

Secret Key Encryption Project Report

Task 1: Frequency Analysis

Description: In this task, we applied frequency analysis to decipher a ciphertext that was encrypted using a monoalphabetic substitution cipher. The goal was to identify the original plaintext based on the frequency of individual letters and common letter pairs (bigrams and trigrams) in English.

Observations:

- By analyzing the frequency of letters in the ciphertext, we matched high-frequency letters to common letters in the English language.
- Bigram and trigram frequencies were analyzed, helping us identify frequent letter combinations and map them to common English word patterns.

Approach:

- Used the freq.py script to generate 1-gram, 2-gram, and 3-gram statistics from the ciphertext file. This script reads the file and identifies the most common patterns to help decipher the text.
- Mapped frequent ciphertext letters to common English letters such as 'E', 'T', 'A', etc
- Replaced ciphertext characters gradually to reveal the original message.

```
#!/usr/bin/env python3

from collections import Counter
import re

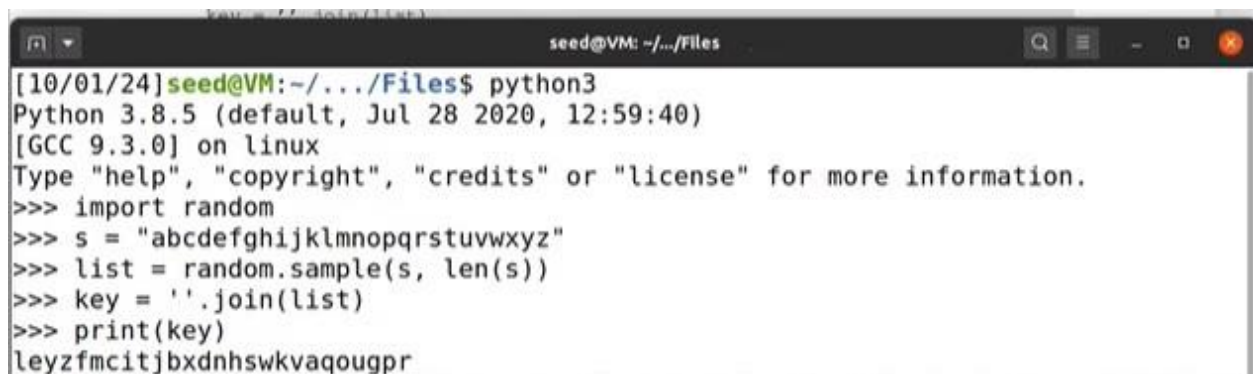
TOP_K = 20
N_GRAM = 3

# Generate all the n-grams for value n
def ngrams(n, text):
    for i in range(len(text) - n + 1):
        # Ignore n-grams containing white space
        if not re.search(r'\s', text[i:i+n]):
            yield text[i:i+n]

# Read the data from the ciphertext
with open('ciphertext.txt') as f:
    text = f.read()

# Count, sort, and print out the n-grams
for N in range(N_GRAM):
    print("-----")
    print("{}-gram (top {}):".format(N+1, TOP_K))
    counts = Counter(ngrams(N+1, text)) # Count
    sorted_counts = counts.most_common(TOP_K) # Sort
    for ngram, count in sorted_counts:
        print("{}: {}".format(ngram, count)) # Print
```

Task1 (freq.py)



```
seed@VM: ~/.../Files
[10/01/24] seed@VM:~/.../Files$ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> s = "abcdefghijklmnopqrstuvwxyz"
>>> list = random.sample(s, len(s))
>>> key = ''.join(list)
>>> print(key)
leyzfmcitjbxdnhswkvaougpr
```

Task1 (imported random in order to generate key)

```
Seed-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 1 19:32 seed@VM: ~/../Files
[10/01/24]seed@VM:~/../Files$ ./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
qy: 39
vq: 33
vi: 32
nn: 32
```

Task1 (This is the n-gram that was produced)

```
Seed-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 1 19:33 seed@VM: ~/.../Files

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
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
ytn: 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
[10/01/24]seed@VM:~/.../Files$
```

Task1 (This is the n-gram that was produced)

```
Oct 1 19:34
ciphertext.txt
~/Desktop/Labsetup/Files
Save
1ytn xqavhq yzhu xu qzupvd lmat qnncq vxgzy hmrty vbynh ytmq ixur qyhvurn
2vlvhpq yhme ytn gyrrnh bnaiq imsn v uxuvrnvuhmvu yxx
3
4ytn vlvhpq hvan lvq gxxsnupnp gd ytn pncmqn xb tvhfn lnuqynmu vy myq xzyqny
5vup ytn veevhnuu mceixqmxu xb tmq bmic axcevud vy ytn nup vup my lvq qtenp gd
6ytn ncnhrnua xb cnyxx ymcnq ze givasrllu eximymaq vhcavupd vaymfmqc vup
7v uyvmxuvi axufnhqvymxu vq ghmbb vup cvp vq v bnfhn phnvc vxgzy ltnytnh ytnhn
8xzrty yx gn v ehngmpnuu lmubhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
9nkyhv ixur gnayzqn ytn xqavhq lnhn cxfnp yx ytn bmqy lnnsnup mu cvhat yx
10vfxmp axubimaymur lmyt ytn aixqmur anhnxcud xb ytn lmuynh xidcemaq ytvusq
11ednxuratvur
12
13xun gmr jznqymxu qzhxhzupmur ytmq dnvhq vavpncd vlvhpq mq txl xh mb ytn
14anhnxcud lmi l vpphnq cnyxx nqenamviid vbynh ytn rxipnu rixgnq lmat gnacn
15v ozgmivuy axcmurxzy evhyd bxh ymcnq ze ytn cxfncnuu qenvhtnvpnp gd
16exlnhbzi txiidlxxp lxcnu ltx tnienn hvmqn cmiimxuq xb pxliivhq yx bmrty qnkzvi
17tvhvqqcnuu vhxzup ytn axzuyhd
18
19qmruvimur ytnmh qzeexhy rxipnu rixgnq vyynupnq qlvytnp ytnqnfinq mu givas
20qexhynp iveni emuq vup qzupnp xbb vxgzy qnkmqy exlnh mcgvivuanq bhxc ytn hnp
21avheny vup ytn qyvrn xu ytn vmh n lvq aviinp zxy vxgzy evd munjzmyd vbynh
22myq bxhcnh vuatxh avyy qvpinh jzmy xuan qtn invhnp yty qtn lvq cvsmur bvh
23inqq ytvu v cvin axtxqy vup pzhmur ytn anhnxcud uvyvimn exhyvuv yxxs v gizuy
24vup qvymqbdmur pmr vy ytn viicvin hxqynh xb uxcmuvynp pmhnayxhq txl axzip
25ytvy gn yxeenp
26
27vq my yzhuq zxy vy invqy mu ynbcq xb ytn xqavhq my ehxgvgid lxuy gn
28
29lxcnu mufxifnp mu ymcnq ze qvmp yty vlytxzrt ytn rixgnq qmrumbmnp ytn
30mumymvmymfnq ivzuat ytn unfn muynupnp my yx gn ozqy vu vlvhpq qnvqxu
31avcevmru xh xun yty gnacn vqxxamvynp xuid lmyt hnpavheny vaymxuq muqynvp
32v qexsnqlxcvu qvmp ytn rhxze mq lxhsmur gntmup aixqnp pxxhq vup tvq qmuan
33vcvqqnp cmiimxu bxh myq inrvi pnbnuqn bzup lmat vbynh ytn rixgnq lvq
34bixxnp lmyt ytxzqvupq xb pxuvymxuq xb xh inq bhxc enxein mu qxcn
35axzuyhmq
36
37
38ux avii yx lnh givas rxluq lnuu zxy mu vpfvuan xb ytn xqavhq ytxzrt ytn
39cxfncnuu lmi vixqy anhyvmuid gn hnbhnuanp gnbxhn vup pzhmur ytn anhnxcud
40nqenamviid qmuan fxavi cnyxx qzeexhynhq imsn vqind ozpp ivzhv pnhu vup
41umaxin smpcvu vhn qatnpzin ehngnuynhq
42
43vuxvtnh bnvyzhn xb ytmq anvaxu ux xun hnviid suxla ltx ma rxmur vx lmu anay
```

Task1 (ciphertext)

Based on the n-gram, we can analyze the ciphertext.txt file. The encryption key could be as follows:

ytn → THE

y → T

t → H

n → E

tn → HE

.

.

.

so on...

```
Seed-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 1 19:29
seed@VM: ~/.../Files

the oscars turn on sunday which seems about right after this long strange
awards trip the bagger feels like a nonagenarian too

the awards race was bookended by the demise of harvey weinstein at its outset
and the apparent implosion of his film company at the end and it was shaped by
the emergence of metoo times up blackgown politics armcandy activism and
a national conversation as brief and mad as a fever dream about whether there
ought to be a president winfrey the season didnt just seem extra long it was
extra long because the oscars were moved to the first weekend in march to
avoid conflicting with the closing ceremony of the winter olympics thanks
pyeongchang

one big question surrounding this years academy awards is how or if the
ceremony will address metoo especially after the golden globes which became
a jubilant comingout party for times up the movement spearheaded by
powerful hollywood women who helped raise millions of dollars to fight sexual
harassment around the country

signaling their support golden globes attendees swathed themselves in black
sporting lapel pins and sounded off about sexist power imbalances from the red
carpet and the stage on the air e was called out about pay inequity after
its former anchor catt sadler quit once she learned that she was making far
less than a male cohost and during the ceremony natalie portman took a blunt
and satisfying dig at the allmale roster of nominated directors how could
that be topped

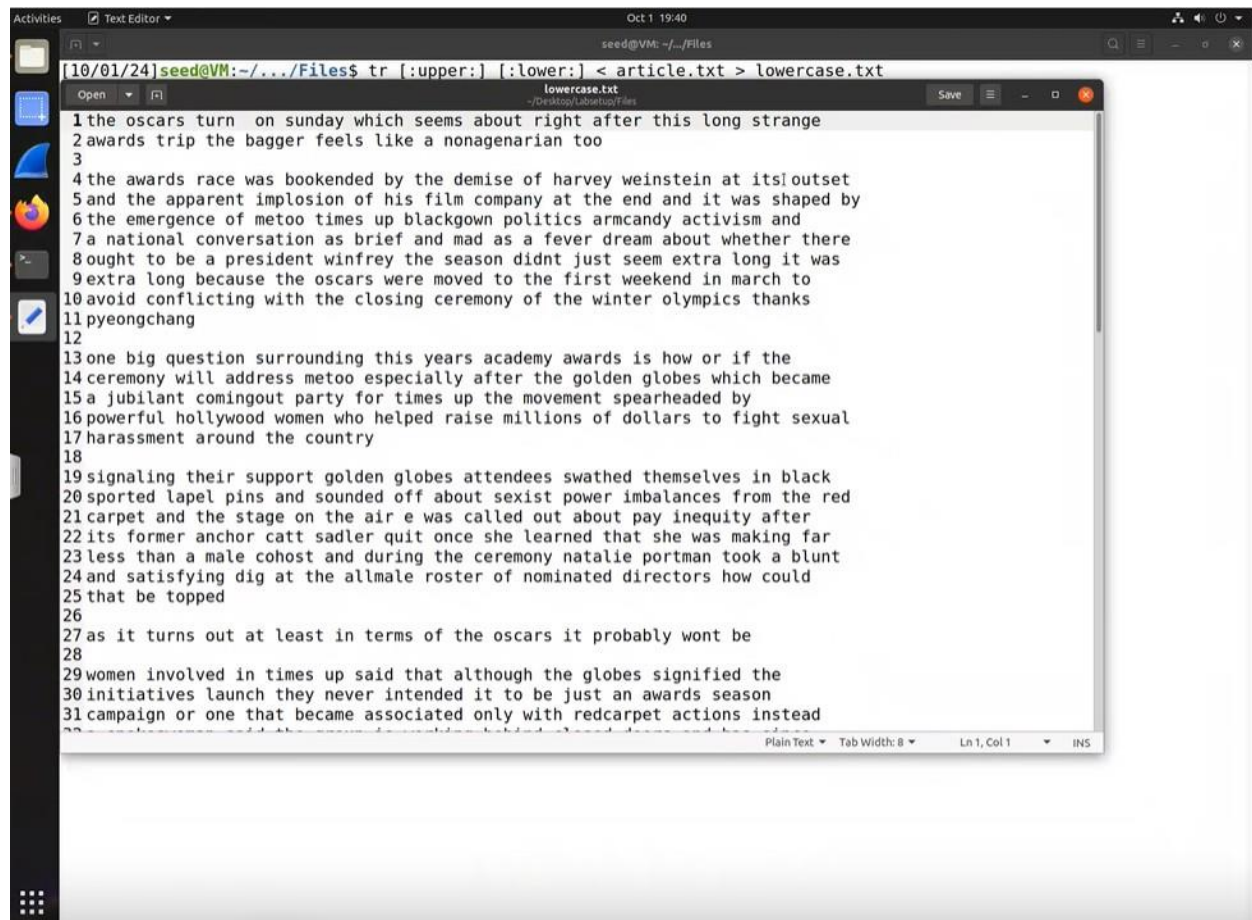
as it turns out at least in terms of the oscars it probably wont be

women involved in times up said that although the globes signified the
initiatives launch they never intended it to be just an awards season
campaign or one that became associated only with redcarpet actions instead
a spokeswoman said the group is working behind closed doors and has since
amassed million for its legal defense fund which after the globes was
flooded with thousands of donations of or less from people in some
countries

no call to wear black gowns went out in advance of the oscars though the
movement will almost certainly be referenced before and during the ceremony
especially since vocal metoo supporters like ashley judd laura dern and
nicole kidman are scheduled presenters

1,1
```

Task1 (decrypted ciphertext)



The screenshot shows a Linux desktop environment with a terminal window open. The terminal prompt is `seed@VM: ~/Files`. The command `tr [:upper:] [:lower:] < article.txt > lowercase.txt` has been executed. The output is a text file named `lowercase.txt` containing the following text:

```
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 sported 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
```

Task1 (tr [:upper:] [:lower:] < article.txt > lowercase.txt has been used)

The actual alphabet letter goes first, followed by the cipher key it corresponds to:

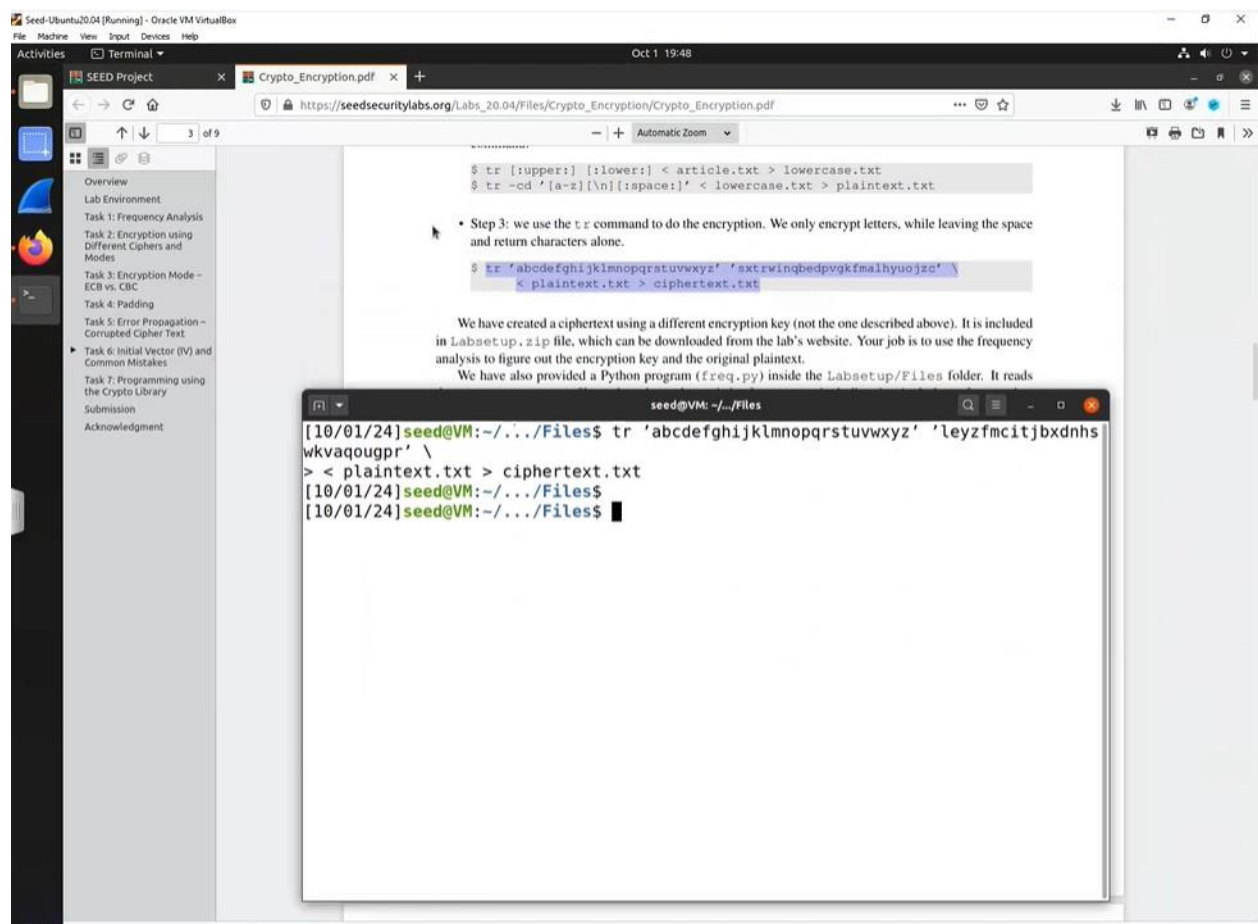
- A = V • B = G • C = A • D = P • E = N • F = B • G = R • H = T • I = M • J = W
- K = S • L = I • M = C • N = U • O = X • P = E • Q = O • R = H • S = Q • T = Y
- U = Z • V = F • W = L • X = K • Y = D • Z = J


```
Oct 1 19:40
seed@VM: ~/Files
[10/01/24]seed@VM:~/Files$ tr [:upper:] [:lower:] < article.txt > lowercase.txt
[10/01/24]seed@VM:~/Files$ tr -cd '[a-z][\n]':[:space:]' < lowercase.txt > plaintext.txt
```

plaintext.txt

```
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|sported 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
```

Task1 (tr -cd '[a-z][\n]':[:space:]' < lowercase.txt > plaintext.txt has been used)



Task1 (tr 'abcdefghijklmnopqrstuvwxyz' 'sxtwinqbedpvgkfmalhyuojzc' \ < plaintext.txt > ciphertext.txt has been used)

```
Seed-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Text Editor
Oct 1 19:49
ciphertxt.txt
Save
1 aif hvylkv aqkn hn vqnzlp uityi vffdv lehqa ktcia lmafk aitu xhnc vaklncf
2 lulkzv akts aif elccfk mffxv xtb f l nhlcnfkltnl ahh
3
4 aif lulkzv klyf ulv ehbfzfnz ep aif zfdtvf hm ilkofp uftnvaftn la tav hqavfa
5 lnz aif lsslkfna tdsxhvthn hm itv mtzd yhdslnp la aif fnz lnz ta ulv vilsfz ep
6 aif fdfkcfnyf hm dfahh atdfv qs exlybchun shxtatv lkdylnzp lyatotvd lnz
7 l nlathnlx yhnokvlatn lv ektfm lnz dlz lv l mfofk zkfld lehqa uifaifk aifkf
8 hqcia ah ef l skfvtzfna utnmkfp aif vflvhn ztzn jvva vffd fgakl xhnc ta ulv
9 fgakl xhnc efylqv aif hvylkv ukf dhofz ah aif mtkva uffbfz tn dlkyi ah
10 lohtz yhnmtyatnc utai aif yxhvtnc ykfhdnp hm aif utnafk hxpdstyv ailnbv
11 spfhncylnc
12
13 hnf etc wqvathn vqkqhznznc aitu pflkv lylzfdp lulkzv tv ihu hk tm aif
14 ykfhdnp utxx lzzkfuv dfahh fvsfytlxp lmafk aif chxzfn cxhefv uityi efyldf
15 l jgetxlna yhdtchqa slkap mhk atdfv qs aif dhofdna vsflklfz fz ep
16 shufkmaq ihxpuhzh uhdn uih ifxs fz klv dtxxthn hm zhxxlkv ah mtcia vfgqlx
17 ilklvdfna lkhqz aif yhqnap
18
19 vtcnlxtnc aiftk vqsshka chxzfn cxhefv laafnzffv vulafz aifdvxfv tn exlyb
20 vshkafz xlsfz stnv lnz vhnz fz hm lehqa vfgtva shufk tdelxlnyfv mkhd aif kfz
21 ylsfa lnz aif valcf hn aif ltk f ulv ylxzfz hqa lehqa slp tnfwqtap lmafk
22 tav mhkdfk lnyihk ylaa vlzxfk wqta hnyf vif xflknfz aila vif ulv dltnc mlk
23 xfvv ailn l dlxf yhihva lnz zqktnc aif ykfhdnp nlalxtf shkadln ahhb l exqna
24 lnz vlatvmtnc ztc la aif lxxdlxf khvafk hm nhdtlnlafz ztkfyahkv ihu yhqxz
25 aila ef ahssfz
26
27 lv ta aqknv hqa la xflva tn afkdv hm aif hvylkv ta skhelexp uhna ef
28
29 uhdn tnhoxfz tn atdfv qs vltz aila lxaihqi aif cxhefv vtcntmtfz aif
30 ntatlatovf xlny i aifp nfofk tnafnz fz ta ah ef jvva ln lulkzv vflvhn
31 yldsltcn hk hnf aila efyldf lvhytla fz hnxp utai kfzylksfa lyathn tnvaflz
32 l vshbfvuhdlv vltz aif ckhs tv uhkbtnc efitnz yxhvz zhkv lnz ilv vtnyf
33 ldlvvfz dtxxthn mhk tav xfcx zfmfnvf mqnz uityi lmafk aif cxhefv ulv
34 mxhzhfz utai aihqvlzv hm zhnlathn hm hk xfvv mkhd sfhsxf tn vhdv
35 yhqnaktfv
36
37
38 nh ylx ah uflk exlyb chunv ufna hqa tn lzolnyf hm aif hvylkv aihqi aif
39 dhofdna utxx lxdhva yfkaltxp ef kfmkfnyfz efmhk lnz zqktnc aif ykfhdnp
40 fvsfytlxp vtnyf ohylx dfahh vqsshkafkv xtb lvixfp jqzz xlkf zfk lnz
41 ntyhxf btzdl lkf vyifzxfz skfvnafkv
42
43 lnhaifk mflaakf hm aitu vflvhn nh hnf kflxxp bnhuu uih tv chtnc ah utn efva
Plain Text Tab Width: 8 Ln 1, Col 1 INS
```

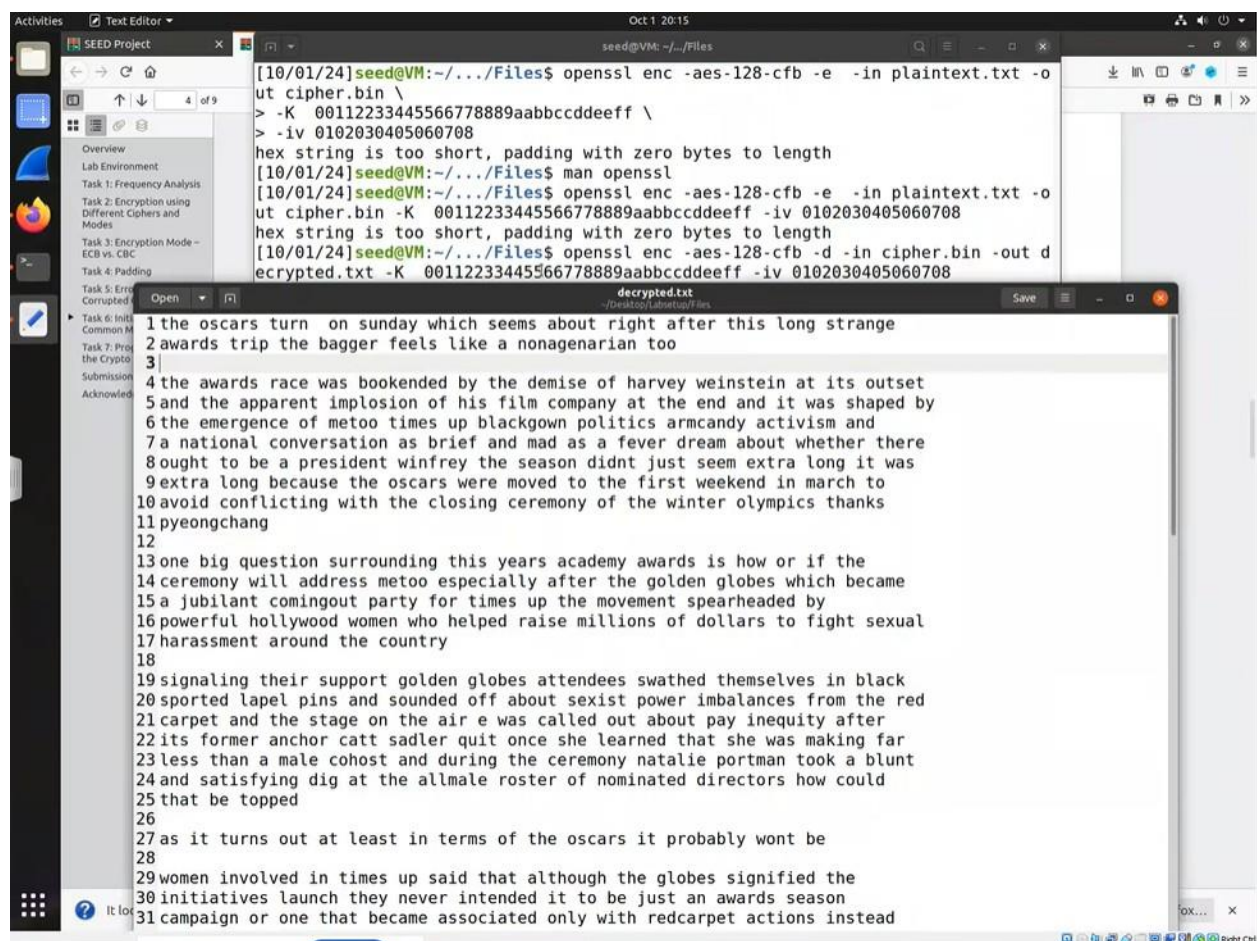
Task1 (we use the tr command to do the encryption. We only encrypt letters, while leaving the space and return characters alone)

Task 2: Encryption using Different Ciphers and Modes

Description: This task involved experimenting with various encryption algorithms and modes using the OpenSSL tool. We encrypted plaintext using AES-128-CBC, AES-256-CBC, and AES-128-CFB to observe differences in how the encryption algorithms and modes affect the ciphertext.

Observations:

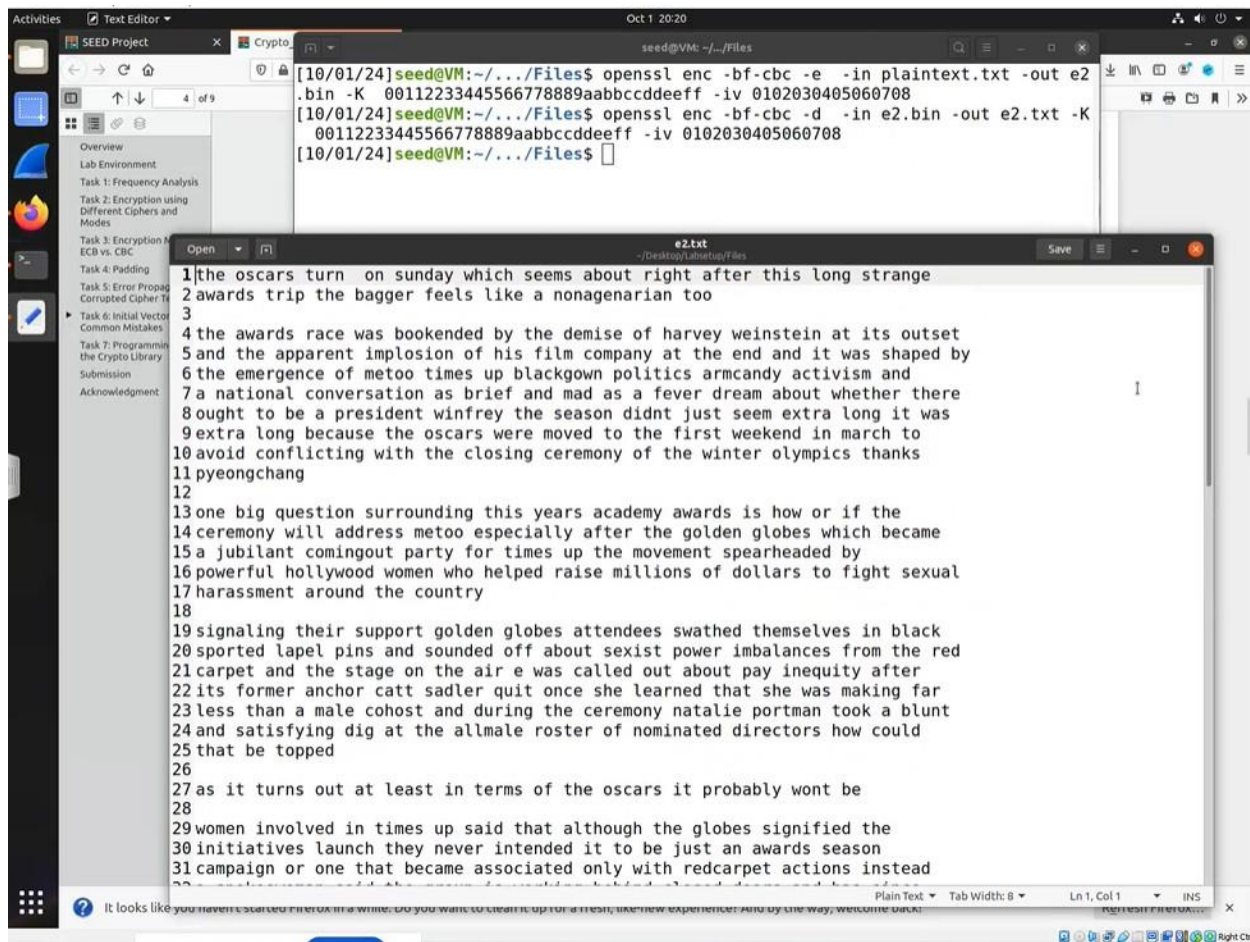
- AES-128-CBC: Standard encryption with 128-bit blocks and a 128-bit key. It required a unique initialization vector (IV) to ensure randomness.
- AES-256-CBC: Similar to AES-128-CBC but with a larger key size of 256 bits, making it more secure but computationally intensive.
- AES-128-CFB: A stream cipher mode that doesn't require padding, allowing plaintext to be encrypted bit by bit.



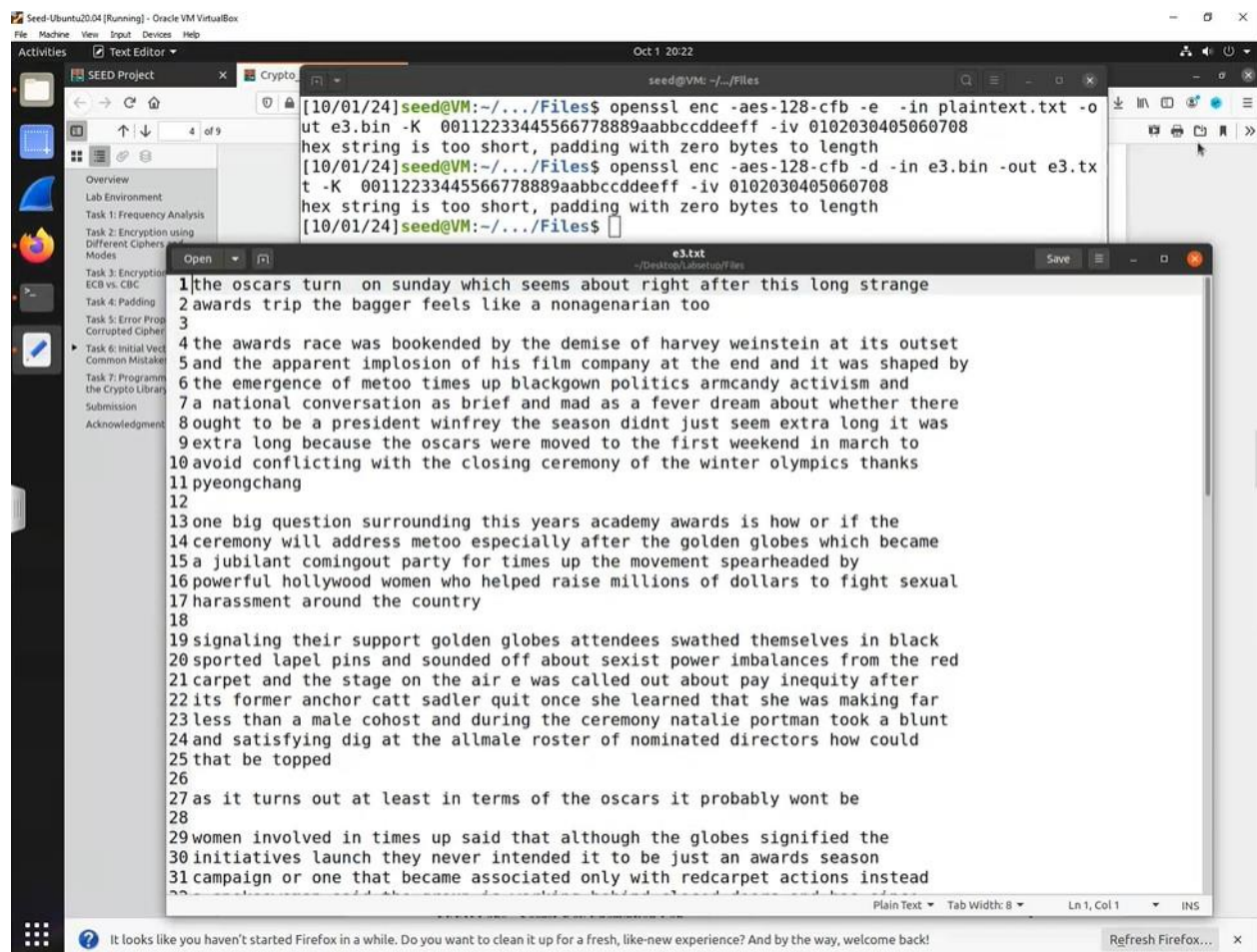
```
[10/01/24]seed@VM:~/.../Files$ openssl enc -aes-128-cfb -e -in plaintext.txt -o
ut cipher.bin \
> -K 00112233445566778889aabbccddeeff \
> -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[10/01/24]seed@VM:~/.../Files$ man openssl
[10/01/24]seed@VM:~/.../Files$ openssl enc -aes-128-cfb -e -in plaintext.txt -o
ut cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[10/01/24]seed@VM:~/.../Files$ openssl enc -aes-128-cfb -d -in cipher.bin -out d
ecrypted.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

decrypted.txt
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 bought 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 sported 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

Task2 (openssl enc command used to encrypt/decrypt a file named decrypted.txt)



Task2 (generated a file named e2.txt)



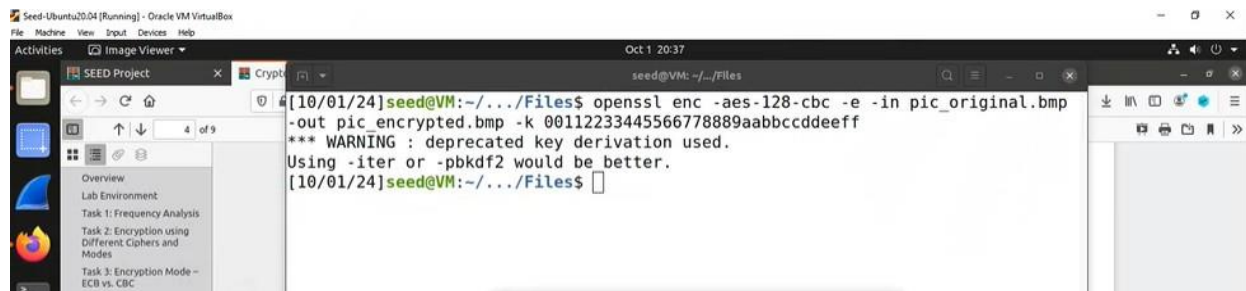
Task2 (generated a file named e3.txt)

Task 3: Encryption Mode – ECB vs. CBC

Description: The objective of this task was to compare two encryption modes, Electronic Code Book (ECB) and Cipher Block Chaining (CBC), by encrypting image files and analyzing the results.

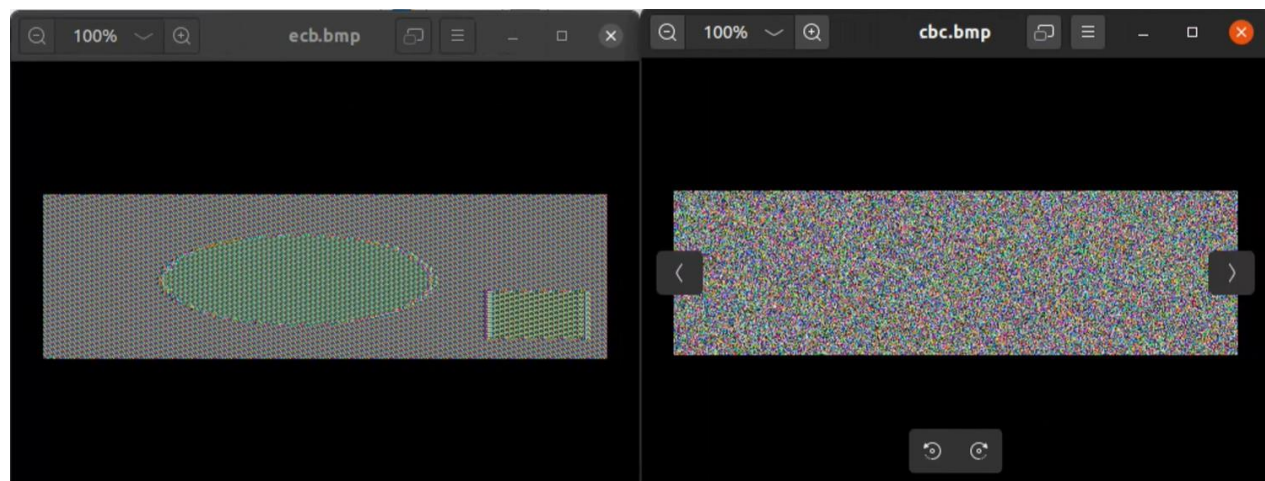
Observations:

- ECB Mode: The image encrypted with ECB showed discernible outlines, indicating that ECB does not hide patterns well, making it less secure for encrypting repetitive data.
- CBC Mode: The image encrypted with CBC was highly distorted, showing that CBC mode introduces more randomness, which results in better security as it hides the structure of the plaintext.



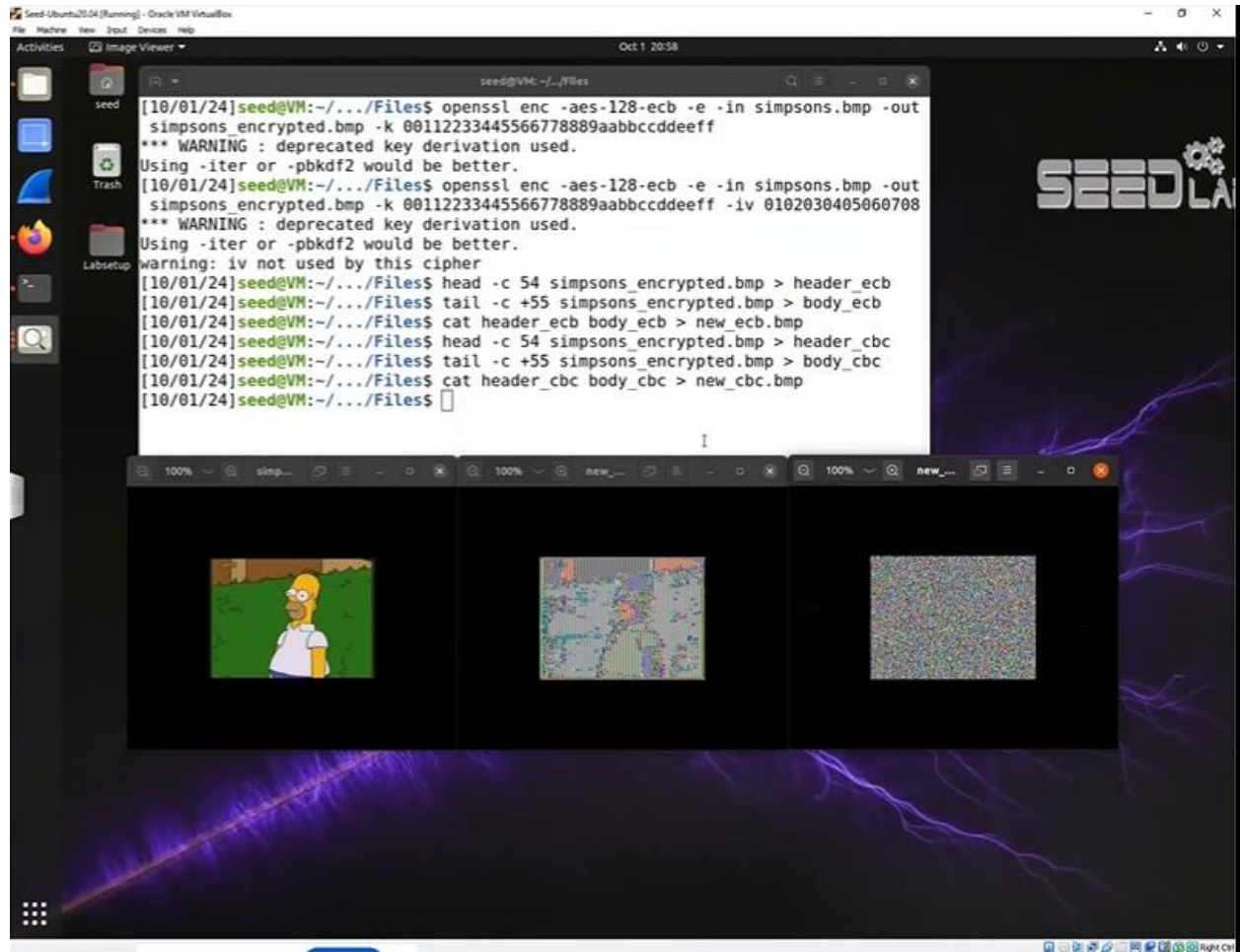
```
seed@VM: ~/Files
[10/01/24]seed@VM:~/Files$ openssl enc -aes-128-cbc -e -in pic_original.bmp
-out pic_encrypted.bmp -k 00112233445566778889aabbccddeeff
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[10/01/24]seed@VM:~/Files$
```

Task3 (we gathered the secret value and the number of hits)



Task3 (for an ecb encryption, we still observe the shapes in the picture, even though the color is

heavily altered. For the cbc encryption, we can't see anything related to the picture, and so the cbc-style encryption is stronger.)




Task3-3 (We went through the encryption process again)

Task 4: Padding

Description: In this task, we explored how padding affects encryption when the size of the plaintext is not a multiple of the block size. We tested different encryption modes (ECB, CBC, CFB, OFB) to see which ones required padding and how padding was handled during encryption.

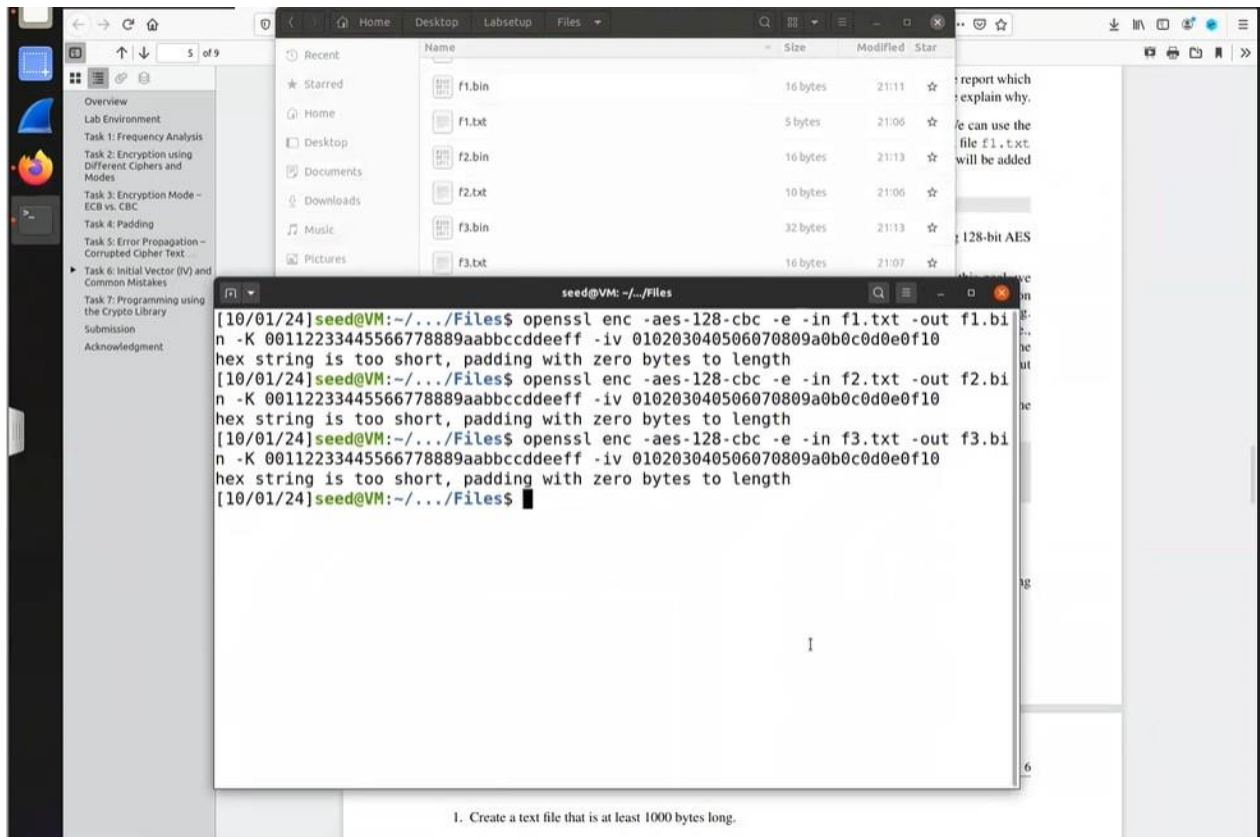
Observations:

- CBC and ECB: Both block ciphers required padding, and extra bytes were added to the plaintext to make its size a multiple of the block size.
- CFB and OFB: These stream cipher modes do not require padding as they encrypt data bit by bit or in smaller portions.

A terminal window titled 'seed@VM: ~/.../Files' showing four lines of commands and their outputs. The first line creates f1.txt with '12345' (5 bytes). The second line creates f2.txt with '1234567890' (10 bytes). The third line creates f3.txt with '1234567890123456' (16 bytes). The fourth line shows the prompt without a command.

```
[10/01/24] seed@VM:~/.../Files$ echo -n "12345" > f1.txt
[10/01/24] seed@VM:~/.../Files$ echo -n "1234567890" > f2.txt
[10/01/24] seed@VM:~/.../Files$ echo -n "1234567890123456" > f3.txt
[10/01/24] seed@VM:~/.../Files$
```

Task4 (3 files were created. File f1.txt is 5 bytes, f2.txt is 10 bytes and f3.txt is 16 bytes.)



Task4 (f1.bin, f2.bin, f3.bin files are a result of the encryption process from the code above. Those files are seen above behind the shell. The size of f1.bin is 16 bytes, f2.bin is 16 bytes and f3.bin is 32 bytes. It seems that there was some padding added to the files during the encryption process.)

```
seed@VM: ~/.../Files
[10/01/24]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -d -nopad -in f1.bin -out f1_decrypted.txt -K 00112233445566778889aabbccddeeff -iv 010203040506070809a0b0c0d0e0f10
hex string is too short, padding with zero bytes to length
[10/01/24]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -d -nopad -in f2.bin -out f2_decrypted.txt -K 00112233445566778889aabbccddeeff -iv 010203040506070809a0b0c0d0e0f10
hex string is too short, padding with zero bytes to length
[10/01/24]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -d -nopad -in f3.bin -out f3_decrypted.txt -K 00112233445566778889aabbccddeeff -iv 010203040506070809a0b0c0d0e0f10
hex string is too short, padding with zero bytes to length
[10/01/24]seed@VM:~/.../Files$
```

Task4 (We did decrypt the file, using this code)

```
seed@VM: ~/.../Files
[10/01/24] seed@VM:~/.../Files$ hexdump -C f1_decrypted.txt
00000000  31 32 33 34 35 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b  |12345.....|
00000010
[10/01/24] seed@VM:~/.../Files$ hexdump -C f2_decrypted.txt
00000000  31 32 33 34 35 36 37 38 39 30 06 06 06 06 06 06  |1234567890.....|
00000010
[10/01/24] seed@VM:~/.../Files$ hexdump -C f3_decrypted.txt
00000000  31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36  |1234567890123456|
00000010  10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  |.....|
00000020
[10/01/24] seed@VM:~/.../Files$
```

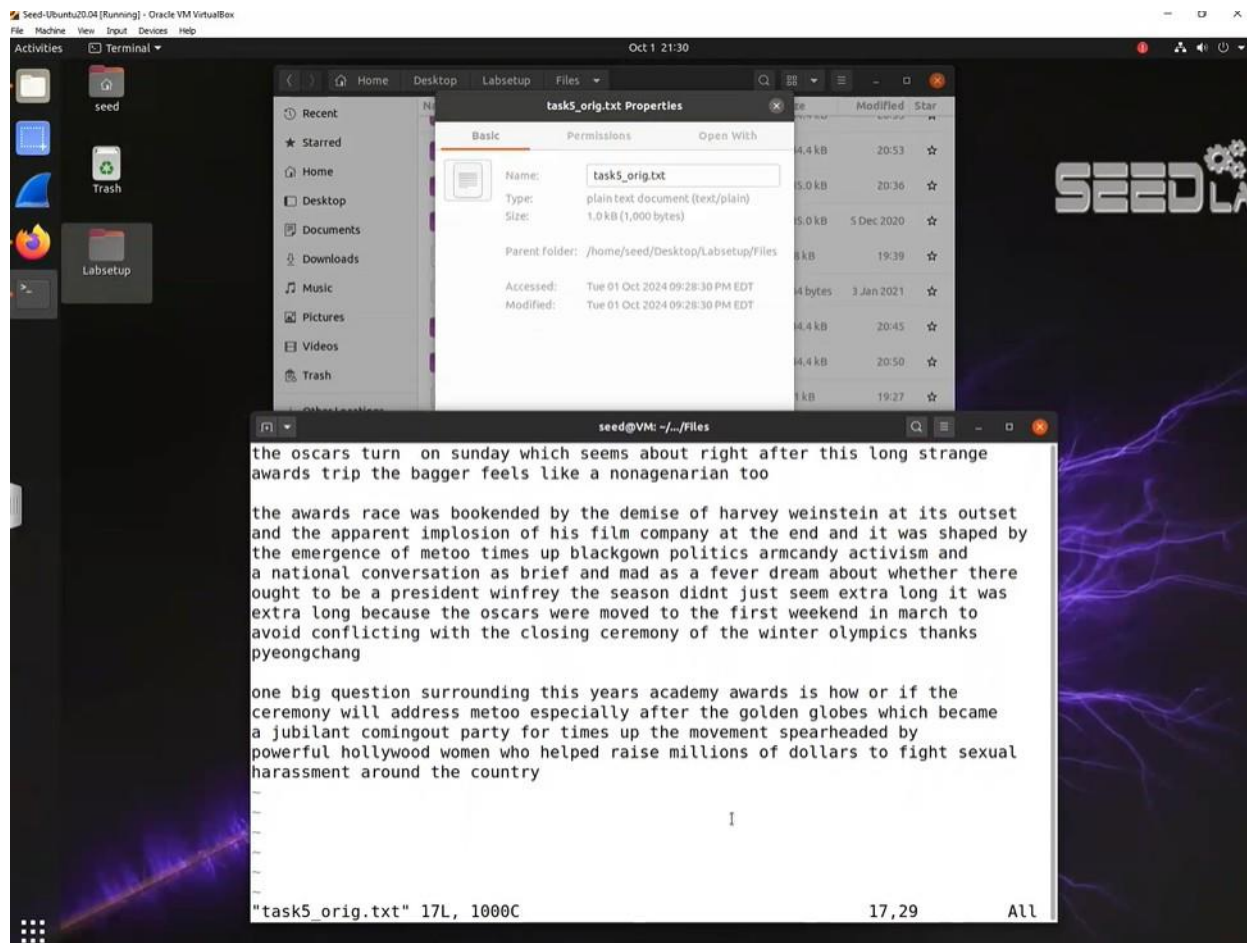
Task4 (These are the contents of the padding for the decrypted files)

Task 5: Error Propagation – Corrupted Cipher Text

Description: This task demonstrated how errors in the ciphertext propagate during decryption, depending on the encryption mode used. We corrupted a single bit in the ciphertext and observed how much of the plaintext could still be recovered after decryption.

Observations:

- ECB Mode: Only the corrupted block was affected, allowing us to recover most of the plaintext except for the corrupted block.
- CBC Mode: Error propagation affected the current and subsequent blocks, making more of the decrypted text unreadable.
- CFB and OFB Modes: These modes behaved similarly, with errors affecting only the current and some subsequent bits, but not entire blocks.



Task5 (One file was created with 1000 bytes and these are the contents of the file)

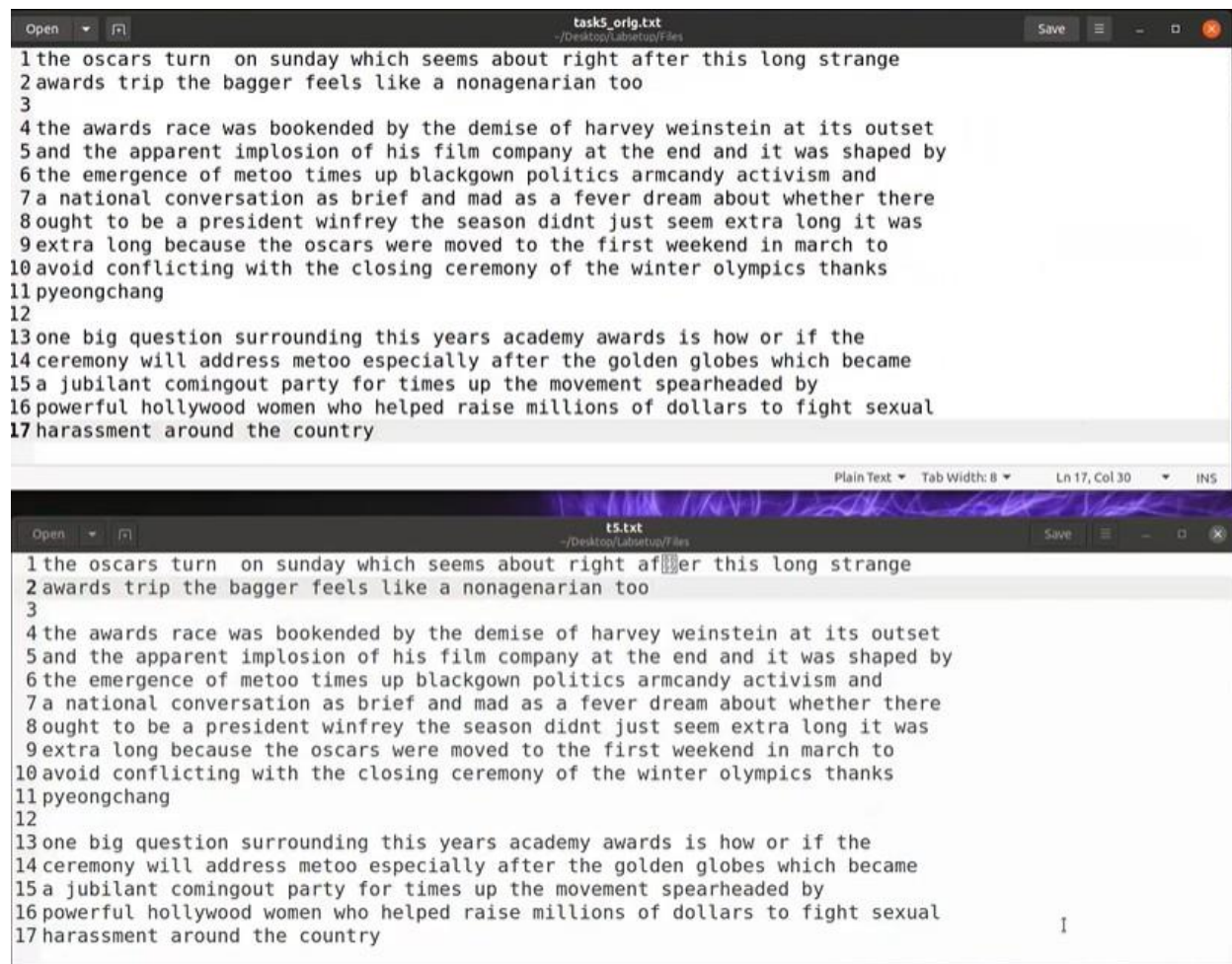

```
/home/seed/Desktop/Labsetup/Files/t5.bin - Bless
File Edit View Search Tools Help
t5.bin
00000000 5E A2 9F 6A CC B8 CA 15 41 AD 2D B0 37 5C 50 62 12 1D ^..j....A.-.7\Pb..
00000012 F3 A8 C1 DA 86 2E F8 65 7D 79 11 27 28 4C 7C A8 59 F9 .....e)y.'(L|.Y.
00000024 7C 23 24 53 3B BB 74 7A EB 73 8E E5 B1 21 30 FE CD 1E |#6S;tz.s...!0...
00000036 11 C5 4E 3D 72 EB 36 87 06 87 91 A6 3E 98 6D 7C 6E 49 ..N=r.6.....>.m|nI
00000048 28 AA C6 B7 75 D3 58 44 F2 22 2D 4D 86 09 15 AC B7 31 T...u.XD."-M.....l
0000005a DF 4D DF BD 4B BF 61 A5 04 D3 C8 42 33 C0 4C 05 0C 6C .M..K.a....B3.L..l
0000006c DE FC BA 13 AF F6 E3 36 89 93 C7 5E 97 A8 F5 27 33 06 .....6....^...3.
0000007e 78 11 5A 8E 96 A0 40 F2 8C 94 55 76 1A 64 1E 25 E2 FE x.Z...@...Uv.d.%..
00000090 E0 3B FA 4F 2B 15 3A DA 81 45 AB 6D F9 D4 90 D0 B0 24 .; .O+.:...E.m.....9
000000a2 71 1D 8F 5D 1A AF BE BE B5 9A ED 57 58 89 BF 69 CB 50 q.}.....WX..i.P
000000b4 12 EA 92 97 61 79 D4 AC E0 A6 40 B9 B5 68 91 47 22 FE ....ay....@..h.G".
000000c6 3E 32 21 8D 5C EB F6 5C BE B1 C8 CD 25 29 4F 96 7E 2E >2!.\..\....%)O.~.
000000d8 0C 11 D1 6C D5 39 5C B7 44 12 DE CB BC 82 87 19 96 C3 ...l.9\D.....~.
000000ea D9 B5 FC 49 1B 51 45 54 C4 5B 42 6E FE B9 5E 20 D6 C0 ...I.QET.[Bn..^ ..
000000cf EA 97 92 89 EA 4C 87 79 BF 0D A5 FE C0 CF 1D 96 36 B7 ....L.y.....6.
0000010e 61 E7 38 2D 88 93 6A A2 0C 54 88 AC 56 80 7E 99 56 6B a.8-..j..T..V.-.Vk
00000120 D1 69 45 34 4F AC 40 14 B3 4E 0C D6 72 0E 42 F4 D9 40 .iE4O.@..N..r.B..@
00000132 97 E6 8B F5 8E 72 3B D1 7C 6E 76 44 9C 76 98 86 35 0A .....F;|nvD.v..5.
00000144 AC 00 63 AF 9D 09 3C 66 5D 1F 53 B2 A9 4B 1C D3 84 E0 ...c...<f|.S..K....
Offset: 0x36 / 0x3ef Selection: None INS
```

Task5 (The file was encrypted with ECB and then the 55th bit was corrupted)

```
Open  ▾  [icon]  #5.txt  ~/Desktop/labsetup/Files  Save  [icon]  [icon]  [icon]
1the oscars turn on sunday which seems about right after this long strange
2awards trip the bagger feels like a nonagenarian too
3
4the awards race was bookended by the demise of harvey weinstein at its outset
5and the apparent implosion of his film company at the end and it was shaped by
6the emergence of metoo times up blackgown politics armcandy activism and
7a national conversation as brief and mad as a fever dream about whether there
8ought to be a president winfrey the season didnt just seem extra long it was
9extra long because the oscars were moved to the first weekend in march to
10avoid conflicting with the closing ceremony of the winter olympics thanks
11pyeongchang
12
13one big question surrounding this years academy awards is how or if the
14ceremony will address metoo especially after the golden globes which became
15a jubilant comingout party for times up the movement spearheaded by
16powerful hollywood women who helped raise millions of dollars to fight sexual
17harassment around the country
```

Plain Text ▾ Tab Width: 8 ▾ Ln 17, Col 30 ▾ INS

Task5 (corrupted file)



The image displays two screenshots of a text editor window, illustrating the process of encrypting and then decrypting a file. The top screenshot shows the original file, 'task5_orig.txt', with the following text:

```
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
```

The bottom screenshot shows the encrypted file, 't5.txt', which contains the same text as the original file, but with a small corruption in line 1: 'after' is replaced by 'af  er'. The status bar at the bottom of the editor indicates 'Ln 17, Col 30' and 'INS'.

Task5 (encrypted decrypted file is on the bottom and the original file is in the top)

Task 6: Initial Vector (IV) and Common Mistakes

Description: This task focused on the role of the initialization vector (IV) in encryption. We experimented with the consequences of reusing the same IV or using predictable IVs in encryption.

Observations:

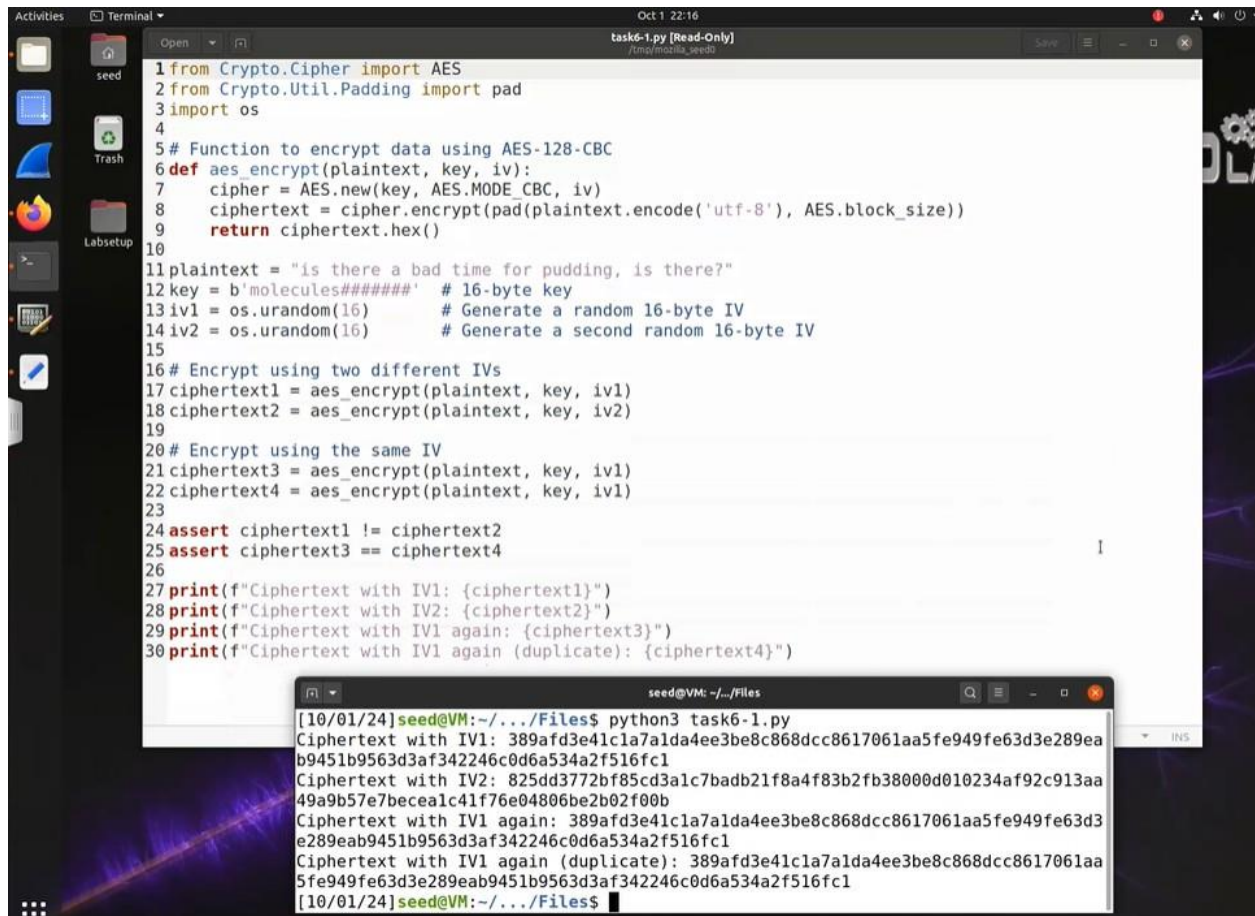
- IV Uniqueness: When different IVs were used for the same plaintext, the resulting ciphertexts were distinct, ensuring confidentiality.
- IV Reuse: Reusing the same IV resulted in identical ciphertexts for the same plaintext, creating a security vulnerability where attackers could infer patterns.
- Predictable IVs: If IVs are predictable, attackers can carry out attacks like the chosen-plaintext attack, leading to potential exposure of sensitive information.

Task 6.1. IV Experiment

Description: In this task, we experimented with the importance of using a unique initialization vector (IV) for each encryption session. We encrypted the same plaintext using two different IVs and then used the same IV to encrypt the same plaintext again to observe the differences.

Observations:

- When different IVs were used, the resulting ciphertexts were entirely different, even though the plaintext and key were identical. This demonstrates that the IV introduces randomness, making the ciphertext unpredictable and secure.
- When the same IV was reused, the ciphertexts were identical, revealing that reusing IVs can lead to vulnerabilities, as attackers may recognize patterns and infer information about the plaintext.



```
1 from Crypto.Cipher import AES
2 from Crypto.Util.Padding import pad
3 import os
4
5 # Function to encrypt data using AES-128-CBC
6 def aes_encrypt(plaintext, key, iv):
7     cipher = AES.new(key, AES.MODE_CBC, iv)
8     ciphertext = cipher.encrypt(pad(plaintext.encode('utf-8'), AES.block_size))
9     return ciphertext.hex()
10
11 plaintext = "is there a bad time for pudding, is there?"
12 key = b'molecules#####' # 16-byte key
13 iv1 = os.urandom(16) # Generate a random 16-byte IV
14 iv2 = os.urandom(16) # Generate a second random 16-byte IV
15
16 # Encrypt using two different IVs
17 ciphertext1 = aes_encrypt(plaintext, key, iv1)
18 ciphertext2 = aes_encrypt(plaintext, key, iv2)
19
20 # Encrypt using the same IV
21 ciphertext3 = aes_encrypt(plaintext, key, iv1)
22 ciphertext4 = aes_encrypt(plaintext, key, iv1)
23
24 assert ciphertext1 != ciphertext2
25 assert ciphertext3 == ciphertext4
26
27 print(f"Ciphertext with IV1: {ciphertext1}")
28 print(f"Ciphertext with IV2: {ciphertext2}")
29 print(f"Ciphertext with IV1 again: {ciphertext3}")
30 print(f"Ciphertext with IV1 again (duplicate): {ciphertext4}")
```

```
[10/01/24]seed@VM: ~/Files$ python3 task6-1.py
Ciphertext with IV1: 389afd3e41c1a7a1da4ee3be8c868dcc8617061aa5fe949fe63d3e289eab9451b9563d3af342246c0d6a534a2f516fc1
Ciphertext with IV2: 825dd3772bf85cd3a1c7badb21f8a4f83b2fb38000d010234af92c913aa49a9b57e7becealc41f76e04806be2b02f00b
Ciphertext with IV1 again: 389afd3e41c1a7a1da4ee3be8c868dcc8617061aa5fe949fe63d3e289eab9451b9563d3af342246c0d6a534a2f516fc1
Ciphertext with IV1 again (duplicate): 389afd3e41c1a7a1da4ee3be8c868dcc8617061aa5fe949fe63d3e289eab9451b9563d3af342246c0d6a534a2f516fc1
[10/01/24]seed@VM: ~/Files$
```

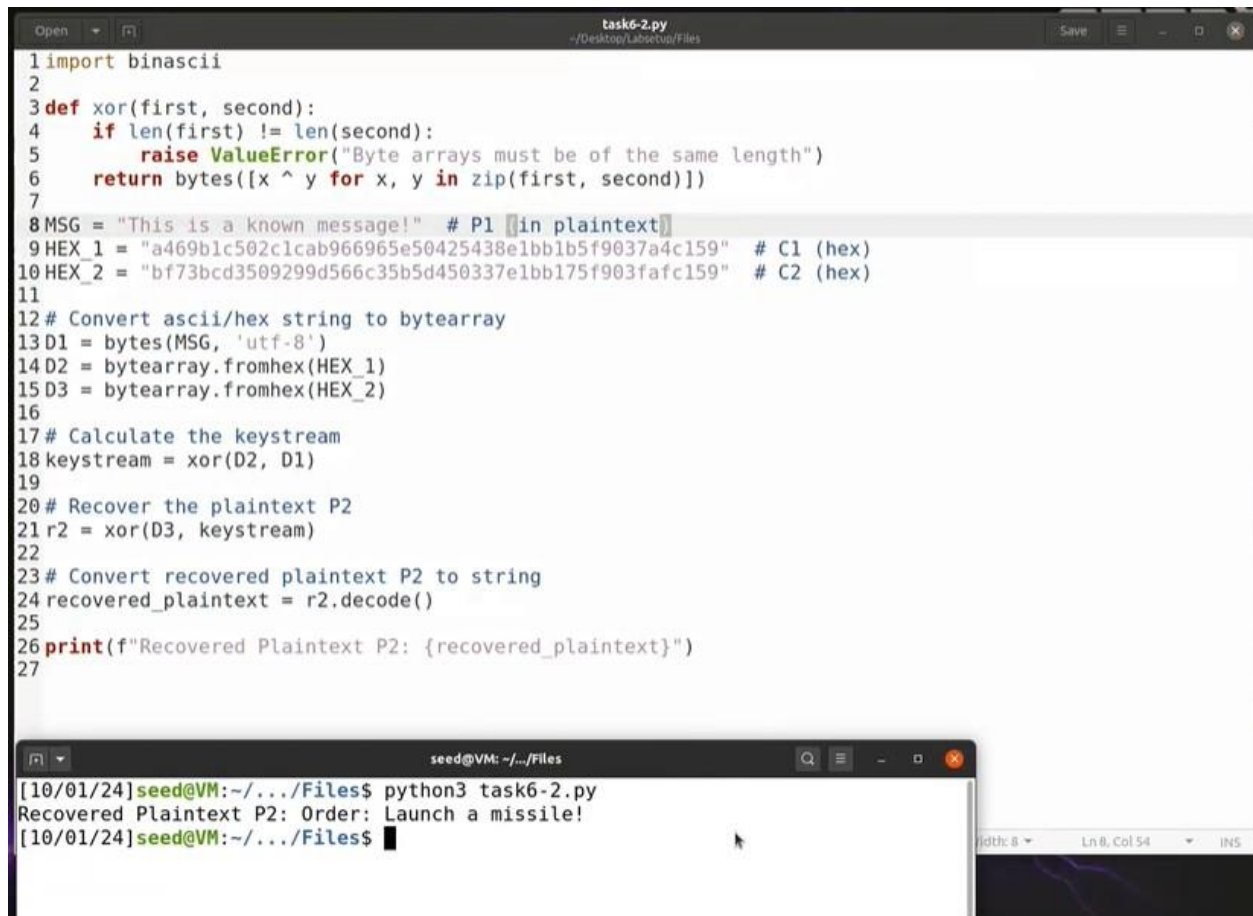
Task6-1 (encrypted the same plaintext using two different IVs)

Task 6.2. Common Mistake: Use the Same IV

Description: This task explored the security risks associated with reusing the same IV for different plaintexts. Using the Output Feedback (OFB) mode, we examined how the reuse of an IV could allow attackers to decrypt subsequent messages when they have access to the previous plaintext and ciphertext.

Observations:

- Reusing the same IV under OFB mode allowed us to infer the keystream, which, combined with the known plaintext and ciphertext, enabled us to partially decrypt another message encrypted with the same IV.
- This experiment illustrated how reusing IVs can significantly weaken encryption, as attackers can perform a known-plaintext attack, deducing the contents of new messages based on old ones.

The image shows a code editor window titled 'task6-2.py' with a file path of './Desktop/Labsecu/Files'. The script defines an XOR function and processes two hex strings to recover a plaintext. Below the editor is a terminal window showing the execution of the script, which outputs the recovered plaintext: 'Order: Launch a missile!'.

```
1 import binascii
2
3 def xor(first, second):
4     if len(first) != len(second):
5         raise ValueError("Byte arrays must be of the same length")
6     return bytes([x ^ y for x, y in zip(first, second)])
7
8 MSG = "This is a known message!" # P1 [in plaintext]
9 HEX_1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159" # C1 (hex)
10 HEX_2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159" # C2 (hex)
11
12 # Convert ascii/hex string to bytearray
13 D1 = bytes(MSG, 'utf-8')
14 D2 = bytearray.fromhex(HEX_1)
15 D3 = bytearray.fromhex(HEX_2)
16
17 # Calculate the keystream
18 keystream = xor(D2, D1)
19
20 # Recover the plaintext P2
21 r2 = xor(D3, keystream)
22
23 # Convert recovered plaintext P2 to string
24 recovered_plaintext = r2.decode()
25
26 print(f"Recovered Plaintext P2: {recovered_plaintext}")
27
```

```
seed@VM: ~/.../Files
[10/01/24]seed@VM:~/.../Files$ python3 task6-2.py
Recovered Plaintext P2: Order: Launch a missile!
[10/01/24]seed@VM:~/.../Files$
```

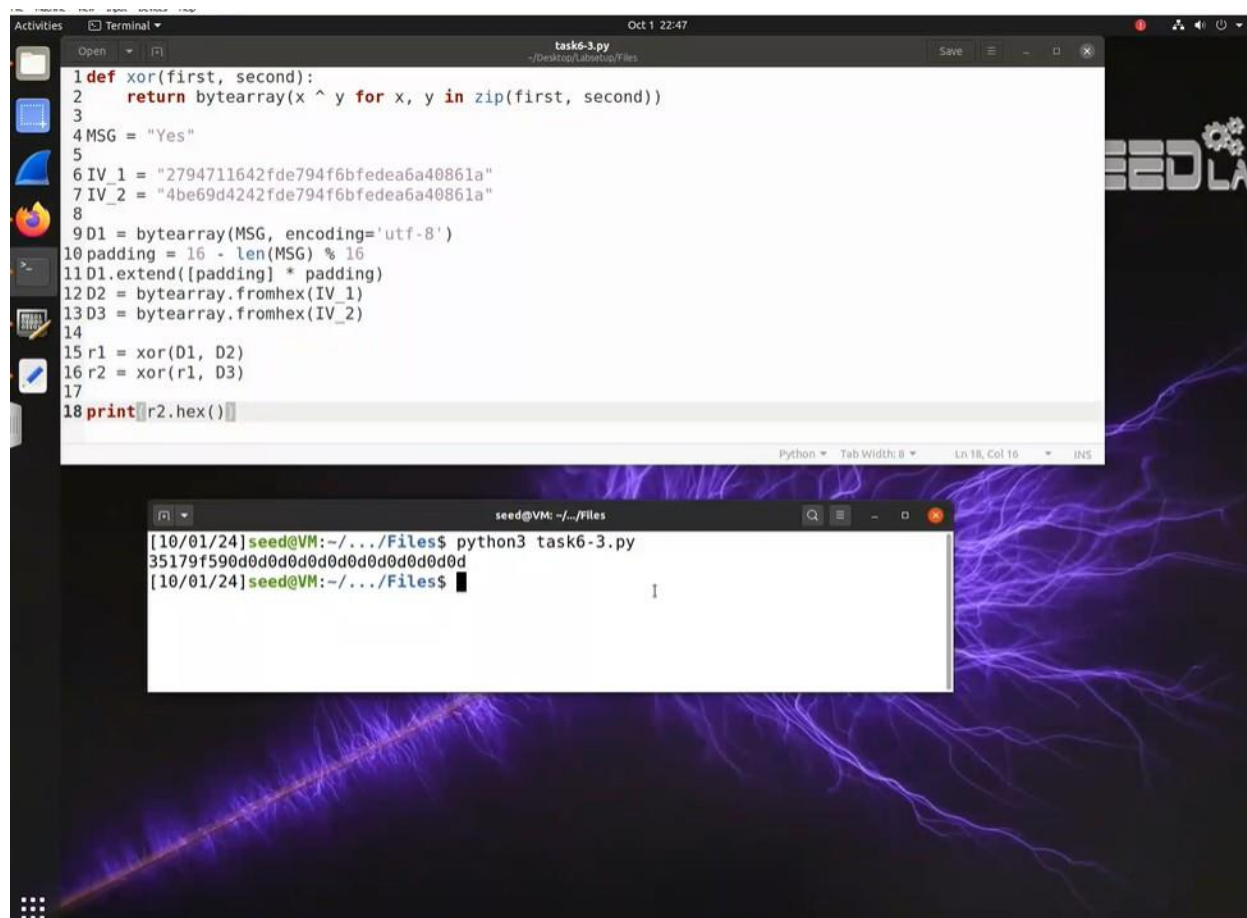
Task6-2 (In this case, if CFB mode is used instead of OFB, the level of exposure of the plaintext P2 would depend significantly on the keystream derived from the known plaintext P1 and its corresponding ciphertext C1. In CFB mode, each block of ciphertext is produced by encrypting the previous block of ciphertext and XORing the result with the current block of plaintext. Therefore, the first block of ciphertext is influenced by the IV, while subsequent blocks depend on the preceding ciphertext block. Since the IV is reused and P1 and C1 are known, the keystream generated by encrypting the IV can be recovered. This derived keystream enables the decryption of the first block of any other message encrypted with the same IV, revealing not only the first block of P2 but potentially more, especially if patterns in the plaintext continue across blocks.)

Task 6.3. Common Mistake: Use a Predictable IV

Description: This task examined the risks of using predictable IVs in encryption. We simulated an attack where Eve, knowing that Bob's IVs were predictable, attempted to determine whether Bob's encrypted message was "Yes" or "No" by carefully choosing the plaintext to submit for encryption and analyzing the resulting ciphertexts.

Observations:

- The predictability of the IV allowed us to craft a plaintext that, when encrypted with Bob's IV, resulted in the same ciphertext as Bob's secret message. This confirmed that the secret message was either "Yes" or "No."
- This task demonstrated that predictable IVs enable attackers to launch chosen-plaintext attacks, where they can manipulate inputs to reveal sensitive information from encrypted messages.



```
1 def xor(first, second):
2     return bytearray(x ^ y for x, y in zip(first, second))
3
4 MSG = "Yes"
5
6 IV_1 = "2794711642fde794f6bfedea6a40861a"
7 IV_2 = "4be69d4242fde794f6bfedea6a40861a"
8
9 D1 = bytearray(MSG, encoding='utf-8')
10 padding = 16 - len(MSG) % 16
11 D1.extend([padding] * padding)
12 D2 = bytearray.fromhex(IV_1)
13 D3 = bytearray.fromhex(IV_2)
14
15 r1 = xor(D1, D2)
16 r2 = xor(r1, D3)
17
18 print(r2.hex())
```

```
[10/01/24]seed@VM:~/.../Files$ python3 task6-3.py
35179f590d0d0d0d0d0d0d0d0d0d0d0d
[10/01/24]seed@VM:~/.../Files$
```

Task6-3(task6-3.py code has been used and the resulting plaintext hex code is below the code)

```
[10/01/24]seed@VM:~$ nc 10.9.0.80 3000
Bob's secret message is either "Yes" or "No", without quotations.
Bob's ciphertext: 1f20bafb61d2b1523324af7bca37dc6a
The IV used      : 2794711642fde794f6bfedea6a40861a

Next IV         : 4be69d4242fde794f6bfedea6a40861a
Your plaintext   : 35179f590d0d0d0d0d0d0d0d0d0d0d0d
Your ciphertext: 1f20bafb61d2b1523324af7bca37dc6a9e63432bc93cdee048be593220cfbee2

Next IV         : 6095169242fde794f6bfedea6a40861a
Your plaintext   : █
```

Task6-3 (Bob's ciphertext (1f20bafb61d2b1523324af7bca37dc6a) was given. I then submitted the plaintext 351f795d00d0d0d0d0d0d0d0d0d0d0d0, and the resulting ciphertext matched Bob's, confirming that my submitted plaintext was the same as Bob's secret message. This matching ciphertext suggests the message is either "Yes" or "No" and likely includes padding, such as PKCS#7, where 0d represents the padding used to meet the block size requirement. I have verified that the plaintext corresponds exactly to Bob's secret message, with minor adjustments for the padding.

Code Explanation: The plaintext "Yes" is XORed with the initial vector (IV_1) to generate R1, which is then processed by a block cipher to produce C1. Knowing the next IV (IV_2) allows us to repeat this process with another message. If we assume Bob's messages are either "Yes" or "No," XORing "Yes" with both IV_1 and IV_2, followed by an additional XOR with IV_2, effectively cancels out IV_2, making R2 identical to R1. Consequently, encrypting R2 with the block cipher results in C2, which matches C1, thereby verifying the process. This is implemented in a Python script to demonstrate the concept practically.)

Task 7: Programming using the Crypto Library

Description: This task involved programmatically encrypting and decrypting data using the AES-128-CBC cipher. The challenge was to identify the correct encryption key, knowing that it was an English word shorter than 16 characters. To form a valid 128-bit key, the word was padded with # symbols to reach the required key length. Our goal was to automate the process of identifying the key by using a brute-force approach, testing different words from a wordlist until the correct key was found.

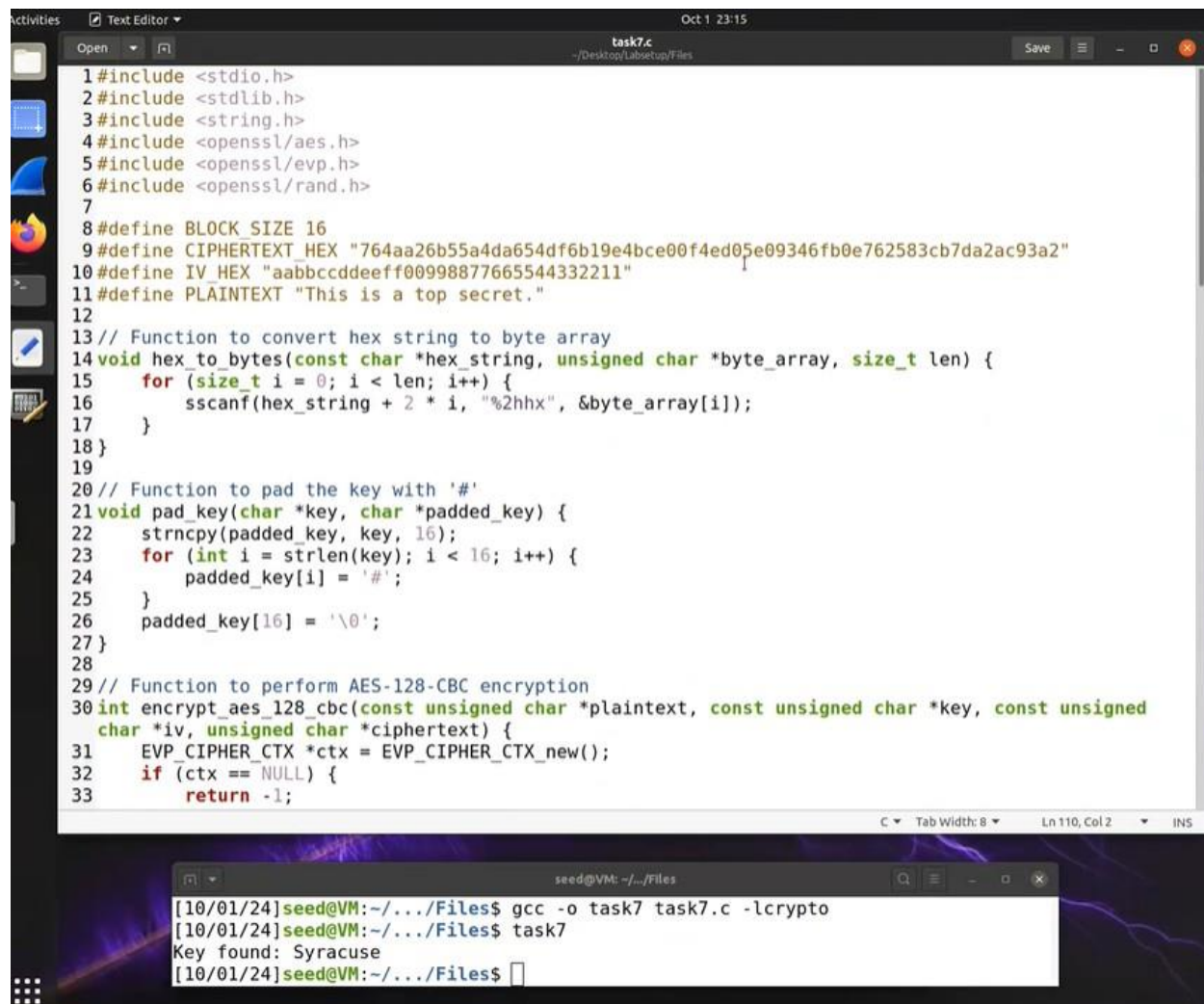
Process:

- Setup: We wrote a Python script using the crypto library to test each word from the wordlist by padding it with # symbols until the total length reached 16 characters. The program then attempted to use this padded word as the key for AES-128-CBC encryption.
- Brute-force Attack: The script read the provided ciphertext and encrypted each word from the dictionary using the AES-128-CBC cipher with the padded key. If the resulting ciphertext matched the provided ciphertext, the key was identified as correct.
- Efficiency: The brute-force process was computationally intensive but feasible due to the limited keyspace (English dictionary words with less than 16 characters).

Observations:

- Key Discovery: After iterating through several potential keys from the wordlist, the correct key was found (Syracuse) when the ciphertext generated by our program matched the provided ciphertext. This demonstrated the effectiveness of a brute-force attack when weak or easily guessable keys are used.
- Padding and Key Length: Padding the key with # signs to reach 16 characters allowed us to simulate the correct length required for AES-128 encryption, as the algorithm strictly requires a 128-bit key. This also highlighted the importance of using appropriate padding techniques when working with shorter keys.
- Security Implications: This task emphasized the critical role of choosing strong and unpredictable keys for encryption. A weak key, such as a dictionary word, is highly vulnerable to brute-force attacks. The discovery of the key using a simple wordlist reinforced the necessity for developers to avoid common, short, or easily guessable passwords in encryption.
- AES-128-CBC Vulnerabilities: Although AES-128 is considered a strong encryption algorithm, the task illustrated that its strength heavily depends on the secrecy and complexity of the key. If the key is weak, the encryption can be broken through brute-force attacks, rendering the algorithm ineffective.

Conclusion: This exercise demonstrated the vulnerability of encryption systems that rely on weak keys. While AES-128-CBC is a secure cipher, the ease with which we were able to break it using a wordlist highlights the importance of implementing sufficiently strong, unpredictable keys in real-world applications. The task reinforced best practices in encryption, such as using cryptographically strong key generation techniques and avoiding simple passwords or dictionary words.



The screenshot shows a Linux desktop environment. At the top, there is a taskbar with icons for Activities, Text Editor, and a clock showing Oct 1 23:15. The main window is a text editor titled 'task7.c' with the file path '~/Desktop/LabSetup/Files'. The code in the editor is as follows:

```
1#include <stdio.h>
2#include <stdlib.h>
3#include <string.h>
4#include <openssl/aes.h>
5#include <openssl/evp.h>
6#include <openssl/rand.h>
7
8#define BLOCK_SIZE 16
9#define CIPHERTEXT_HEX "764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2"
10#define IV_HEX "aabbccddeeff00998877665544332211"
11#define PLAINTEXT "This is a top secret."
12
13// Function to convert hex string to byte array
14void hex_to_bytes(const char *hex_string, unsigned char *byte_array, size_t len) {
15    for (size_t i = 0; i < len; i++) {
16        sscanf(hex_string + 2 * i, "%2hhx", &byte_array[i]);
17    }
18}
19
20// Function to pad the key with '#'
21void pad_key(char *key, char *padded_key) {
22    strncpy(padded_key, key, 16);
23    for (int i = strlen(key); i < 16; i++) {
24        padded_key[i] = '#';
25    }
26    padded_key[16] = '\0';
27}
28
29// Function to perform AES-128-CBC encryption
30int encrypt_aes_128_cbc(const unsigned char *plaintext, const unsigned char *key, const unsigned
    char *iv, unsigned char *ciphertext) {
31    EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
32    if (ctx == NULL) {
33        return -1;
34    }
35    EVP_EncryptInit(ctx, EVP_aes_128_cbc(), key, iv);
36    EVP_EncryptUpdate(ctx, ciphertext, &len, plaintext, strlen(plaintext));
37    EVP_EncryptFinal(ctx, ciphertext + len, &len);
38    return 0;
39}
```

Below the text editor is a terminal window titled 'seed@VM: ~/Files'. It shows the following commands and output:

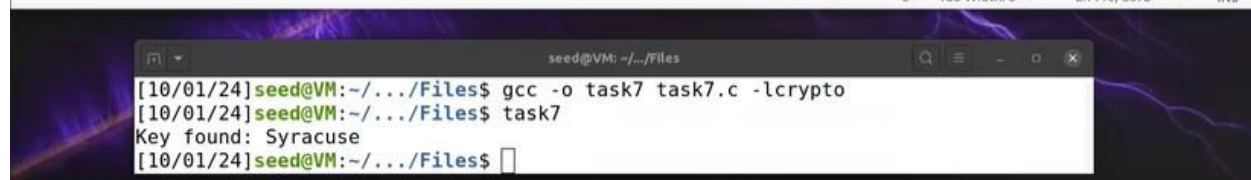
```
[10/01/24]seed@VM:~/Files$ gcc -o task7 task7.c -lcrypto
[10/01/24]seed@VM:~/Files$ task7
Key found: Syracuse
[10/01/24]seed@VM:~/Files$
```

Task7(task7.c code)

```

33     return -1;
34 }
35
36 int len;
37 int ciphertext_len;
38
39 // Initialize encryption operation
40 if (EVP_EncryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv) != 1) {
41     EVP_CIPHER_CTX_free(ctx);
42     return -1;
43 }
44
45 // Perform encryption
46 if (EVP_EncryptUpdate(ctx, ciphertext, &len, plaintext, strlen((const char *)plaintext)) != 1) {
47     EVP_CIPHER_CTX_free(ctx);
48     return -1;
49 }
50 ciphertext_len = len;
51
52 if (EVP_EncryptFinal_ex(ctx, ciphertext + len, &len) != 1) {
53     EVP_CIPHER_CTX_free(ctx);
54     return -1;
55 }
56 ciphertext_len += len;
57
58 EVP_CIPHER_CTX_free(ctx);
59 return ciphertext_len;
60 }
61
62 // Function to compare ciphertexts
63 int compare_ciphertexts(const unsigned char *ciphertext1, const unsigned char *ciphertext2, size_t
    len) {
64     return memcmp(ciphertext1, ciphertext2, len);
65 }
66

```



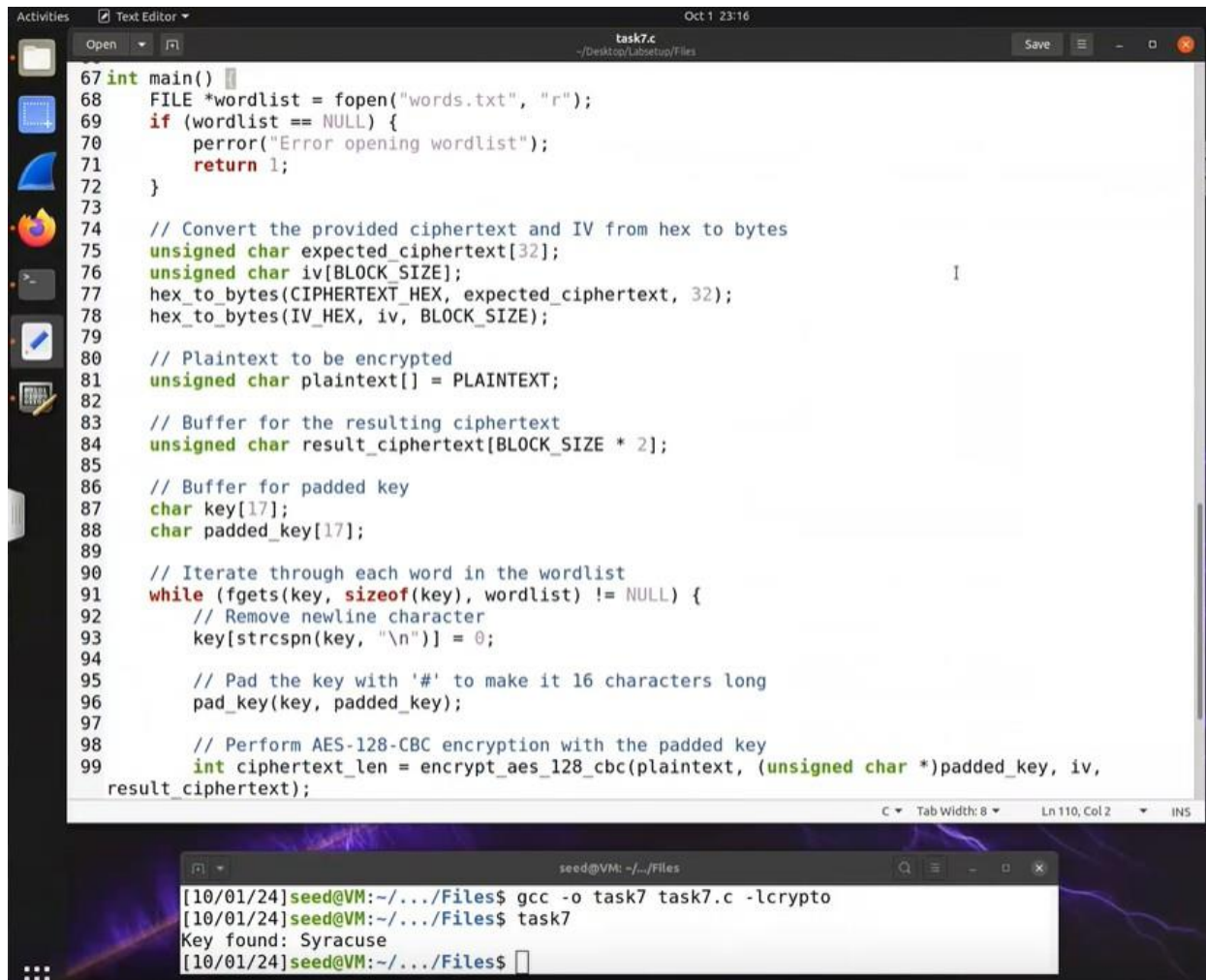
A terminal window titled 'seed@VM: ~/.../Files' showing the execution of the task7.c program. The user runs 'gcc -o task7 task7.c -lcrypto' and then 'task7'. The output shows 'Key found: Syracuse'.

```

[10/01/24]seed@VM:~/.../Files$ gcc -o task7 task7.c -lcrypto
[10/01/24]seed@VM:~/.../Files$ task7
Key found: Syracuse
[10/01/24]seed@VM:~/.../Files$

```

Task7(continuation of task7.c code)

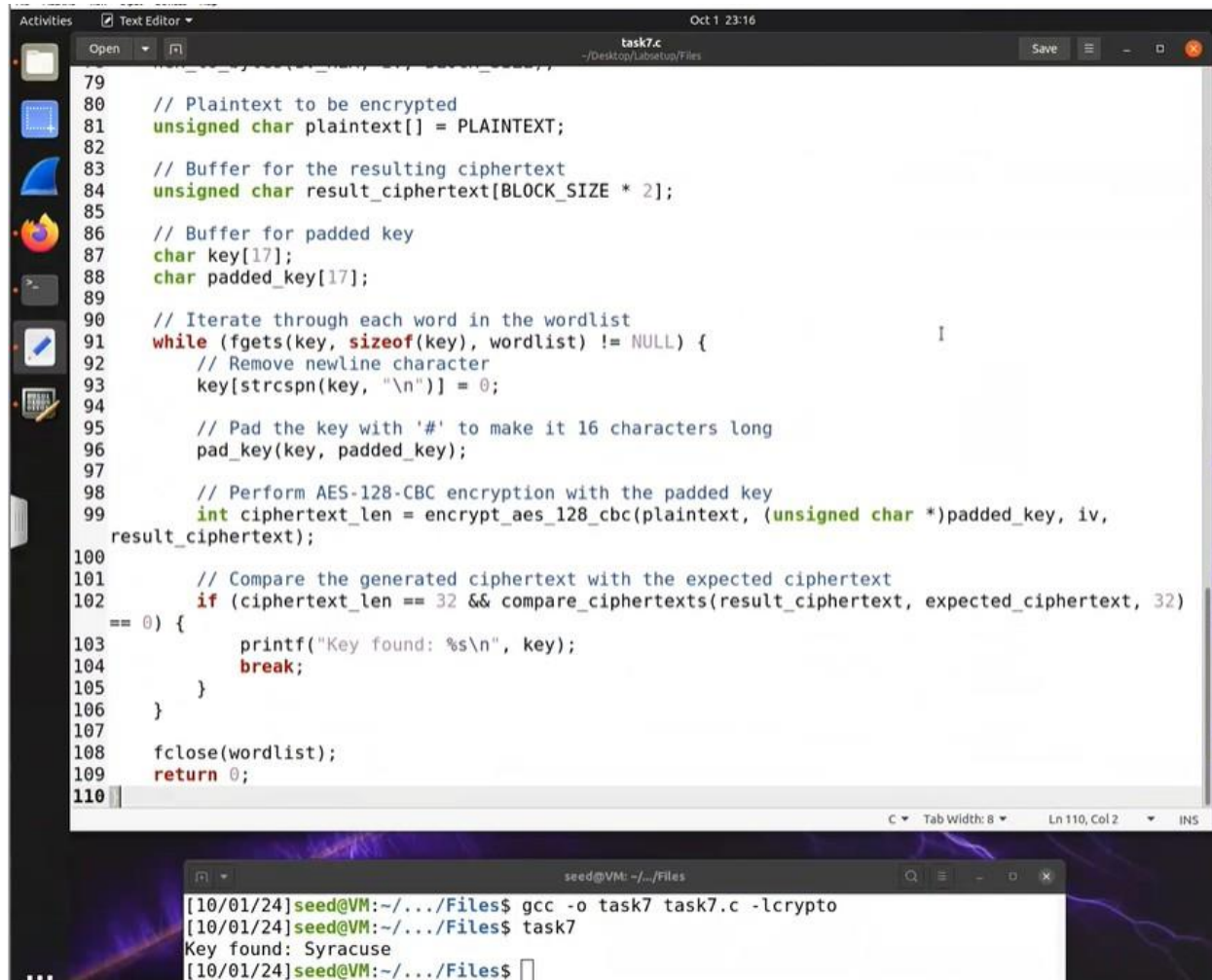


The screenshot shows a Linux desktop environment. At the top, there is a taskbar with icons for Activities, Text Editor, and a clock showing Oct 1 23:16. The main window is a text editor titled 'task7.c' with the path '~/.Desktop/Labsetup/Files'. It contains C code for a program that reads words from 'words.txt', converts hex ciphertext and IV to bytes, and performs AES-128-CBC encryption. Below the text editor is a terminal window titled 'seed@VM: ~/.Files'. It shows the compilation of 'task7.c' using 'gcc -o task7 task7.c -lcrypto', the execution of 'task7', and the output 'Key found: Syracuse'.

```
67 int main() {
68     FILE *wordlist = fopen("words.txt", "r");
69     if (wordlist == NULL) {
70         perror("Error opening wordlist");
71         return 1;
72     }
73
74     // Convert the provided ciphertext and IV from hex to bytes
75     unsigned char expected_ciphertext[32];
76     unsigned char iv[BLOCK_SIZE];
77     hex_to_bytes(CIPHERTEXT_HEX, expected_ciphertext, 32);
78     hex_to_bytes(IV_HEX, iv, BLOCK_SIZE);
79
80     // Plaintext to be encrypted
81     unsigned char plaintext[] = PLAINTEXT;
82
83     // Buffer for the resulting ciphertext
84     unsigned char result_ciphertext[BLOCK_SIZE * 2];
85
86     // Buffer for padded key
87     char key[17];
88     char padded_key[17];
89
90     // Iterate through each word in the wordlist
91     while (fgets(key, sizeof(key), wordlist) != NULL) {
92         // Remove newline character
93         key[strcspn(key, "\n")] = 0;
94
95         // Pad the key with '#' to make it 16 characters long
96         pad_key(key, padded_key);
97
98         // Perform AES-128-CBC encryption with the padded key
99         int ciphertext_len = encrypt_aes_128_cbc(plaintext, (unsigned char *)padded_key, iv,
result_ciphertext);
```

```
[10/01/24]seed@VM:~/.../Files$ gcc -o task7 task7.c -lcrypto
[10/01/24]seed@VM:~/.../Files$ task7
Key found: Syracuse
[10/01/24]seed@VM:~/.../Files$
```

Task7(continuation of task7.c code)



The screenshot shows a Linux desktop environment. The top panel displays the date and time as 'Oct 1 23:16'. The main window is a text editor titled 'task7.c' with the file path '~/.Desktop/Libsetup/Files'. The code in the editor is as follows:

```
79
80 // Plaintext to be encrypted
81 unsigned char plaintext[] = PLAINTEXT;
82
83 // Buffer for the resulting ciphertext
84 unsigned char result_ciphertext[BLOCK_SIZE * 2];
85
86 // Buffer for padded key
87 char key[17];
88 char padded_key[17];
89
90 // Iterate through each word in the wordlist
91 while (fgets(key, sizeof(key), wordlist) != NULL) {
92     // Remove newline character
93     key[strcspn(key, "\n")] = 0;
94
95     // Pad the key with '#' to make it 16 characters long
96     pad_key(key, padded_key);
97
98     // Perform AES-128-CBC encryption with the padded key
99     int ciphertext_len = encrypt_aes_128_cbc(plaintext, (unsigned char *)padded_key, iv,
100 result_ciphertext);
101
102     // Compare the generated ciphertext with the expected ciphertext
103     if (ciphertext_len == 32 && compare_ciphertexts(result_ciphertext, expected_ciphertext, 32)
104 == 0) {
105         printf("Key found: %s\n", key);
106         break;
107     }
108 }
109 fclose(wordlist);
110 return 0;
```

Below the text editor is a terminal window titled 'seed@VM: ~/Files'. It shows the following commands and output:

```
[10/01/24]seed@VM:~/Files$ gcc -o task7 task7.c -lcrypto
[10/01/24]seed@VM:~/Files$ task7
Key found: Syracuse
[10/01/24]seed@VM:~/Files$
```

Task7(continuation of task7.c code)