAGE ON THE AIR E WAS CALLED OUT ABOUT PAY INEQUITY AFTER
22 ITS FORMER ANCHOR CATT SADLER QUIT ONCE SHE LEARNED THAT SHE WAS MAKING FAR
23 LESS THAN A MALE COHOST AND DURING THE CEREMONY NATALIE PORTMAN TOOK A BLUNT
24 AND SATISFYING DIG AT THE ALLMALE ROSTER OF NOMINATED DIRECTORS HOW COULD
25 THAT BE TOPPED
26
27 AS IT TURNS OUT AT LEAST IN TERMS OF THE OSCARS IT PROBABLY WONT BE
28
29 WOMEN INVOLVED IN TIMES UP SAID THAT ALTHOUGH THE GLOBES SIGNIFIED THE
30 INITIATIVES LAUNCH THEY NEVER INTENDED IT TO BE JUST AN AWARDS SEASON
31 CAMPAIGN OR ONE THAT BECAME ASSOCIATED ONLY WITH REDCARPET ACTIONS INSTEAD
32 A SPOKESWOMAN SAID THE GROUP IS WORKING BEHIND CLOSED DOORS AND HAS SINCE
33 AMASSED MILLION FOR ITS LEGAL DEFENSE FUND WHICH AFTER THE GLOBES WAS
34 FLOODED WITH THOUSANDS OF DONATIONS OF OR LESS FROM PEOPLE IN SOME
35 COUNTRIES
36
37
38 NO CALL TO WEAR BLACK GOWNS WENT OUT IN ADVANCE OF THE OSCARS THOUGH THE
39 MOVEMENT WILL ALMOST CERTAINLY BE REFERENCED BEFORE AND DURING THE CEREMONY
```

To conclude, we can say that this task highlights the effectiveness of frequency analysis and systematic decryption techniques in deciphering monoalphabetic substitution ciphers, showcasing the importance of pattern recognition and iterative decryption in the process.

Task 2: Encryption using Different Ciphers and Modes

- 1. AES (aes-192-ctr)
 - In this example, we use Advanced Encryption Standard (AES) which uses a 192-bit key for encryption.
 - When used, CTR mode converts a block cipher into a stream cipher by encrypting a counter and XORing the result with the plaintext. This style is renowned for being parallelizable and efficient.

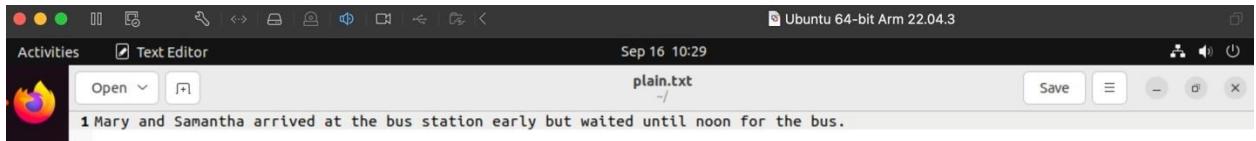


Fig: The plain text.

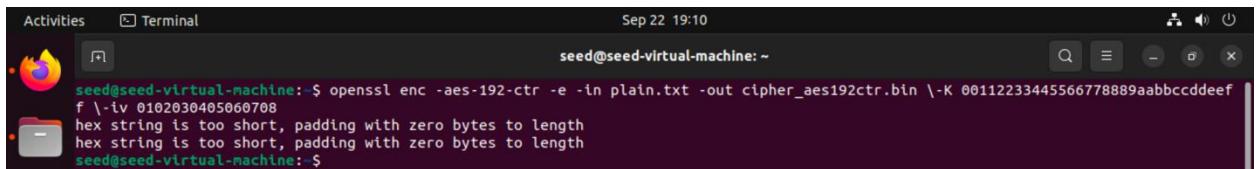


Fig: The plain text encrypted using AES.



Fig: The encrypted text.

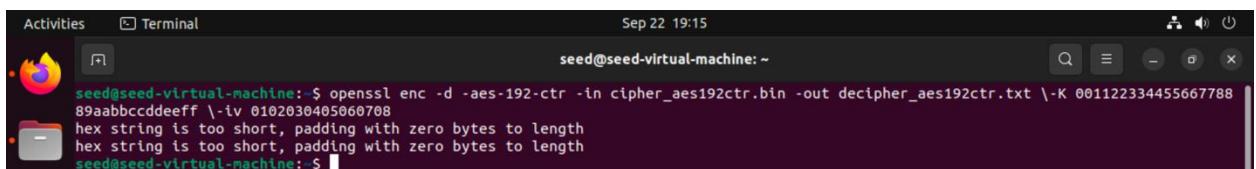


Fig: The decryption command.

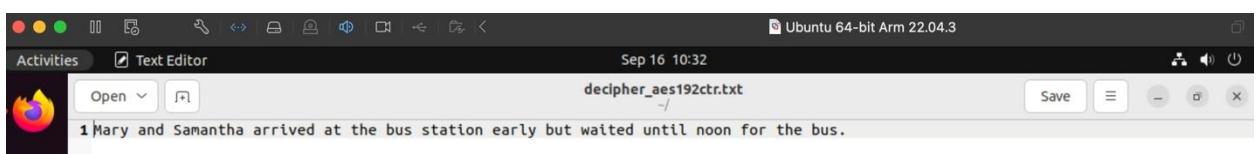


Fig: The original text after decryption.

2. ARIA (aria-256-cfb)

- In this example, we use ARIA, a South Korean encryption algorithm. It uses a 256-bit key for encryption.
 - The CFB mode of operation enables the encryption of single bytes as opposed to whole blocks. In this mode, the following plaintext byte is encrypted using the preceding ciphertext block.

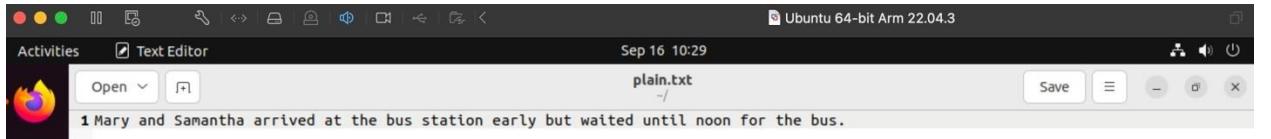


Fig: The plain text.

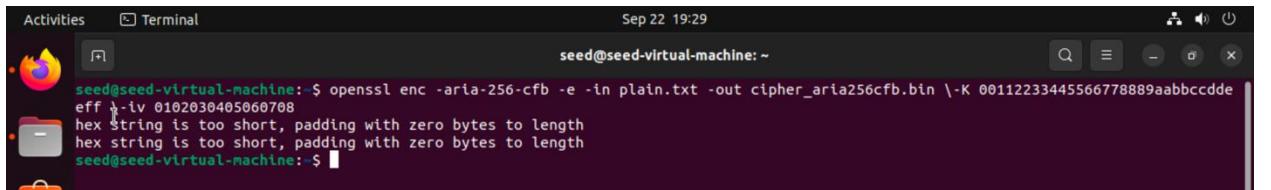


Fig: The plain text encrypted using ARIA.



Fig: The encrypted text.

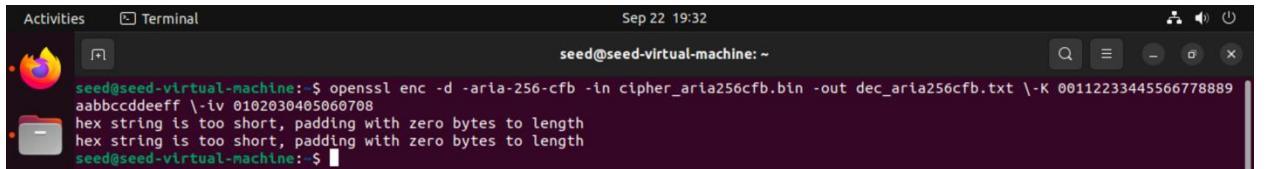


Fig: The decryption command.

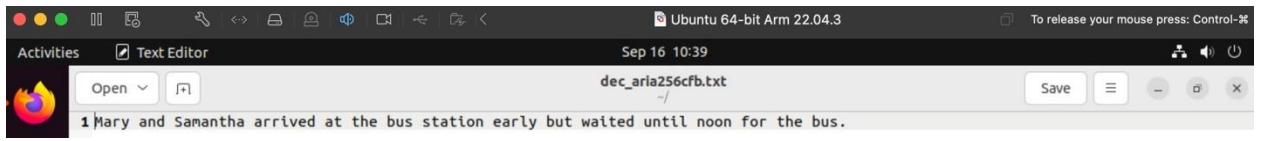


Fig: The original text after decryption.

3. CAMELLIA (camellia-128-cbc)

- In this example, we use Camellia, which is a symmetric key block cipher and uses a 128-bit key for encryption.
 - Each block of plaintext in CBC mode is encrypted after being XORed with the block of ciphertext that came before it. It offers discretion and is frequently used for secure conversations.

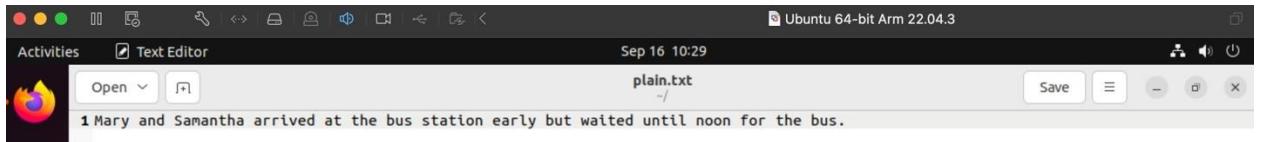


Fig: The plain text.

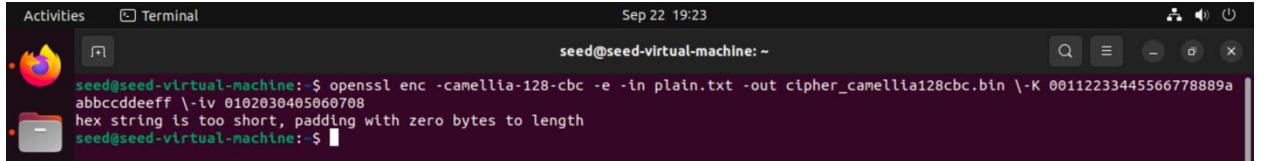


Fig: The plain text encrypted using CAMELLIA.



Fig: The encrypted text.

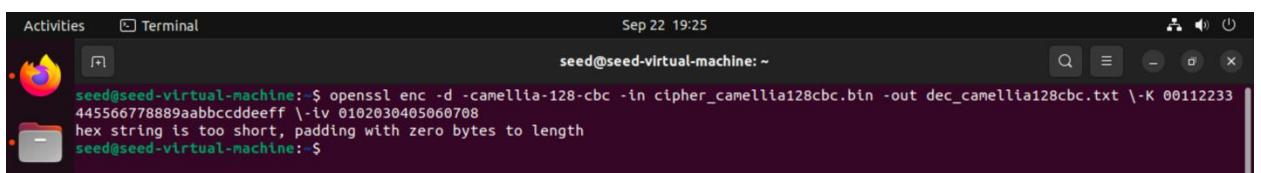


Fig: The decryption command.

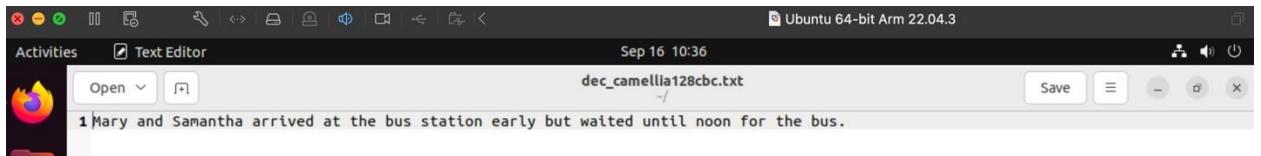


Fig: The original text after decryption.

Observation:

In all of these experiments the following text came up "hex key too short, padding with zero bytes to length". This indicates that the key specified for encryption is shorter than what is expected by the encryption algorithm for example, AES-192 requires a 192-bit key, which is 24 bytes, and AES-256 requires a 256-bit key, which is 32 bytes. When this happens, the encryption tool (in this case, OpenSSL) will automatically pad the provided key with zero bytes to meet the required key length for the chosen encryption algorithm.

Since both the key and IV are supplied in the command, OpenSSL doesn't need to interactively ask the user for an encryption password. This can be useful in scenarios where you want to automate encryption processes and don't want to rely on manual input.

Task 3: Encryption Mode – ECB vs. CBC

Encrypting an image file using different encryption modes (ECB and CBC) and then analyzing the results is an interesting experiment to understand the differences between these modes and their impact on image encryption. Below are the steps to perform this experiment using the provided image file (pic_original.bmp) and some additional observations with a picture of my choice.

Step 1: Encrypt the Image



Fig: pic_original.bmp

This is the original pic that we will encrypt now using the following commands.

Encrypt using ECB mode:

```
$ openssl enc -aes-256-ecb -e -in pic_original.bmp -out enc_ecb.bmp
```

Encrypt using CBC mode:

```
$ openssl enc -aes-256-cbc -e -in pic_original.bmp -out enc_cbc.bmp
```

This will create two encrypted image files: enc_ecb.bmp and enc_cbc.bmp.

To make the encrypted files appear as legitimate BMP images, we need to replace their headers with the headers from the original image. We can use the commands you provided:

Extract header from pic_original.bmp.

```
$ head -c 54 pic_original.bmp > header
```

Extract data from encrypted_ecb.bmp (skip the first 54 bytes).

```
$ tail -c +55 enc_ecb.bmp > body_ecb
```

Extract data from encrypted_cbc.bmp (skip the first 54 bytes).

```
$ tail -c +55 enc_cbc.bmp > body_cbc
```

Combine headers and bodies to create new image.

```
$ cat header body_ecb > new_ecb.bmp
```

```
$ cat header body_cbc > new_cbc.bmp
```

Step 3: Display the Encrypted Images

Now, let's view the encrypted images using an image viewer like "eog"

```
$ eog new_ecb.bmp  
$ eog new_cbc.bmp
```

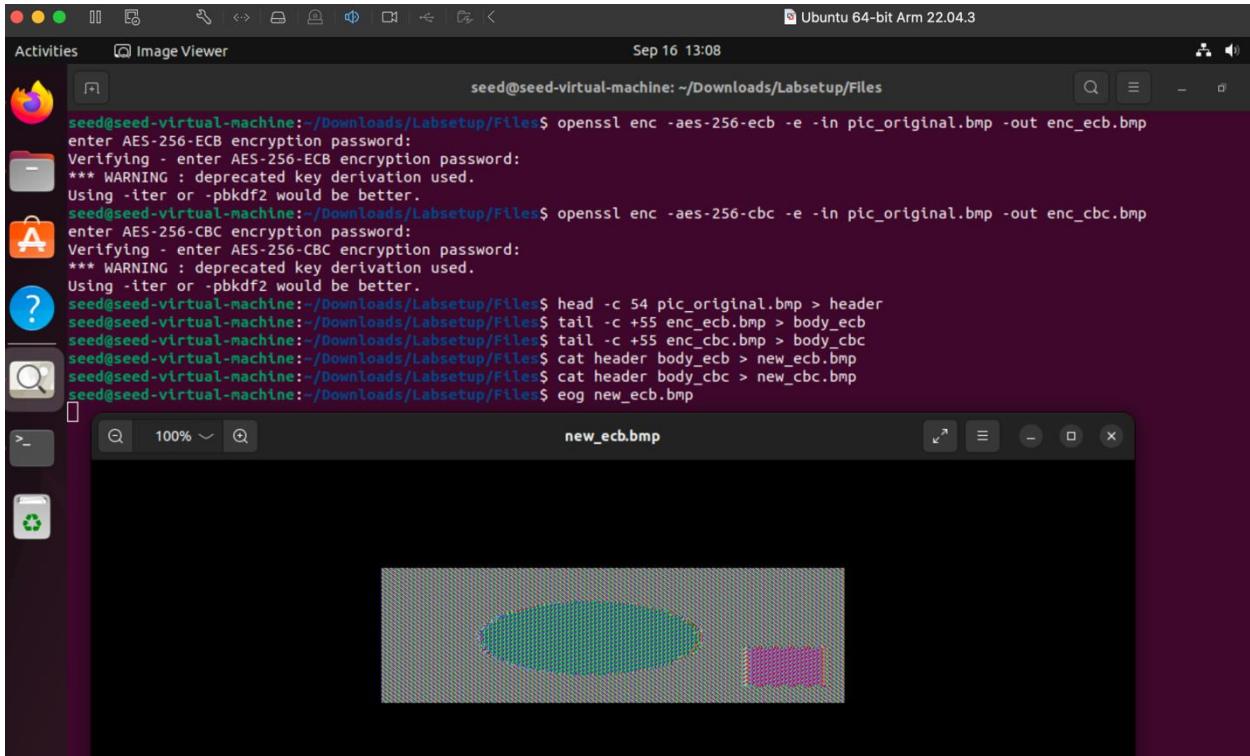
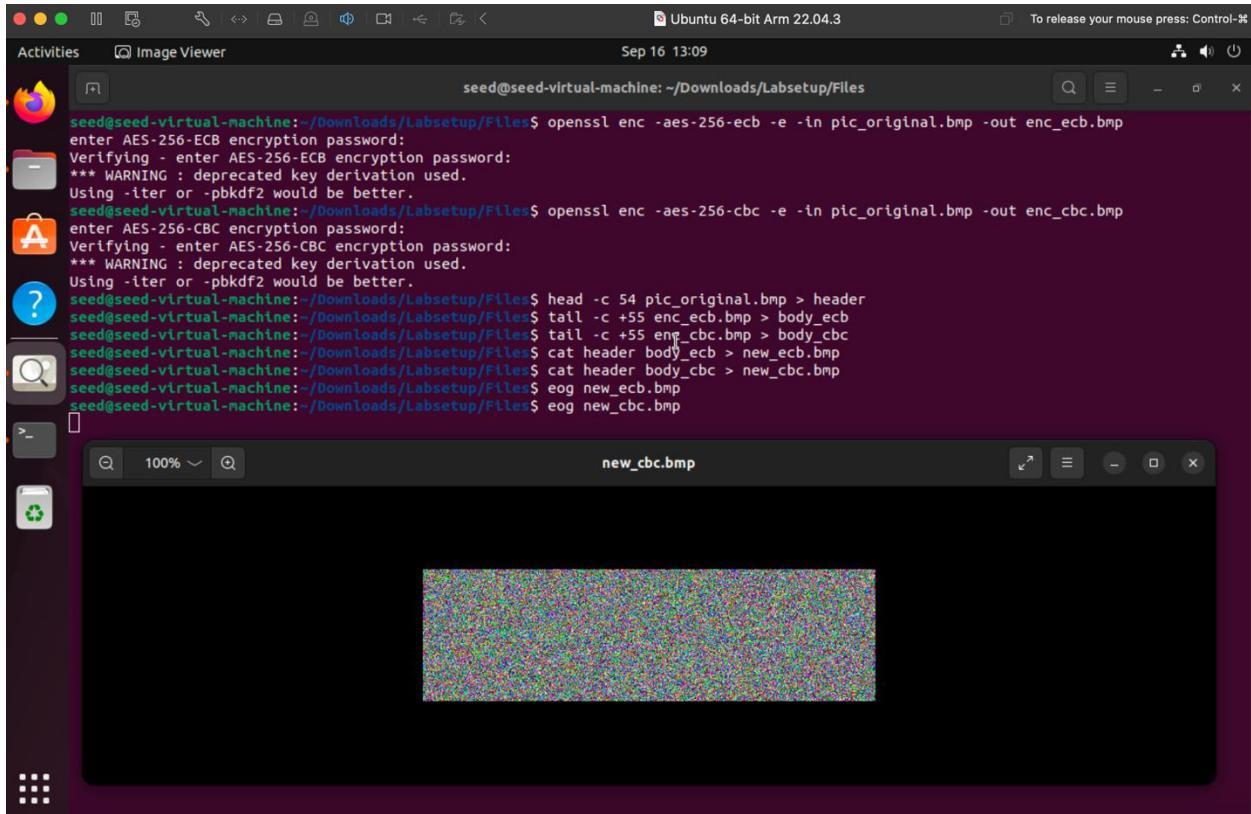


Fig: Encrypted using ECB mode



A screenshot of a Linux desktop environment. At the top, there's a dock with icons for a browser, file manager, terminal, and other applications. The terminal window shows a command-line session where an image is encrypted using AES-256-ECB and AES-256-CBC modes, and then the headers are replaced. The image viewer window shows a highly noisy, multi-colored image labeled "new_cbc.bmp".

```
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ openssl enc -aes-256-ecb -e -in pic_original.bmp -out enc_ecb.bmp
enter AES-256-ECB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ openssl enc -aes-256-cbc -e -in pic_original.bmp -out enc_cbc.bmp
enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ head -c 54 pic_original.bmp > header
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tail -c +55 enc_ecb.bmp > body_ecb
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ tail -c +55 enc_cbc.bmp > body_cbc
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ cat header body_ecb > new_ecb.bmp
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ cat header body_cbc > new_cbc.bmp
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ eog new_ecb.bmp
seed@seed-virtual-machine:~/Downloads/Labsetup/Files$ eog new_cbc.bmp
```

Fig: Encrypted using CBC mode

Observations:

ECB Mode: There is probably some repetition and visual organization in the encrypted picture when you examine “new_ecb.bmp”. The ECB mode separately encrypts each block of data, which might result in patterns that are easy to see, especially in pictures with distinctive patterns or colors. You could see sections or blocks that resemble different areas of the original image, depending on the content of the image.

CBC Mode: If you look at “new_cbc.bmp”, you'll see that the encrypted picture looks less organized and lacks the distinct patterns that you may observe in ECB mode. This is so that patterns may be hidden, and greater security can be provided, as CBC mode encrypts the current block using feedback from earlier blocks.

Additional Experiment with a Different Image:

For an additional experiment, we repeated the same process with a different image. Encrypt the image using both ECB and CBC modes, replace the headers, and observe the differences. The new image in the same .bmp format is shown below.



Fig: sample_2.bmp

The screenshot below shows the same set of commands being executed for the sample_2.bmp image file.

The screenshot shows a terminal window on an Ubuntu 64-bit Arm 22.04.3 system. The terminal output is as follows:

```
seed@seed-virtual-machine:~/Downloads$ openssl enc -aes-256-ecb -e -in sample_2.bmp -out enc_sample_ecb.bmp
enter AES-256-ECB encryption password:
Verifying - enter AES-256-ECB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/Downloads$ openssl enc -aes-256-cbc -e -in sample_2.bmp -out enc_sample_cbc.bmp
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/Downloads$ head -c 54 sample_2.bmp > header
seed@seed-virtual-machine:~/Downloads$ tail -c +55 enc_sample_ecb.bmp > body_sample_ecb
seed@seed-virtual-machine:~/Downloads$ tail -c +55 enc_sample_cbc.bmp > body_sample_cbc
seed@seed-virtual-machine:~/Downloads$ cat header body_sample_ecb > new_sample_ecb.bmp
seed@seed-virtual-machine:~/Downloads$ cat header body_sample_cbc > new_sample_cbc.bmp
seed@seed-virtual-machine:~/Downloads$ eog new_sample_ecb.bmp
seed@seed-virtual-machine:~/Downloads$ eog new_sample_cbc.bmp
seed@seed-virtual-machine:~/Downloads$
```

This is the “new_sample_ecb.bmp” file which is the which is encrypted using ECB mode.

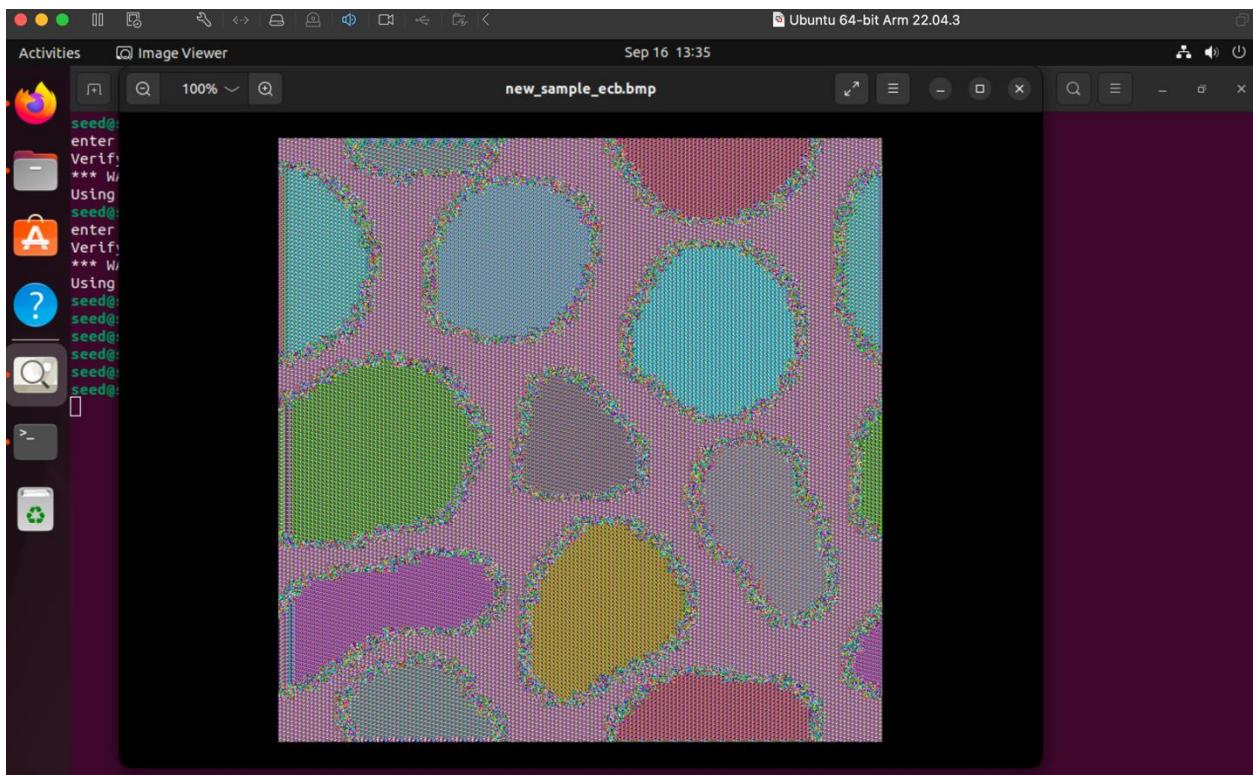


Fig: New image encrypted using ECB mode

This is the “new_sample_cbc.bmp” file which is encrypted using CBC mode.

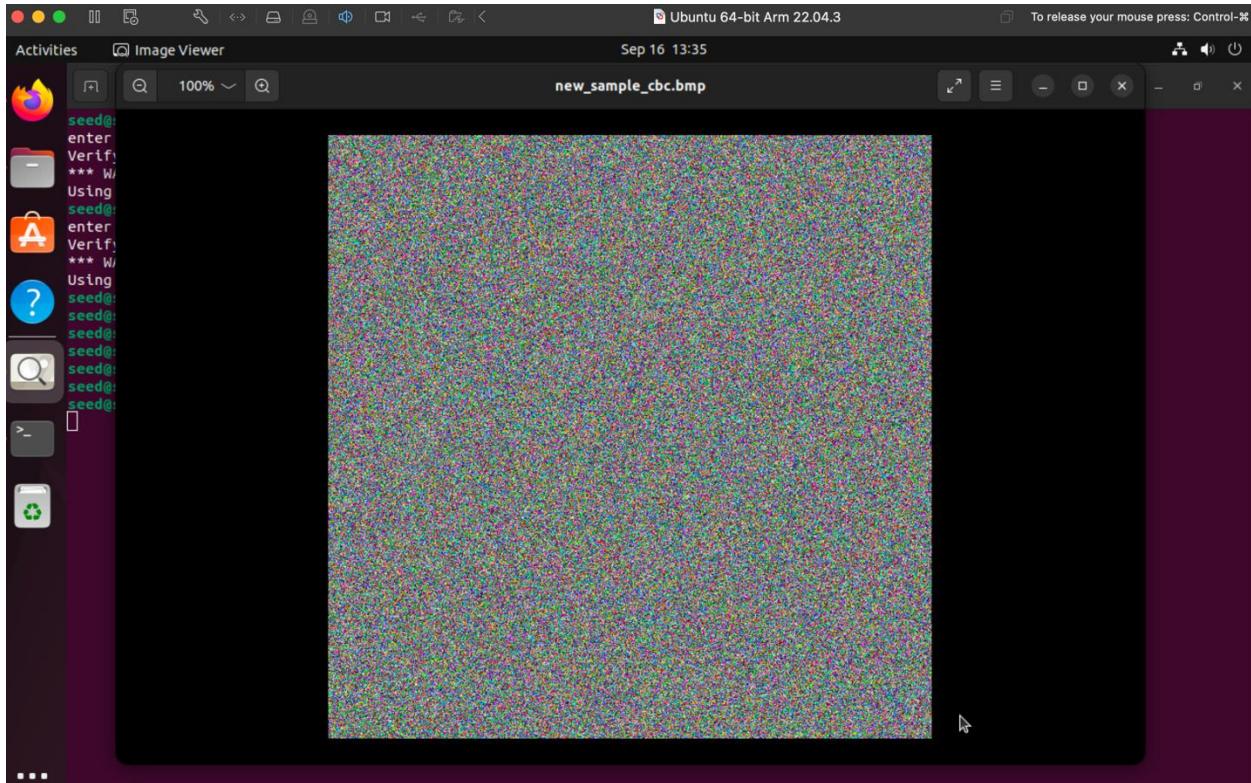


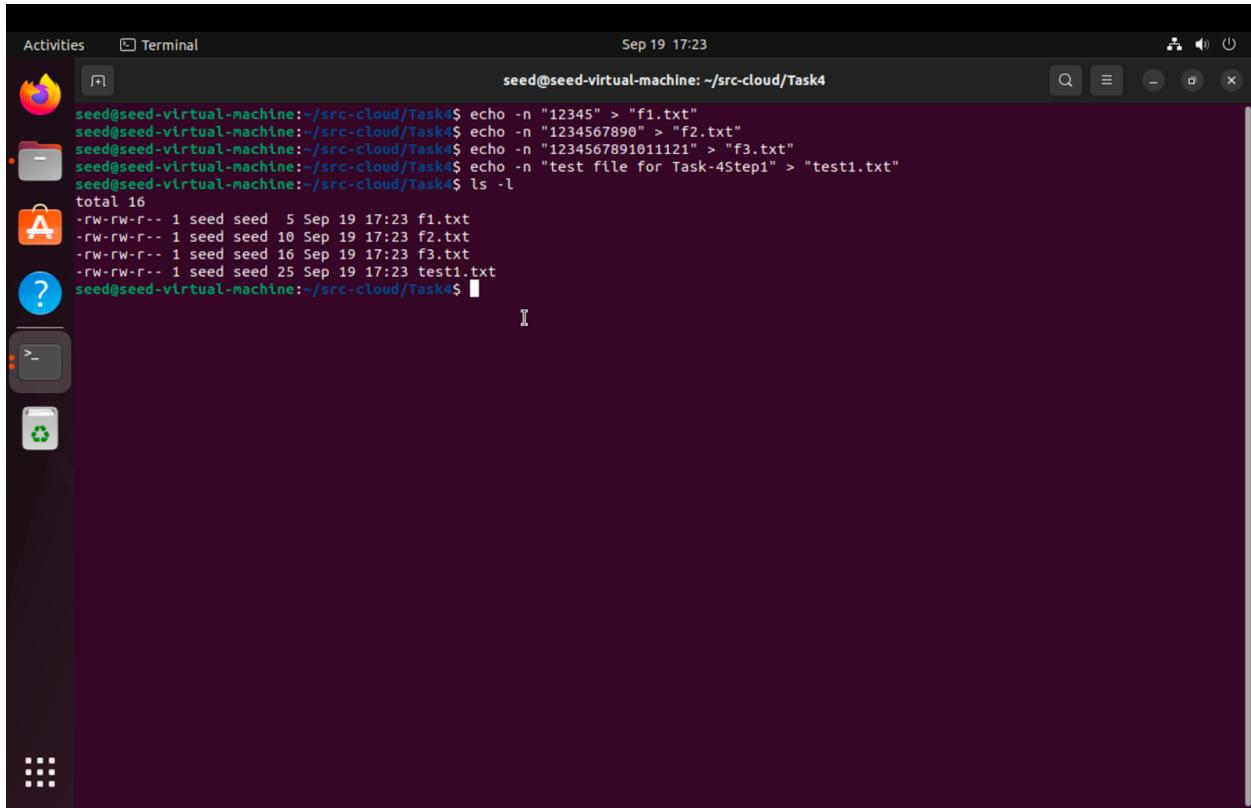
Fig: New image encrypted using CBC mode

Observation:

We found that the same observations hold true: ECB mode reveals patterns in the encrypted image, while CBC mode produces a more randomized result. This experiment helps highlight the importance of using appropriate encryption modes for different types of data and the impact it has on the security of encrypted content.

Task 4: Padding

Making the necessary files



A screenshot of a Linux desktop environment showing a terminal window. The terminal title is "seed@seed-virtual-machine: ~/src-cloud/Task4". The date and time at the top right are "Sep 19 17:23". The terminal window contains the following command-line session:

```
seed@seed-virtual-machine:~/src-cloud/Task4$ echo -n "12345" > "f1.txt"
seed@seed-virtual-machine:~/src-cloud/Task4$ echo -n "1234567890" > "f2.txt"
seed@seed-virtual-machine:~/src-cloud/Task4$ echo -n "1234567891011121" > "f3.txt"
seed@seed-virtual-machine:~/src-cloud/Task4$ echo -n "test file for Task-4Step1" > "test1.txt"
seed@seed-virtual-machine:~/src-cloud/Task4$ ls -l
total 16
-rw-rw-r-- 1 seed seed 5 Sep 19 17:23 f1.txt
-rw-rw-r-- 1 seed seed 10 Sep 19 17:23 f2.txt
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f3.txt
-rw-rw-r-- 1 seed seed 25 Sep 19 17:23 test1.txt
seed@seed-virtual-machine:~/src-cloud/Task4$
```

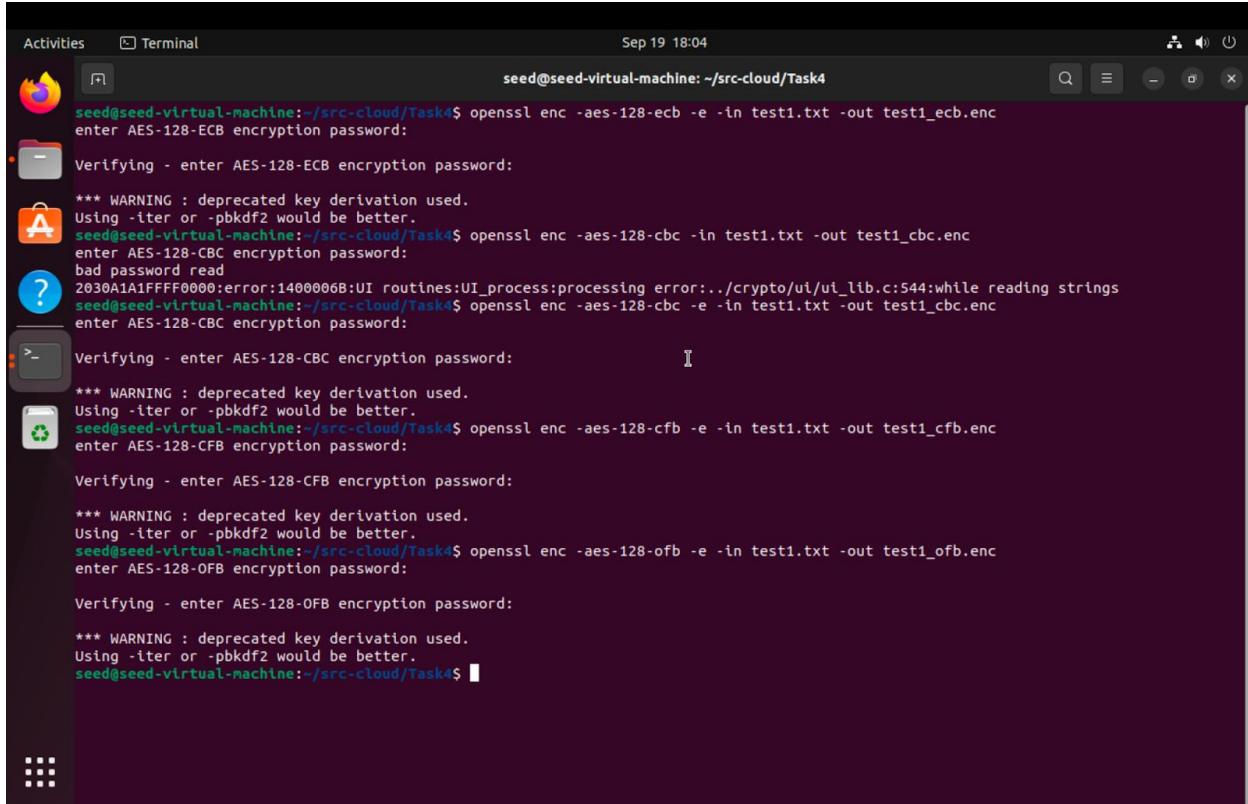
Here we can see that we have made 4 files f1.txt, f2.txt, f3.txt for the 2) step in the task which are 5,10,16 bytes respectively and test1.txt which is our own file of size 25 bytes

1)

We have to use ECB, CBC, CFB and OFB to encrypt the file test1.txt

Done as follows:

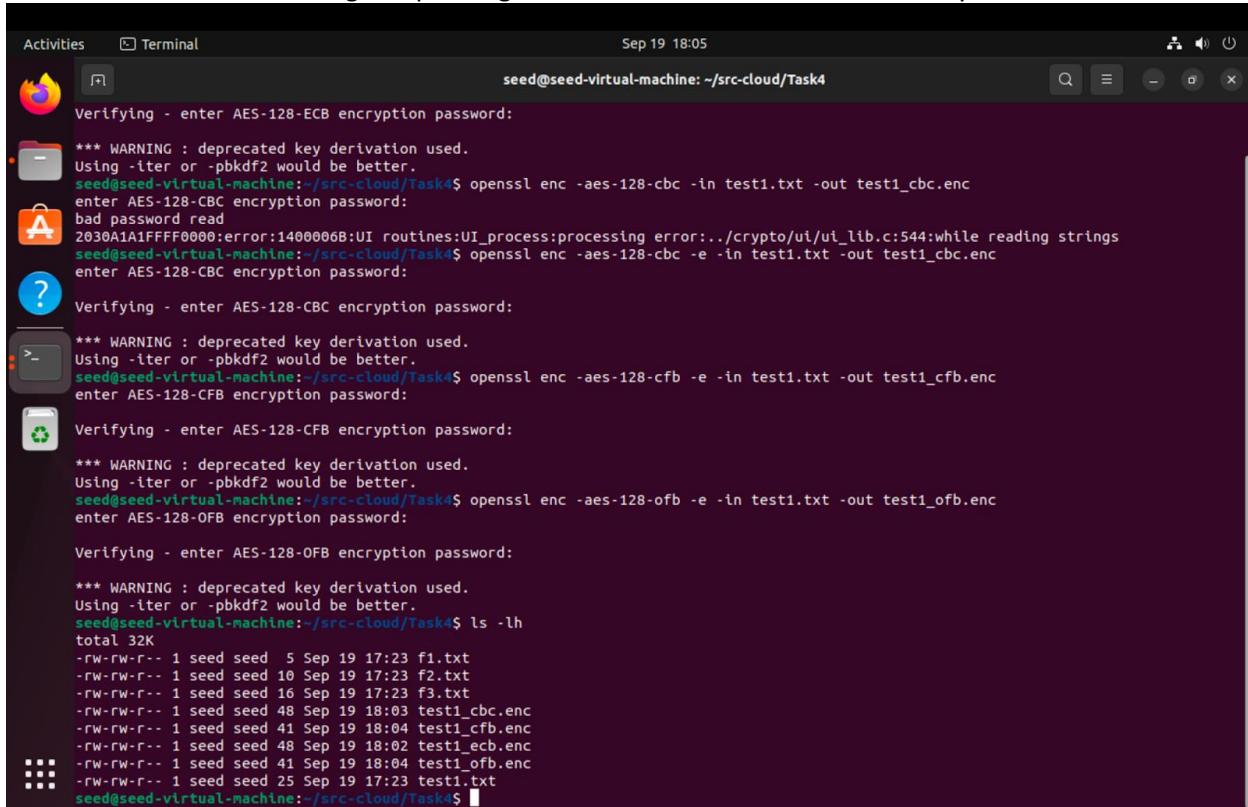
Password: 2537167



Activities Terminal Sep 19 18:04

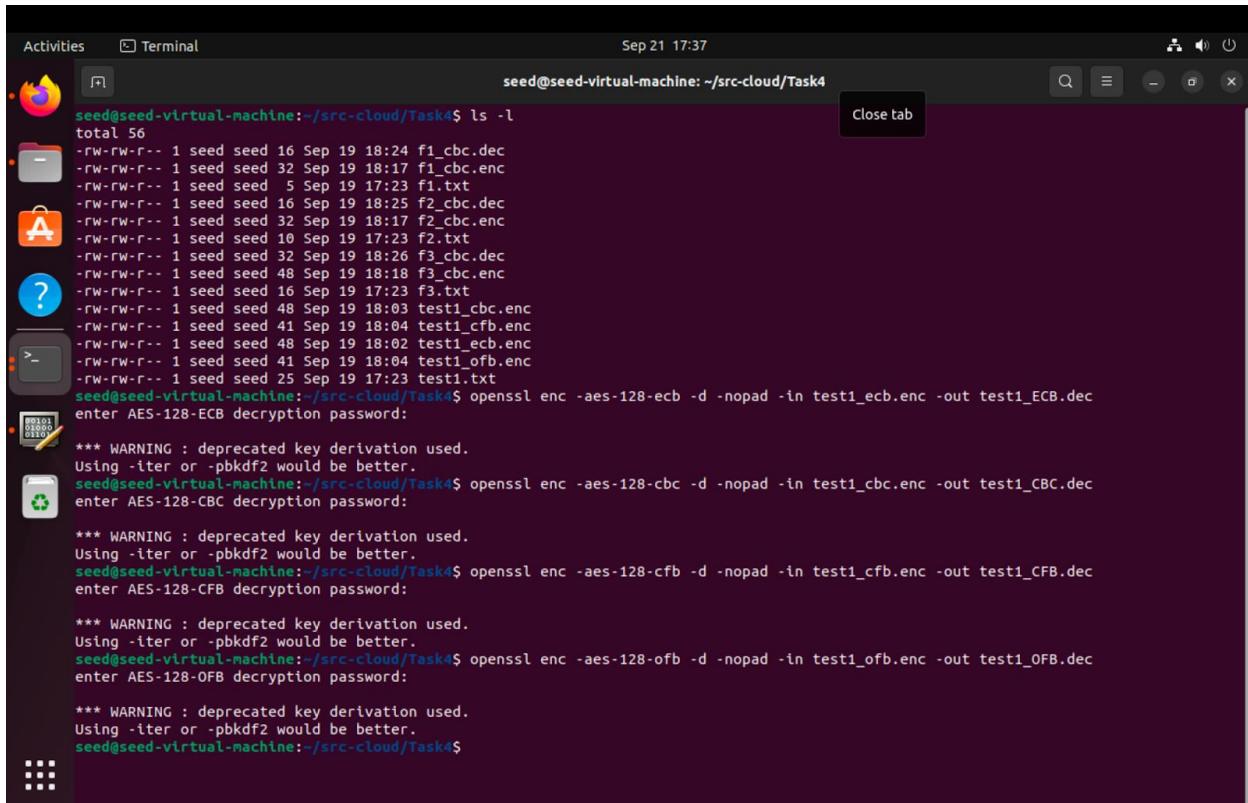
```
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-ecb -e -in test1.txt -out test1_ecb.enc
enter AES-128-ECB encryption password:
Verifying - enter AES-128-ECB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-cbc -in test1.txt -out test1_cbc.enc
enter AES-128-CBC encryption password:
bad password read
2030A1A1FFFF0000:error:1400006B:UI routines:UI_process:processing error:../crypto/ui/ui_lib.c:544:while reading strings
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-cbc -e -in test1.txt -out test1_cbc.enc
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-cfb -e -in test1.txt -out test1_cfb.enc
enter AES-128-CFB encryption password:
Verifying - enter AES-128-CFB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-ofb -e -in test1.txt -out test1_ofb.enc
enter AES-128-OFB encryption password:
Verifying - enter AES-128-OFB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$
```

To check which modes have given padding we use ls -l to check the number of bytes

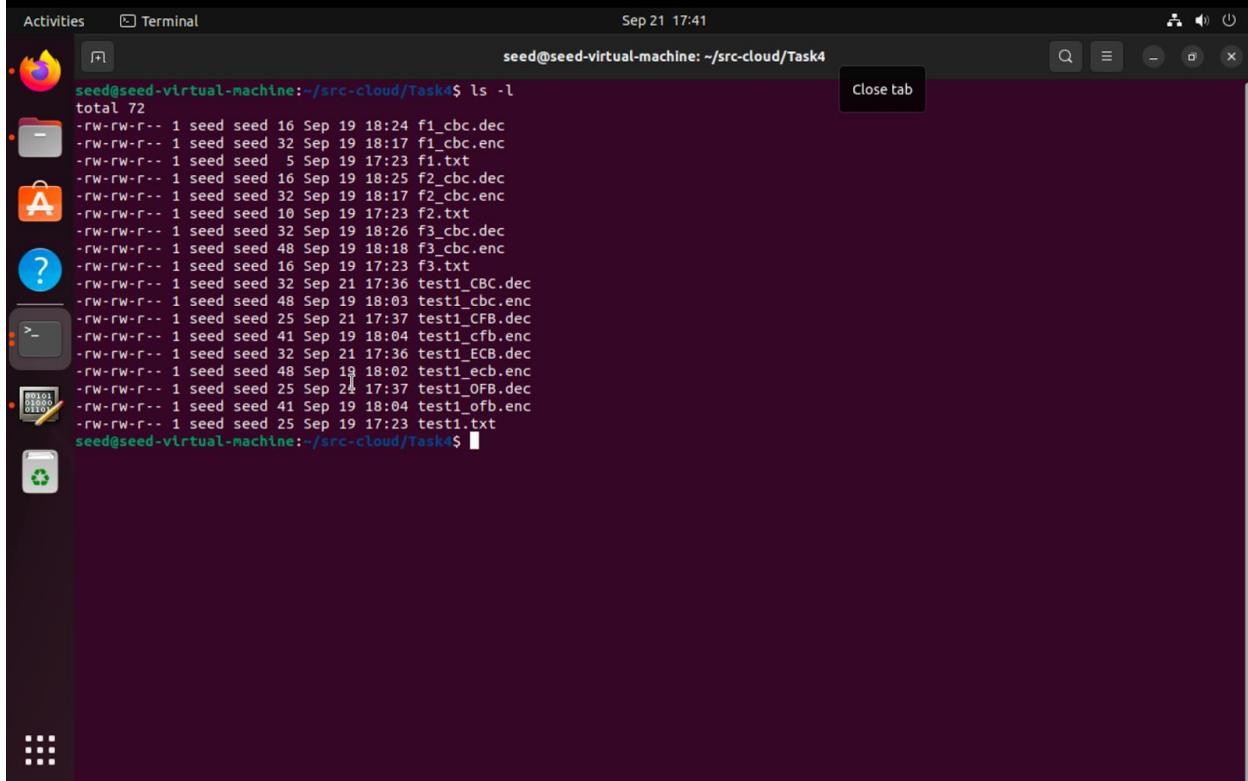


Activities Terminal Sep 19 18:05

```
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-cbc -in test1.txt -out test1_cbc.enc
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-cbc -e -in test1.txt -out test1_cbc.enc
enter AES-128-CBC encryption password:
bad password read
2030A1A1FFFF0000:error:1400006B:UI routines:UI_process:processing error:../crypto/ui/ui_lib.c:544:while reading strings
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-cbc -e -in test1.txt -out test1_cbc.enc
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-cfb -e -in test1.txt -out test1_cfb.enc
enter AES-128-CFB encryption password:
Verifying - enter AES-128-CFB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-ofb -e -in test1.txt -out test1_ofb.enc
enter AES-128-OFB encryption password:
Verifying - enter AES-128-OFB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ ls -lh
total 32K
-rw-rw-r-- 1 seed seed 5 Sep 19 17:23 f1.txt
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f2.txt
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f3.txt
-rw-rw-r-- 1 seed seed 48 Sep 19 18:03 test1_cbc.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_cfb.enc
-rw-rw-r-- 1 seed seed 48 Sep 19 18:02 test1_ecb.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_ofb.enc
-rw-rw-r-- 1 seed seed 25 Sep 19 17:23 test1.txt
seed@seed-virtual-machine: ~/src-cloud/Task4$
```



```
Activities Terminal Sep 21 17:37
seed@seed-virtual-machine: ~/src-cloud/Task4$ ls -l
total 56
-rw-rw-r-- 1 seed seed 16 Sep 19 18:24 f1_cbc.dec
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f1_cbc.enc
-rw-rw-r-- 1 seed seed 5 Sep 19 17:23 f1.txt
-rw-rw-r-- 1 seed seed 16 Sep 19 18:25 f2_cbc.dec
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f2_cbc.enc
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f2.txt
-rw-rw-r-- 1 seed seed 32 Sep 19 18:26 f3_cbc.dec
-rw-rw-r-- 1 seed seed 48 Sep 19 18:18 f3_cbc.enc
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f3.txt
-rw-rw-r-- 1 seed seed 48 Sep 19 18:03 test1_cbc.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_cfb.enc
-rw-rw-r-- 1 seed seed 48 Sep 19 18:02 test1_ecb.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_ofb.enc
-rw-rw-r-- 1 seed seed 25 Sep 19 17:23 test1.txt
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-ecb -d -nopad -in test1_ecb.enc -out test1_ECB.dec
enter AES-128-ECB decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-cbc -d -nopad -in test1_cbc.enc -out test1_CBC.dec
enter AES-128-CBC decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-cfb -d -nopad -in test1_cfb.enc -out test1_CFB.dec
enter AES-128-CFB decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$ openssl enc -aes-128-ofb -d -nopad -in test1_ofb.enc -out test1_OFB.dec
enter AES-128-OFB decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/src-cloud/Task4$
```



```
Activities Terminal Sep 21 17:41
seed@seed-virtual-machine: ~/src-cloud/Task4$ ls -l
total 72
-rw-rw-r-- 1 seed seed 16 Sep 19 18:24 f1_cbc.dec
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f1_cbc.enc
-rw-rw-r-- 1 seed seed 5 Sep 19 17:23 f1.txt
-rw-rw-r-- 1 seed seed 16 Sep 19 18:25 f2_cbc.dec
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f2_cbc.enc
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f2.txt
-rw-rw-r-- 1 seed seed 32 Sep 19 18:26 f3_cbc.dec
-rw-rw-r-- 1 seed seed 48 Sep 19 18:18 f3_cbc.enc
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f3.txt
-rw-rw-r-- 1 seed seed 32 Sep 21 17:36 test1_CBC.dec
-rw-rw-r-- 1 seed seed 48 Sep 19 18:03 test1_cfb.enc
-rw-rw-r-- 1 seed seed 25 Sep 21 17:37 test1_CFB.dec
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_ecb.enc
-rw-rw-r-- 1 seed seed 32 Sep 21 17:36 test1_ECB.dec
-rw-rw-r-- 1 seed seed 48 Sep 19 18:02 test1_ofb.enc
-rw-rw-r-- 1 seed seed 25 Sep 21 17:37 test1_OFB.dec
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_ofb.enc
-rw-rw-r-- 1 seed seed 25 Sep 19 17:23 test1.txt
seed@seed-virtual-machine: ~/src-cloud/Task4$
```

According to this :

We can see that ECB and CBC will pad the 25 byte file with 7 bytes to make it 32 bytes.

CFB and OFB will not give any padding as it turns a block cipher into a stream cipher therefore it doesn't need any padding

ECB Mode: Each block is encrypted separately from the others in ECB mode. Padding is a method used to make sure that the plaintext message to be encrypted is a multiple of the block size before encryption occurs in the ECB (Electronic Codebook) style of encryption. In our example padding is done. When padding is applied in ECB mode, additional bytes are often added to the plaintext to increase its size to a multiple of the block size.

CBC Mode: In this mode before encryption, each block in CBC mode is XORed with the block of ciphertext before it. This tells us that during the encryption the ciphertext of the previous block must be known in order for the block to be encrypted. Padding is necessary to make the plaintext length a multiple of the block size in order to guarantee that the final block can be encrypted. Padding is done according to the PKCS#5 Padding scheme which is commonly used.

To make sure that the plaintext message to be encrypted is a multiple of the block size before encryption occurs, the CBC technique of encryption uses padding. Similar to ECB mode, CBC mode necessitates the division of the plaintext into blocks, with each block having a size that corresponds to the block size of the encryption method (for example, 128 bits for AES-128).

CFB and OFB Mode: CFB and OFB does not add padding as it turns the block cipher into a stream cipher so it can handle a plaintext file with any length without any padding necessary.

In our example we can see no padding is done. Padding is often employed differently in CFB and OFB encryption modes than it is in block cipher modes like CBC or ECB. Both the CFB and OFB stream cipher modes work bit-by-bit or byte-by-byte rather than splitting the plaintext into predetermined blocks.

In conclusion, padding is needed when the plaintext size is not a multiple of the block size for CBC and ECB forms of encryption. This is necessary since both modes work with fixed-size blocks, and padding provides appropriate alignment if the plaintext is not aligned with the block size. In both CBC and ECB modes, padding is required because the encryption of each block depends on the ciphertext block that came before it.

To accomplish block alignment, however, the CFB and OFB stream cipher modes do not have the same severe requirements.

2)

Here we do enc to the files f1.txt , f2.txt , f3.txt to find out the padding that takes place for the files accordingly of size 5, 10, 16 bytes we have done it below and displayed the size

```
Activities Terminal Sep 19 18:19
seed@seed-virtual-machine:~/src-cloud/Task4$ openssl enc -aes-128-cbc -e -in f1.txt -out f1_cbc.enc
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/src-cloud/Task4$ openssl enc -aes-128-cbc -e -in f2.txt -out f2_cbc.enc
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/src-cloud/Task4$ openssl enc -aes-128-cbc -e -in f3.txt -out f3_cbc.enc
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/src-cloud/Task4$ ls -lh
total 44K
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f1_cbc.enc
-rw-rw-r-- 1 seed seed 5 Sep 19 17:23 f1.txt
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f2_cbc.enc
-rw-rw-r-- 1 seed seed 10 Sep 19 17:23 f2.txt
-rw-rw-r-- 1 seed seed 48 Sep 19 18:18 f3_cbc.enc
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f3.txt
-rw-rw-r-- 1 seed seed 48 Sep 19 18:03 test1_cbc.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_cfb.enc
-rw-rw-r-- 1 seed seed 48 Sep 19 18:02 test1_ecb.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_ofb.enc
-rw-rw-r-- 1 seed seed 25 Sep 19 17:23 test1.txt
seed@seed-virtual-machine:~/src-cloud/Task4$
```

Doing decryption along with the -nopad option.

```
Activities Terminal Sep 19 18:19
seed@seed-virtual-machine:~/src-cloud/Task4$ openssl enc -aes-128-cbc -e -in f1_cbc.enc
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/src-cloud/Task4$ openssl enc -aes-128-cbc -e -in f2_cbc.enc
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/src-cloud/Task4$ openssl enc -aes-128-cbc -e -in f3_cbc.enc
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/src-cloud/Task4$ ls -lh
total 44K
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f1_cbc.enc
-rw-rw-r-- 1 seed seed 5 Sep 19 17:23 f1.txt
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f2_cbc.enc
-rw-rw-r-- 1 seed seed 10 Sep 19 17:23 f2.txt
-rw-rw-r-- 1 seed seed 48 Sep 19 18:18 f3_cbc.enc
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f3.txt
-rw-rw-r-- 1 seed seed 48 Sep 19 18:03 test1_cbc.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_cfb.enc
-rw-rw-r-- 1 seed seed 48 Sep 19 18:02 test1_ecb.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_ofb.enc
-rw-rw-r-- 1 seed seed 25 Sep 19 17:23 test1.txt
seed@seed-virtual-machine:~/src-cloud/Task4$
```

Here we can see for different files the padding is done as follows:

```

seed@seed-virtual-machine:~/src-cloud/Task4$ ls -l
total 56
-rw-rw-r-- 1 seed seed 16 Sep 19 18:24 f1_cbc.dec
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f1_cbc.enc
-rw-rw-r-- 1 seed seed 5 Sep 19 17:23 f1.txt
-rw-rw-r-- 1 seed seed 16 Sep 19 18:25 f2_cbc.dec
-rw-rw-r-- 1 seed seed 32 Sep 19 18:17 f2_cbc.enc
-rw-rw-r-- 1 seed seed 10 Sep 19 17:23 f2.txt
-rw-rw-r-- 1 seed seed 32 Sep 19 18:26 f3_cbc.dec
-rw-rw-r-- 1 seed seed 48 Sep 19 18:18 f3_cbc.enc
-rw-rw-r-- 1 seed seed 16 Sep 19 17:23 f3.txt
-rw-rw-r-- 1 seed seed 48 Sep 19 18:03 test1_cbc.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_cfb.enc
-rw-rw-r-- 1 seed seed 48 Sep 19 18:02 test1_ecb.enc
-rw-rw-r-- 1 seed seed 41 Sep 19 18:04 test1_ofb.enc
-rw-rw-r-- 1 seed seed 25 Sep 19 17:23 test1.txt
seed@seed-virtual-machine:~/src-cloud/Task4$ hexdump -C f1_cbc.dec
00000000 31 32 33 34 35 0b |12345.......
00000010
seed@seed-virtual-machine:~/src-cloud/Task4$ hexdump -C f2_cbc.dec
00000000 31 32 33 34 35 36 37 38 39 30 06 06 06 06 06 |1234567890.....
00000010
seed@seed-virtual-machine:~/src-cloud/Task4$ hexdump -C f3_cbc.dec
00000000 31 32 33 34 35 36 37 38 39 31 30 31 31 31 32 31 |1234567891011121
00000010 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 |.....
00000020
seed@seed-virtual-machine:~/src-cloud/Task4$
```

f1.txt padding is 11 bytes from 5 bytes to get a total of 16 bytes

f2.txt padding is 6 bytes from 10 bytes to get a total of 16 bytes

F3.txt padding is 16 bytes from 16 bytes to get a total of 32 bytes

Task 5: Error Propagation – Corrupted Cipher Text

How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively?

Information Recovered According to different Modes are as Follows:

- **ECB:** Every Block except the corrupted block can be recovered. ie the block with the 55th byte is not recoverable
ECB we can get all the information recovered except the corrupted Block. This is because ECB encrypts each block separately and independently. If a bit in a block gets corrupted, then only that block gets corrupted and not recoverable.
- **CBC:** The block containing the corrupted byte and the next block is unrecoverable.
In CBC mode, if a byte in the encrypted file is changed, the associated block of plaintext is corrupted as well as the subsequent block of plaintext, which is completely altered and corrupted. This occurs as a result of the fact that every block of plaintext is combined with the block before it in the ciphertext, and thus altering only one byte in the ciphertext can lead to a series of mistakes throughout the decryption process. The remaining plaintext is not altered.
- **CFB:** In CFB mode generally if one block gets corrupted that affects several other blocks. Since they are dependent on the shift register . So, a corrupted byte in one block might affect subsequent blocks. In CFB Due to the fact that each block's decryption depends on the ciphertext from the previous stage and the shift register, if one byte in the encrypted block is

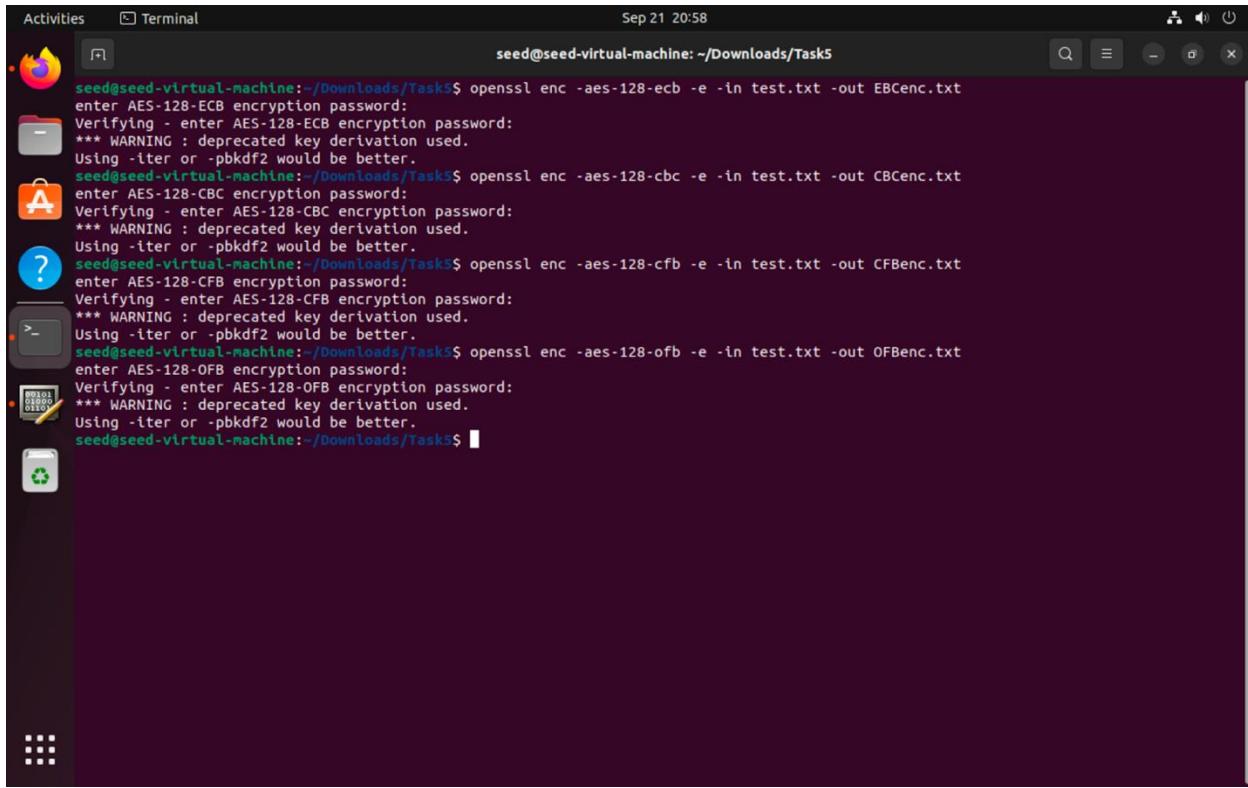
incorrect, it may potentially cause mistakes to spread to following blocks. The first block of plaintext will be impacted by any errors in the first block of ciphertext. Furthermore, because the damaged ciphertext block updates the shift register, the keystream used for next blocks will likewise be tainted. This implies that unless the shift register is emptied and the proper keystream creation starts, mistakes may spread over succeeding blocks.

- **OFB:** Like ECB Every Block except the corrupted block can be recovered .ie the block with the 55th byte is not recoverable. But in OFB it is generally resilient to errors changing one byte only effects that specific byte in a block.
If a byte in the encrypted file changes in OFB (Output Feedback) mode, a different byte will in fact show up in the same spot in the decrypted plaintext block. This is so that faults in the encrypted data only impact the corresponding bits in the decrypted data for that block as OFB mode creates a keystream separately for each block. In contrast to certain other modes, such as CBC, OFB mode does not result in a cascade of mistakes that spread to consecutive blocks when a ciphertext byte is changed. The remaining plaintext is unchanged, and just the block where the modification took place is altered.

Made a file called text.txt of size 1000 bytes



Encrypting the files in different modes using Password: 2537167

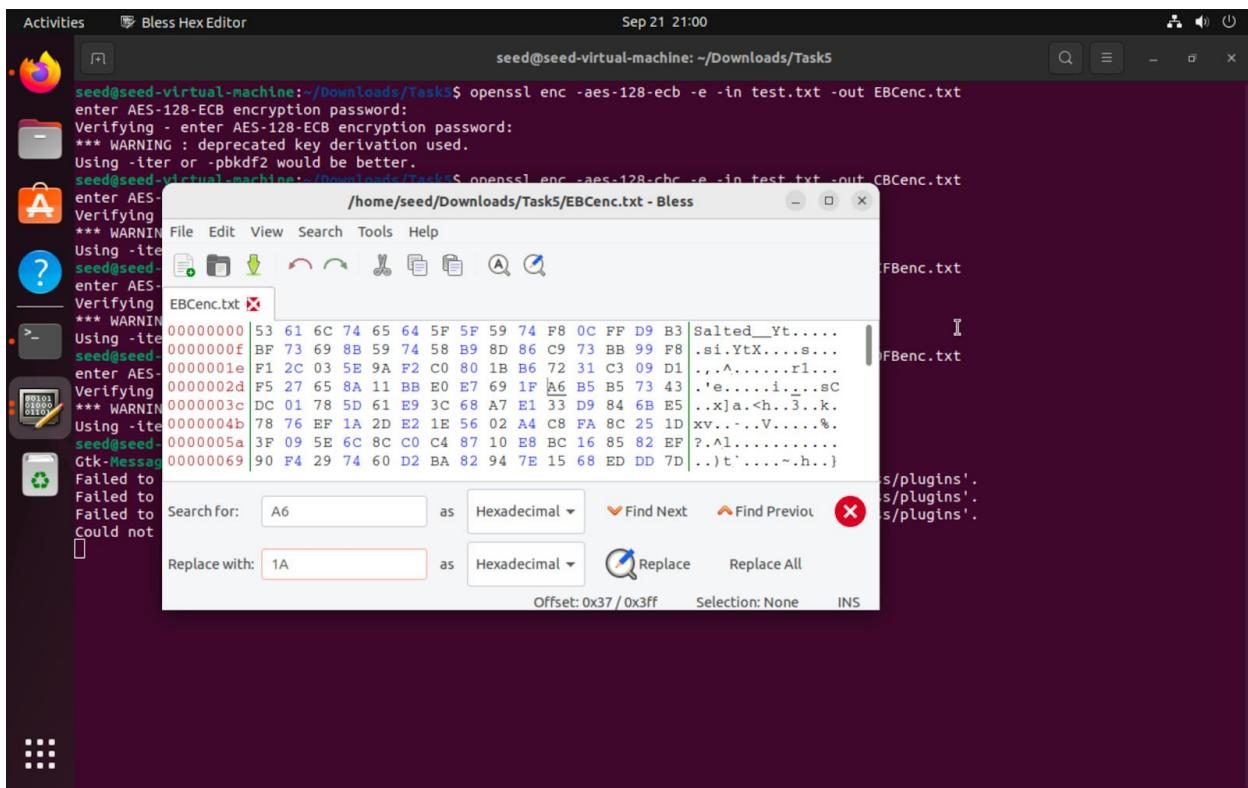


A screenshot of a Ubuntu desktop environment. The terminal window shows the following openssl command outputs:

```
seed@seed-virtual-machine:~/Downloads/Task5$ openssl enc -aes-128-ecb -e -in test.txt -out ECBenc.txt
enter AES-128-ECB encryption password:
Verifying - enter AES-128-ECB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/Downloads/Task5$ openssl enc -aes-128-cbc -e -in test.txt -out CBCenc.txt
enter AES-128-CBC encryption password:
Verifying - enter AES-128-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/Downloads/Task5$ openssl enc -aes-128-cfb -e -in test.txt -out CFBenc.txt
enter AES-128-CFB encryption password:
Verifying - enter AES-128-CFB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/Downloads/Task5$ openssl enc -aes-128-ofb -e -in test.txt -out OFBenc.txt
enter AES-128-OFB encryption password:
Verifying - enter AES-128-OFB encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine:~/Downloads/Task5$
```

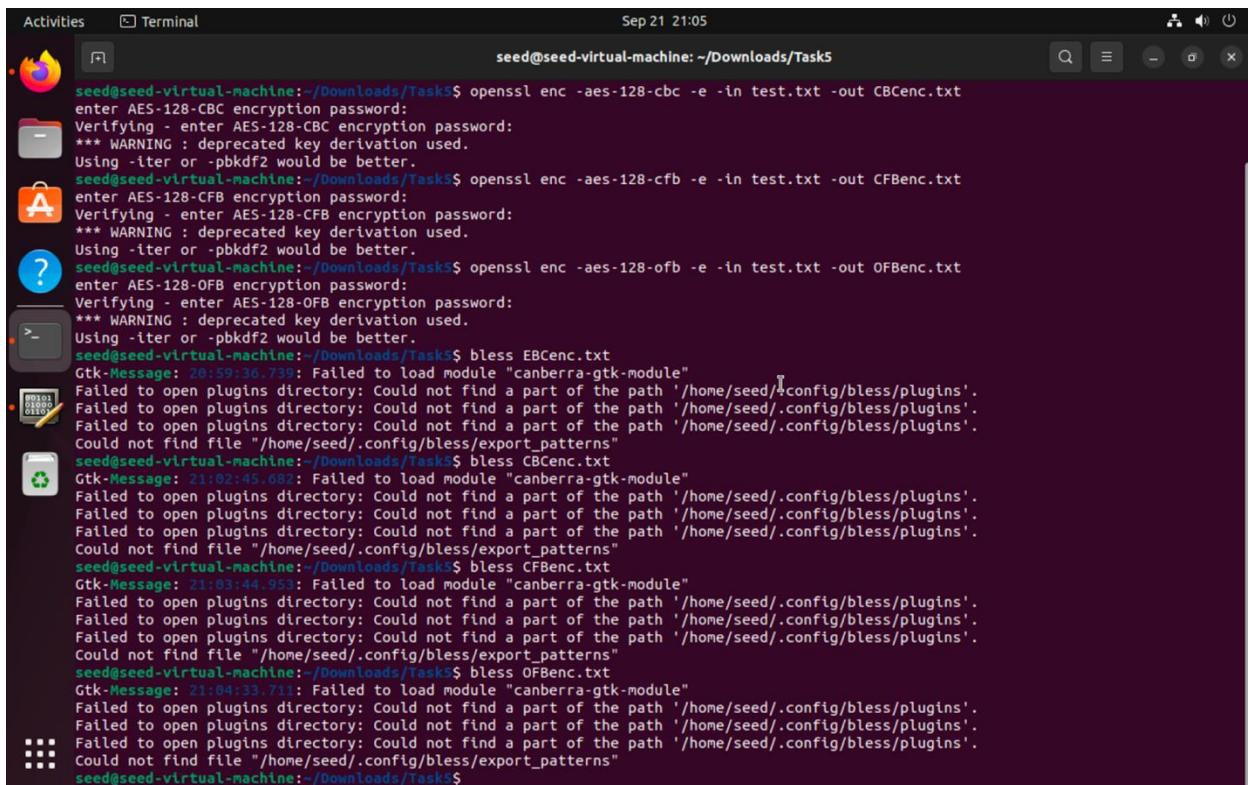
Using Bless to change the 55 bytes in the encrypted text. Here 55th Byte is 37 in Hexadecimal so looking at the 37 positions in Bless.

Here the 55 byte is A6. So, I will corrupt it and replace it with a different character 1A.

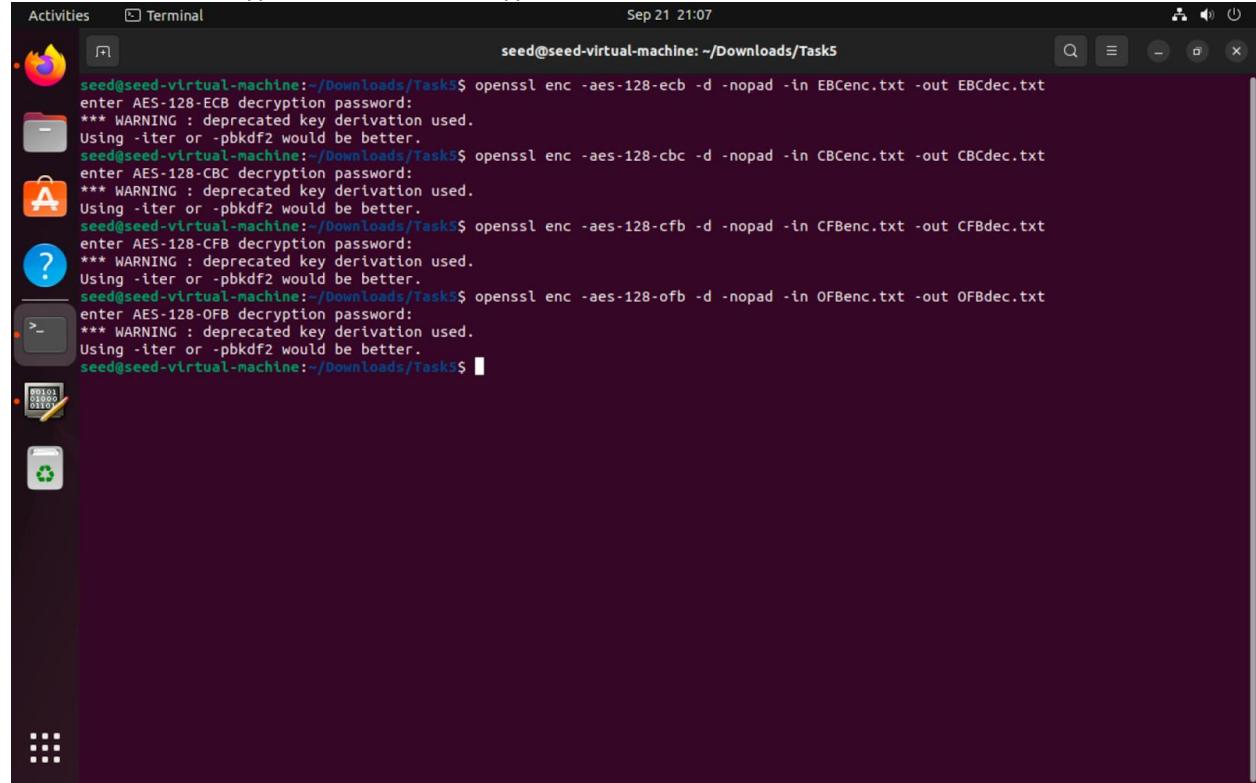


Similarly Doing it for all the modes.

All the enc files are corrupted with their 55 byte corrupted to 1A.



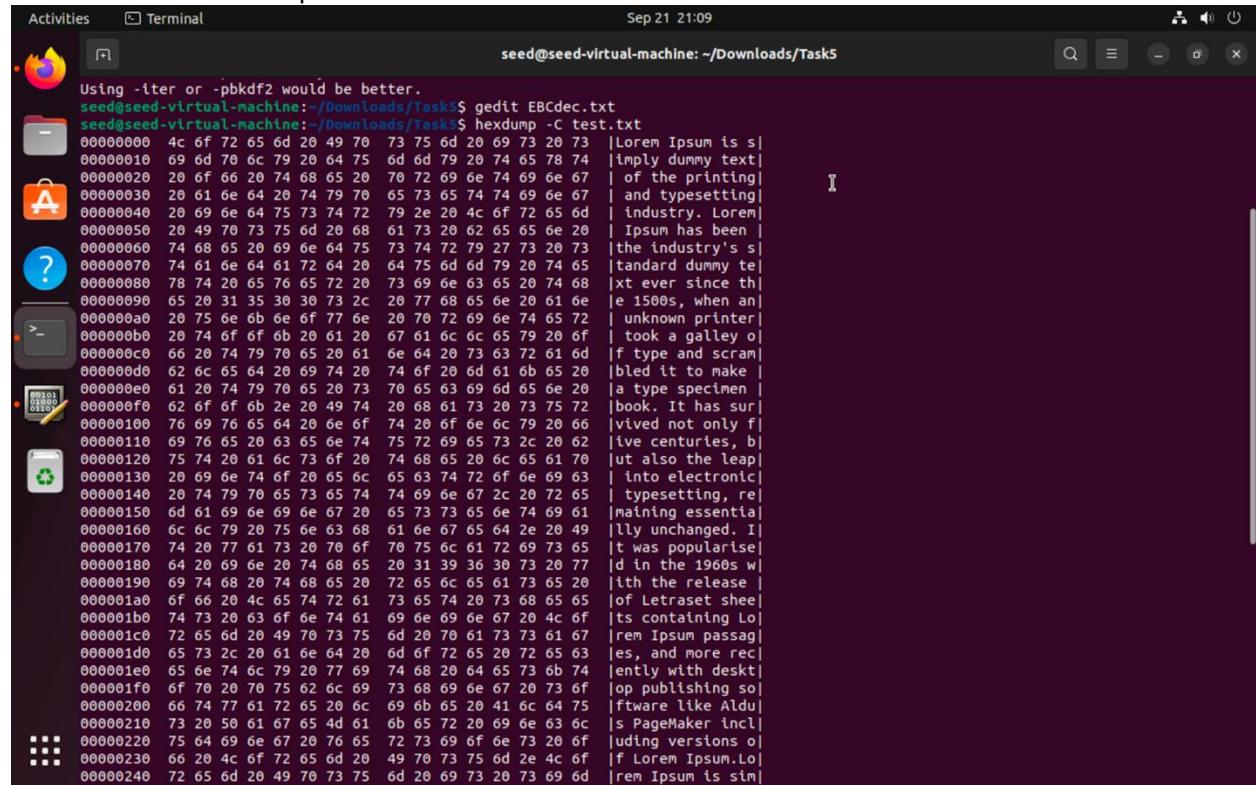
Now let us run decryption for all the encrypted files



```
Activities Terminal Sep 21 21:07
seed@seed-virtual-machine: ~/Downloads/Tasks$ openssl enc -aes-128-ecb -d -nopad -in EBCenc.txt -out EBCdec.txt
enter AES-128-ECB decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/Downloads/Tasks$ openssl enc -aes-128-cbc -d -nopad -in CBCenc.txt -out CBCdec.txt
enter AES-128-CBC decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/Downloads/Tasks$ openssl enc -aes-128-cfb -d -nopad -in CFBenc.txt -out CFBdec.txt
enter AES-128-CFB decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/Downloads/Tasks$ openssl enc -aes-128-ofb -d -nopad -in OFBenc.txt -out OFBdec.txt
enter AES-128-OFB decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/Downloads/Tasks$
```

Now let us check the differences in the file text.txt and the decrypted text in different modes

First here is the hexdump for the test file



```
Activities Terminal Sep 21 21:09
seed@seed-virtual-machine: ~/Downloads/Tasks$ gedit EBCdec.txt
seed@seed-virtual-machine: ~/Downloads/Tasks$ hexdump -C test.txt
Using -iter or -pbkdf2 would be better.
00000000 4c 6f 72 65 6d 20 49 70 73 75 6d 20 69 73 20 73 |Lorem Ipsum is si|
00000010 69 6d 70 6c 79 20 64 75 6d 6d 79 20 74 65 78 74 |mply dummy text|
00000020 20 6f 66 20 74 68 65 20 70 72 69 6e 74 69 6e 67 | of the printing|
00000030 20 61 6e 64 20 74 79 70 65 73 65 74 74 69 6e 67 | and typesetting|
00000040 20 69 6e 64 75 73 74 72 79 2e 20 4c 6f 72 65 6d | industry. Lorem|
00000050 20 49 70 73 75 6d 20 68 61 73 20 62 65 65 6e 20 | Ipsum has been |
00000060 74 68 65 20 69 6e 64 75 73 74 72 79 27 73 20 73 |the industry's si|
00000070 74 61 6e 64 61 72 64 20 64 75 6d 6d 79 20 74 65 |standard dummy te|
00000080 78 74 20 65 76 65 72 20 73 69 6e 63 65 20 74 68 |xt ever since th|
00000090 65 20 31 35 30 30 73 2c 20 77 68 65 6e 20 61 6e |e 1500s, when an|
000000a0 20 75 6e 6b 6e 6f 77 6e 20 70 72 69 6e 74 65 72 | unknown printer|
000000b0 20 74 6f 6f 6b 20 61 20 67 61 6c 6c 65 79 20 6f | took a galley o|
000000c0 66 64 20 74 79 70 65 20 61 6d 64 20 73 63 72 61 |f type and scram|
000000d0 62 6c 65 64 20 69 74 20 74 6f 20 6d 61 6b 65 20 |bled it to make |
000000e0 61 20 74 79 70 65 20 73 70 65 63 69 6d 65 6e 20 |a type specimen |
000000f0 62 6f 6f 6b 2e 20 49 74 20 68 61 73 20 73 75 72 |book. It has sur|
00000100 76 69 76 65 64 20 6e 6f 74 20 6f 6e 6c 79 20 66 |vived not only f|
00000110 69 76 65 20 63 65 6e 74 75 72 69 65 73 2c 20 62 |ive centuries, b|
00000120 75 74 20 61 6c 73 6f 20 74 68 65 20 6c 65 61 70 |ut also the leap|
00000130 20 69 6e 74 6f 20 65 6c 65 63 74 72 6f 6e 69 63 | into electronic|
00000140 20 74 79 70 65 73 65 74 74 69 6e 67 2c 20 72 65 | typesetting, re|
00000150 6d 61 69 6e 69 6e 67 20 65 73 73 65 6e 74 69 61 |maining essentia|
00000160 6c 6c 79 20 75 6e 63 68 61 6e 67 65 64 2e 20 49 |lly unchanged. I|
00000170 74 20 77 61 73 20 70 6f 70 75 6c 61 72 69 73 65 |t was popularise|
00000180 64 20 69 6e 20 74 68 65 20 31 39 36 30 73 20 77 |d in the 1960s w|
00000190 69 74 68 20 74 68 65 20 72 65 6c 65 61 73 65 20 |ith the release |
000001a0 6f 66 20 4c 65 74 72 61 73 65 74 20 73 68 65 65 |of Letraset shee|
000001b0 74 73 20 63 6f 6e 74 61 69 6e 69 6e 67 20 4c 6f |ts containing Lo|
000001c0 72 65 6d 20 49 70 73 75 6d 20 70 61 73 73 61 67 |rem Ipsum passag|
000001d0 65 73 2c 20 61 6e 64 20 6d 6f 72 65 20 72 65 63 |es, and more rec|
000001e0 65 6e 74 6c 79 20 77 69 74 68 20 64 65 73 6b 74 |ently with deskt|
000001f0 6f 70 20 70 75 62 6c 69 73 68 69 6e 67 20 73 6f |op publishing so|
00000200 66 74 77 61 72 65 20 6c 69 6b 65 20 41 6c 64 75 |ftware like Aldu|
00000210 73 20 50 61 67 65 4d 61 6b 65 72 20 69 6e 63 6c |s PageMaker incl|
00000220 75 64 69 6e 67 20 76 65 72 73 69 6f 6e 73 20 6f |uding versions o|
00000230 66 20 4c 6f 72 65 6d 20 49 70 73 75 6d 2e 4c 6f |f Lorem Ipsum. Lo|
00000240 72 65 6d 20 49 70 73 75 6d 20 69 73 20 73 69 6d |rem Ipsum is sim|
```

1. ECB:

Activities Text Editor Sep 21 21:08

EBCdec.txt
~/Downloads/Task5

1 Lorem Ipsum is simply dummy text² and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letras

2

Plain Text Tab Width: 8 Ln 1, Col 54 INS

Activities Terminal Sep 21 21:10

seed@seed-virtual-machine: ~/Downloads/Task5\$ hexdump -C EBCdec.txt

```
000003e0 20 4c 65 74 72 61 73 0a | Letras.|
000003e8
seed@seed-virtual-machine: ~/Downloads/Task5$ hexdump -C EBCdec.txt
00000000 4c 6f 72 65 6d 20 49 70 73 75 6d 20 69 73 20 73 |Lorem Ipsum is s|
00000010 69 6d 70 6e 79 20 64 75 6d 6d 79 20 74 65 78 74 |imply dummy text|
00000020 ad ae b2 5e 9a 23 8b bd 6c 82 9b 22 83 c2 ca d5 |...^.#..!...|...
00000030 20 61 6e 64 20 74 79 70 65 73 65 74 74 69 6e 67 | and typesetting|
00000040 74 69 6e 64 75 73 74 72 79 2e 20 4c 6f 72 65 6d | industry. Lorem|
00000050 20 49 70 73 75 6d 20 68 61 73 20 62 65 65 6e 20 | Ipsum has been |
00000060 74 68 65 20 69 6e 64 75 73 74 72 79 27 73 20 73 |the industry's s|
00000070 74 61 6e 64 61 72 64 20 64 75 6d 6d 79 20 74 65 |standard dummy te|
00000080 78 74 20 65 75 65 72 20 73 69 6e 63 65 20 74 68 |xt ever since th|
00000090 65 20 31 35 30 38 73 2c 20 77 68 65 6e 20 61 6e |e 1500s, when an|
000000a0 20 75 6e 6b 6e 6f 77 6e 20 70 72 69 6e 74 65 72 |unknown printer|
000000b0 20 74 6f 6f 6b 20 61 20 67 61 6c 6c 65 79 20 6f | took a galley o|
000000c0 66 20 74 79 70 65 20 61 6e 64 20 73 63 72 61 6d |f type and scram|
000000d0 62 6c 65 6d 20 69 74 20 74 6f 20 6d 61 6b 65 20 |bled it to make |
000000e0 61 20 74 79 70 65 20 73 70 65 63 69 6d 65 6e 20 |a type specimen |
000000f0 62 6f 6f 6b 2e 20 49 74 20 68 61 73 20 73 75 72 |book. It has sur|
00000100 76 69 76 65 64 20 6e 6f 74 20 6f 6e 6c 79 20 66 |vived not only f|
00000110 69 76 65 20 63 65 6e 74 75 72 69 65 73 2c 20 62 |ive centuries, b|
00000120 75 74 20 61 6c 73 6f 20 74 68 65 20 6c 65 61 70 |ut also the leap|
00000130 20 69 6e 74 6f 20 65 6c 65 63 74 72 6f 6e 69 63 | into electronic|
00000140 20 74 79 70 65 73 65 74 74 69 6e 67 2c 20 72 65 | typesetting, re|
00000150 6d 61 69 6e 69 6e 67 20 65 73 73 65 6e 74 69 61 |maining essential|
00000160 6c 6c 79 20 75 6e 63 68 61 6e 67 65 64 2e 20 49 |lly unchanged. I|
00000170 74 20 77 61 73 20 70 6f 70 75 6c 61 72 69 73 65 |t was popularise|
00000180 64 20 69 6e 20 74 68 65 20 31 39 36 30 73 20 77 |d in the 1960s w|
00000190 69 74 68 20 74 68 65 20 72 65 6c 65 61 73 65 20 |ith the release |
000001a0 6f 66 20 4c 65 74 72 61 73 65 74 20 73 68 65 65 |of Letraset she|
000001b0 74 73 20 63 6f 6e 74 61 69 6e 69 6e 67 20 4c 6f |ets containing Lo|
000001c0 72 65 6d 20 49 70 73 75 6d 20 78 61 73 73 61 67 |rem Ipsum passag|
000001d0 65 73 2c 20 61 6e 64 20 6d 6f 72 65 20 72 65 63 |es, and more rec|
000001e0 65 6e 74 6c 79 20 77 69 74 68 20 64 65 73 6b 74 |ently with desk|
000001f0 6f 70 20 70 75 62 6c 69 73 68 69 6e 67 20 73 6f |op publishing so|
00000200 66 74 77 61 72 65 20 6c 69 6b 65 20 41 6c 64 75 |ftware like Aldu|
00000210 73 20 50 61 67 65 4d 61 6b 65 72 20 69 6e 63 6c |s PageMaker incl|
00000220 75 64 69 6e 67 20 76 65 72 73 69 6f 6e 73 20 6f |uding versions o|
00000230 66 20 4c 6f 72 65 6d 20 49 70 73 75 6d 2e 4c 6f |f Lorem Ipsum. Lo|
00000240 72 65 6d 20 49 70 73 75 6d 20 69 73 20 73 69 6d |rem Ipsum is sim|
```

Comparison between Plaintext and decrypted file only one block is affected other blocks are same.

2. CBC :

A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications like a web browser, file manager, and terminal. The main window is a terminal with a dark background. At the top, it shows the file path 'CBCdec.txt' and the line 'Sep 21 21:11'. The terminal content is a large block of Spanish placeholder text (Lorem Ipsum). Below the terminal, another window is partially visible, showing a file viewer with the same text content. Status bars at the top and bottom of the screen also show the date and time.

Comparison between Plaintext and decrypted file 2 blocks is affected other blocks are same.

3. CFB :

Activities Text Editor Sep 21 21:20

Open Save

CFBdec.txt
~/Downloads/Task5

```
1 Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letras
```

Activities Terminal Sep 21 21:21

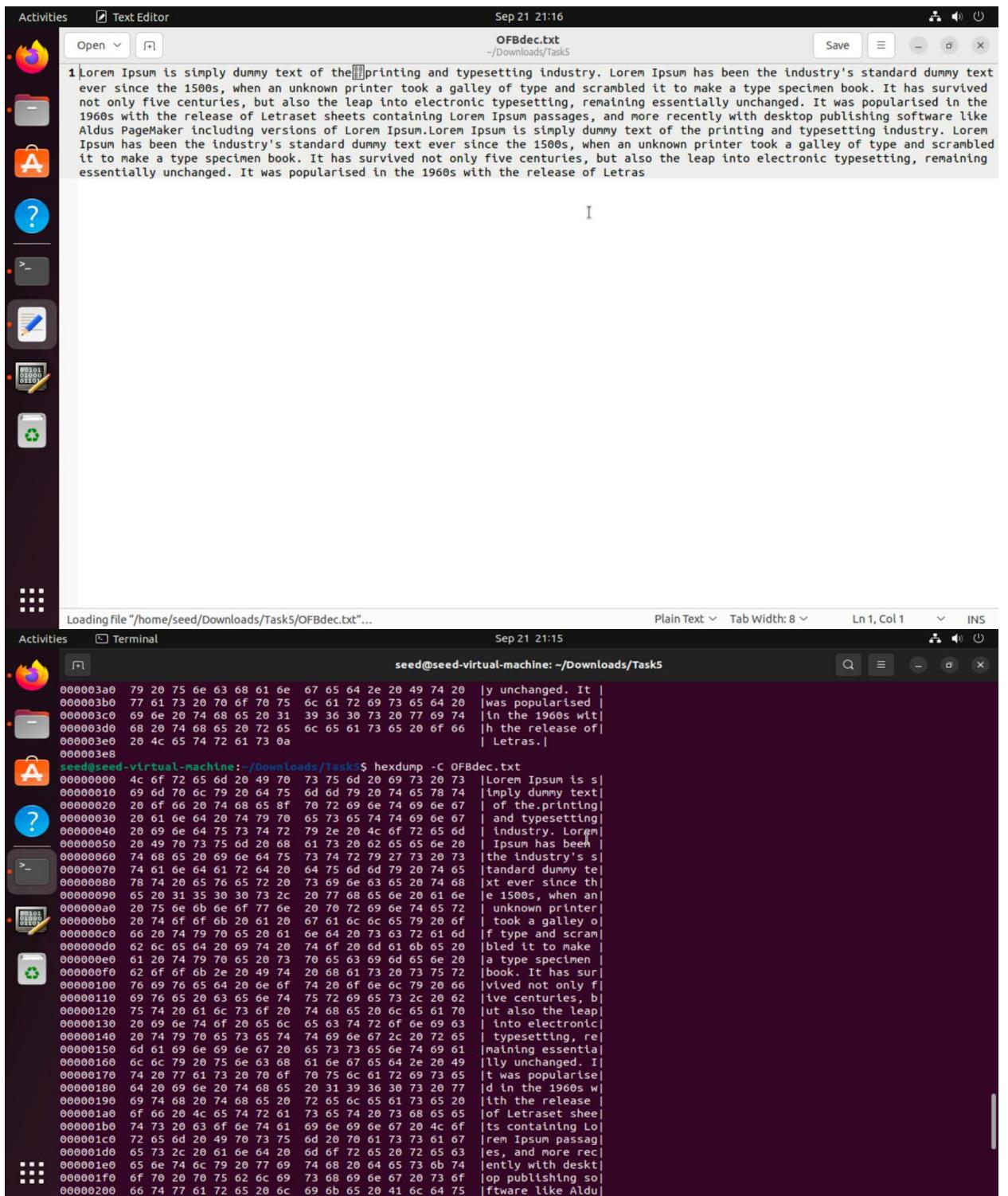
Plain Text Tab Width: 8 Ln 1, Col 1 INS

seed@seed-virtual-machine: ~/Downloads/Task5

```
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
seed@seed-virtual-machine: ~/Downloads/Task5$ hexdump -C CFBdec.txt
00000000 4c 6f 72 65 6d 20 49 70 73 75 6d 20 69 73 20 73  |[Lorem Ipsum is s]
00000010 69 6d 70 6c 79 20 64 75 6d 6d 79 20 74 65 78 74  |imply dummy text|
00000020 20 6f 66 20 74 68 65 27 70 72 69 6e 74 69 6e 67  | of the printing|
00000030 da 0b ee 5d e8 c8 3b 9c d0 55 b6 b5 3d e4 20 bc [...]...;..U..=. .|
00000040 20 69 6e 64 75 73 74 72 79 2e 20 4c 6f 72 65 6d  | industry. Loren|
00000050 20 49 70 73 75 6d 20 68 61 73 20 62 65 65 6e 20  | Ipsum has been |
00000060 74 68 65 20 69 6e 64 75 73 74 72 79 27 73 20 73  |the industry's s|
00000070 74 61 6e 64 61 72 64 20 64 75 6d 6d 79 20 74 65  |standard dummy te|
00000080 78 74 20 65 76 65 72 20 73 69 6e 63 65 20 74 68  |xt ever since th|
00000090 65 20 31 35 30 30 73 2c 20 77 68 65 6e 20 61 6e  |e 1500s, when an|
000000a0 20 75 66 6b 6e 6f 77 6e 20 70 72 69 6e 74 65 72  | unknown printer|
000000b0 20 74 6f 6f 6b 20 61 20 67 61 6c 6c 65 79 20 6f  | took a galley o|
000000c0 66 20 74 79 70 65 20 61 6e 20 73 63 72 61 6d  |f type and scram|
000000d0 62 6c 65 64 20 69 74 20 74 6f 20 6d 61 6b 65 20  |bled it to make |
000000e0 61 20 74 79 70 65 20 73 70 65 63 69 6d 65 6e 20  |a type specimen |
000000f0 62 6f 6f 6b 2e 20 49 74 20 68 61 73 20 73 75 72  |book. It has sur|
00000100 76 69 76 65 64 20 6e 6f 74 20 6f 6e 6c 79 20 66  |vived not only f|
00000110 69 76 65 20 63 65 6e 74 75 72 69 65 73 2c 20 62  |ive centuries, b|
00000120 75 74 20 61 6c 73 6f 20 74 68 65 20 6c 65 61 70  |ut also the leap|
00000130 20 69 6e 74 6f 20 65 6c 65 63 74 72 6f 6e 69 63  | into electronic|
00000140 20 74 79 70 65 73 65 74 74 69 6e 67 2c 20 72 65  | typesetting, re|
00000150 6d 61 69 6e 69 6e 67 20 65 73 73 65 6e 74 69 61  |maining essential|
00000160 6c 6c 79 20 75 6e 63 68 61 6e 67 65 64 2e 20 49  |lly unchanged. I|
00000170 74 20 77 61 73 20 70 6f 70 75 6c 61 72 69 73 65  |t was popularise|
00000180 64 20 69 6e 20 74 68 65 20 31 39 36 30 73 20 77  |d in the 1960s w|
00000190 69 74 68 20 74 68 65 20 72 65 6c 65 61 73 65 20  |ith the release |
000001a0 6f 66 20 4c 65 74 72 61 73 65 74 20 73 68 65 65  |of Letraset sheet|
000001b0 74 73 20 63 6f 6e 74 61 69 6e 69 6e 67 20 4e 6f  |ts containing Lo|
000001c0 72 65 6d 20 49 70 73 75 6d 20 70 61 73 61 67  |rem Ipsum passag|
000001d0 65 73 2c 20 61 6e 64 20 6d 6f 72 65 20 72 65 63  |es, and more rec|
000001e0 65 6e 74 6c 79 20 77 69 74 68 20 64 65 73 6b 74  |ently with desk|
000001f0 6f 70 20 70 75 62 6c 69 73 68 69 6e 67 20 73 6f  |op publishing so|
00000200 66 74 77 61 72 65 20 6c 69 6b 65 20 41 6c 64 75  |ftware like Aldu|
00000210 73 20 50 61 67 65 4d 61 6b 65 72 20 69 6e 63 6c  |s PageMaker incl|
00000220 75 64 69 6e 67 20 76 65 72 73 69 6e 73 20 6f  |uding versions o|
00000230 66 20 4c 6f 72 65 6d 20 49 70 73 75 6d 2e 4c 6f  |f Lorem Ipsum. Lo|
00000240 72 65 6d 20 49 70 73 75 6d 20 69 73 20 73 69 6d  |rem Ipsum is sim|
```

Comparison between Plaintext and decrypted file two blocks are affected other blocks are same.

4. OFB :



```

Activities   Text Editor
Sep 21 21:16
OFBdec.txt
~/Downloads/Task5
Save  □  ×

1|Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Activities   Terminal
Plain Text  Tab Width: 8  Ln 1, Col 1  INS
seed@seed-virtual-machine: ~/Downloads/Task5
Sep 21 21:15
seed@seed-virtual-machine: ~/Downloads/Task5$ hexdump -C OFBdec.txt
000003a0  79 20 75 6e 63 68 61 6e 67 65 64 2e 20 49 74 20 |y unchanged. It |
000003b0  77 61 73 20 70 6f 70 75 6c 61 72 69 73 65 64 20 |was popularised |
000003c0  69 6e 20 74 68 65 20 31 39 36 30 73 20 77 69 74 |in the 1960s wit|
000003d0  68 20 74 68 65 20 72 65 6c 65 61 73 65 20 6f 66 |h the release of|
000003e0  20 4c 65 74 72 61 73 0a                                |Letras.|

000003e8
seed@seed-virtual-machine: ~/Downloads/Task5$ hexdump -C OFBdec.txt
00000000  4c 6f 72 65 6d 20 49 70 73 75 6d 20 69 73 20 73 |Lorem Ipsum is s|
00000010  69 6d 70 6c 79 20 64 75 6d 6d 79 20 74 65 78 74 |imply dummy text|
00000020  20 6f 6d 20 74 68 65 8f 70 72 69 6e 74 69 6e 67 | of the printing|
00000030  20 61 6e 64 20 74 79 70 65 73 65 74 74 69 6e 67 | and typesetting|
00000040  20 69 6e 64 75 73 74 72 79 2e 20 4c 6f 72 65 6d | industry. Lorem|
00000050  20 49 70 73 75 6d 20 68 61 73 20 62 65 65 6e 20 | Ipsum has been |
00000060  74 68 65 20 69 6e 64 75 73 74 72 79 27 73 20 73 |the industry's s|
00000070  74 61 6e 64 61 72 64 20 64 75 6d 6d 79 20 74 65 |standard dummy te|
00000080  78 74 20 65 76 65 72 20 73 69 6e 63 65 20 74 68 |xt ever since th|
00000090  65 20 31 35 30 30 73 2c 20 77 68 65 6e 20 61 6e |e 1500s, when an|
000000a0  20 75 66 6b 6e 6f 77 6e 20 70 72 69 6e 74 65 72 |unknown printer|
000000b0  20 74 6f 6f 6b 20 61 20 67 61 6c 6c 65 79 20 6f | took a galley o|
000000c0  66 20 74 79 70 65 20 61 6e 64 20 73 63 72 61 6d |f type and scram|
000000d0  62 6c 65 64 20 69 74 20 74 6f 20 6d 61 6b 65 20 |bled it to make |
000000e0  61 20 74 79 70 65 20 73 70 65 63 69 6d 65 6e 20 |a type specimen |
000000f0  62 6f 6f 6b 2e 20 49 74 20 68 61 73 20 73 75 72 |book. It has sur|
00000100  76 69 76 65 64 20 6f 74 20 6f 6e 6c 79 20 66 |vived not only f|
00000110  69 76 65 20 63 65 6e 74 75 72 69 65 73 2c 20 62 |ive centuries, b|
00000120  75 74 20 61 6c 73 6f 20 74 68 65 20 6c 65 61 70 |ut also the leap|
00000130  20 69 6e 74 6f 20 65 6c 65 63 74 72 6f 6e 69 63 |into electronic|
00000140  20 74 79 70 65 73 65 74 74 69 6e 67 2c 20 72 65 |typesetting, re|
00000150  6d 61 69 6e 69 6e 67 20 65 73 73 65 6e 74 69 61 |maining essential|
00000160  6c 6c 79 20 75 6e 63 68 61 6e 67 65 64 2e 20 49 |ly unchanged. I|
00000170  74 20 77 61 73 20 70 6f 70 75 6c 61 72 69 73 65 |t was popularise|
00000180  64 20 69 6e 20 74 68 65 20 31 39 36 30 73 20 77 |d in the 1960s w|
00000190  69 74 68 20 74 68 65 20 72 65 6c 65 61 73 65 20 |ith the release |
000001a0  6f 66 20 4c 65 74 72 61 73 65 74 20 73 68 65 65 |of Letraset shee|
000001b0  74 73 20 63 6f 6e 74 61 69 6e 69 6e 67 20 4c 6f |ts containing Lo|
000001c0  72 65 6d 20 49 70 73 75 6d 20 78 61 73 73 61 67 |rem Ipsum passag|
000001d0  65 73 2c 20 61 6e 64 20 6d 6f 72 65 20 72 65 63 |es, and more rec|
000001e0  65 6e 74 6c 79 20 77 69 74 68 20 64 65 73 6b 74 |ently with desk|
000001f0  6f 70 20 70 75 62 6c 69 73 68 69 6e 67 20 73 6f |top publishing so|
00000200  66 74 77 61 72 65 20 6c 69 6b 65 20 41 6c 64 75 |ftware like Aldu|

```

Comparison between Plaintext and decrypted file only one block is affected other blocks are same.(specifically 1 bit)

Summary:

A different byte will appear in the same location in the plaintext block if a byte in the encrypted file changes in OFB mode, as opposed to the plaintext, where the rest of the plaintext is unaffected.

In ECB mode in contrast to the plaintext, if a byte in the encrypted file is changed, the resulting block of plaintext is entirely different, while the remainder of the plaintext is unaffected.

In CBC mode, a single byte change in the ciphertext of a block results in a corrupted block of plaintext at that position and another corrupted block in the next position, while the rest of the plaintext is unaffected. Data integrity and authenticity checks are crucial in cryptographic systems utilizing the CBC mode because of this error propagation feature.

In conclusion, unless the shift register is synced once again, flaws in one block of ciphertext might actually spread to succeeding blocks when using CFB mode. Since any inaccuracy in the ciphertext might possibly cause a chain reaction of faults in the decrypted data, data integrity is essential in CFB mode.