

Secret Encryption

SEED LABS

Contents

Environment Setup	2
Task 1.....	3
Step 1	3
Step 2	3
Step 3	4
Task 2.....	8
Task 3.....	9
Step 1	9
Step 2	10
Task 4.....	12
Step 1	12
ECB Mode.....	12
CBC Mode.....	12
CFB Mode.....	12
OFB Mode	12
Step 2	13

Environment Setup

Building Container Image.

```
seed@VM: ~/.../Labsetup
[09/16/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  encryption_oracle  Files
[09/16/23]seed@VM:~/.../Labsetup$ dcbuild
Building oracle-server
Step 1/8 : FROM handsonsecurity/seed-ubuntu:dev AS builder
dev: Pulling from handsonsecurity/seed-ubuntu
da7391352a9b: Already exists
14428a6d4bcd: Already exists
2c2d948710f2: Already exists
5d39fdfbe330: Already exists
56b236c9d9da: Already exists
1bb168ce59cc: Already exists
588b6963c007: Already exists
c3c83e840346: Downloading [>
c3c83e840346: Downloading [>
c3c83e840346: Downloading [>
c3c83e840346: Downloading [=>
c3c83e840346: Downloading [=>
c3c83e840346: Downloading [=>
c3c83e840346: Pull complete
Digest: sha256:f30e4224cf90ab83606285e4e1b12ef3879d3f6ec24aee991c00aeb52295551
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:dev
---> 89212aee292b
Step 2/8 : COPY . /oracle
```

Initiating Containers.

```
[09/16/23]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (B-10.9.0.6, server-10.9.0.5, A-10.9.0.5, serve
r-10.9.0.6, M-10.9.0.105) for this project. If you removed or renamed this servi
ce in your compose file, you can run this command with the --remove-orphans flag
to clean it up.
Creating oracle-10.9.0.80 ... done
Attaching to oracle-10.9.0.80
oracle-10.9.0.80 | Server listening on 3000 for known_iv
```

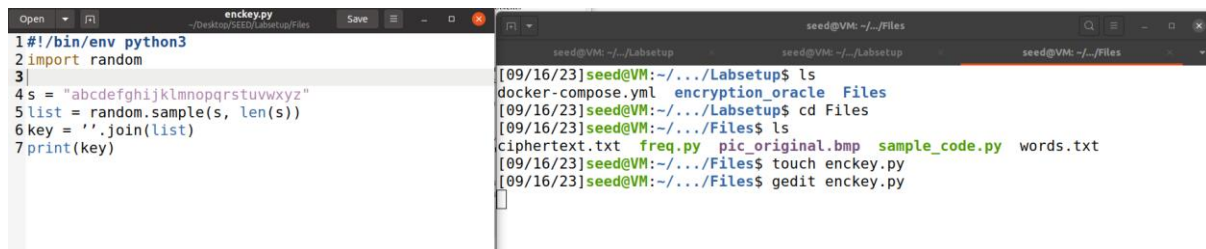
Connecting to the available Docker in a new terminal.

```
seed@VM: ~/.../Labsetup
[09/16/23]seed@VM:~/.../Labsetup$ dockps
ad679b689b9f  oracle-10.9.0.80
[09/16/23]seed@VM:~/.../Labsetup$ docksh ad67
root@ad679b689b9f:/oracle#
```

Task 1

Step 1

Using the provided code in the manual to generate encryption key.



The screenshot shows two windows. The left window is a code editor titled 'enckey.py' with the following Python code:

```
1#!/bin/env python3
2import random
3
4s = "abcdefghijklmnopqrstuvwxyz"
5list = random.sample(s, len(s))
6key = ''.join(list)
7print(key)
```

The right window is a terminal with the following commands and output:

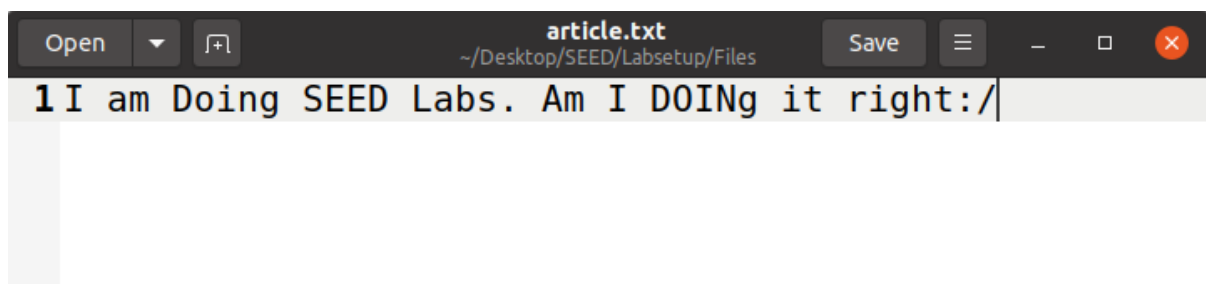
```
seed@VM: ~/.../Files
[09/16/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  encryption_oracle  Files
[09/16/23]seed@VM:~/.../Labsetup$ cd Files
[09/16/23]seed@VM:~/.../Files$ ls
ciphertext.txt  freq.py  pic_original.bmp  sample_code.py  words.txt
[09/16/23]seed@VM:~/.../Files$ touch enckey.py
[09/16/23]seed@VM:~/.../Files$ gedit enckey.py
```

Executing the script to permute the alphabet from a to z using Python and use the permuted alphabet as the key. Each time it is run a randomly generated key is provided.

```
[09/16/23]seed@VM:~/.../Files$ python3 enckey.py
qhwidlopjcrvsexanbftykugzm
[09/16/23]seed@VM:~/.../Files$
```

Step 2

Creating a new text file with some text in it.



The screenshot shows a text editor window titled 'article.txt' with the following text:

```
1 I am Doing SEED Labs. Am I DOING it right:/
```

As shown in the screenshot the command worked and changed the text to lowercase letters.

```
[09/16/23]seed@VM:~/.../Files$ tr [:upper:] [:lower:] < article.txt > lowercase.txt
[09/16/23]seed@VM:~/.../Files$ cat lowercase.txt
i am doing seed labs. am i doing it right:/
[09/16/23]seed@VM:~/.../Files$
```

Now with the shown command I have removed special characters and numbers, if any.

```
[09/16/23]seed@VM:~/.../Files$ tr -cd '[a-z][\n][:space:]' < lowercase.txt > plaintext.txt
[09/16/23]seed@VM:~/.../Files$ cat plaintext.txt
i am doing seed labs am i doing it right
[09/16/23]seed@VM:~/.../Files$
```

Step 3

Generating key from the provided script.

```
[09/16/23]seed@VM:~/.../Files$ python3 enckey.py  
hrvnxolwezukdjfmitytbqspgca  
[09/16/23]seed@VM:~/.../Files$ █
```

I have encrypted the plaintext and made it ciphertext stored in cipher.txt as there is already provided ciphertext.txt.

```
[09/16/23]seed@VM:~/.../Files$ tr 'abcdefghijklmnopqrstuvwxyz' 'hrvnxolwezukd  
iytbqspgca' < plaintext.txt > cipher.txt  
[09/16/23]seed@VM:~/.../Files$ cat cipher.txt  
e hd nfejl txn khrt hd e nfejl eb yelwb  
[09/16/23]seed@VM:~/.../Files$
```

Editing the provided frequency analysis code to read from cipher.txt.

```
freq.py
~/Desktop/SEED/Labsetup/Files
Open Save

1#!/usr/bin/env python3
2
3from collections import Counter
4import re
5
6TOP_K = 20
7N_GRAM = 3
8
9# Generate all the n-grams for value n
10def ngrams(n, text):
11    for i in range(len(text) - n + 1):
12        # Ignore n-grams containing white space
13        if not re.search(r'\s', text[i:i+n]):
14            yield text[i:i+n]
15
16# Read the data from the ciphertext
17with open('cipher.txt') as f:
18    text = f.read()
19
20# Count, sort, and print out the n-grams
21for N in range(N_GRAM):
22    print("-----")
23    print("{}-gram (top {}):".format(N+1, TOP_K))
24    counts = Counter(ngrams(N+1, text)) #
25    sorted_counts = counts.most_common(TOP_K) #
26    for ngram, count in sorted_counts:
27        print("{}: {}".format(ngram, count)) #
28    Print
```

This is the result I got from running Frequency Analysis code.

```
[09/16/23] seed@VM:~/.../Files$ ./freq.py
```

```
-----  
1-gram (top 20):
```

```
e: 6  
h: 3  
n: 3  
l: 3  
d: 2  
f: 2  
j: 2  
t: 2  
x: 2  
b: 2  
k: 1  
r: 1  
y: 1  
w: 1
```

```
-----  
2-gram (top 20):
```

```
hd: 2  
nf: 2  
fe: 2  
ej: 2  
jl: 2  
tx: 1  
xx: 1  
xn: 1  
kh: 1  
hr: 1  
rt: 1  
eb: 1  
ye: 1  
_ : 1
```

```
ye: 1
el: 1
lw: 1
wb: 1
-----
3-gram (top 20):
nfe: 2
fej: 2
ejl: 2
txx: 1
xxn: 1
khr: 1
hrt: 1
yel: 1
elw: 1
lwb: 1
[09/16/23] seed@VM: ~/.../Files$
```

Now I changed the Frequency Analysis to read from provided ciphertext.txt and running the script now gave these results.

```
[09/16/23] 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
gn: 32
aw: 31
-----
3-gram (top 20):
ytn: 78
vup: 30
mur: 20
ynh: 18
xzy: 16
mou: 14
gnq: 14
ytv: 13
nqv: 13
vii: 13
bxh: 13
lvq: 12
nuy: 12
vym: 12
uvy: 11
```

Task 2

I will be using the plaintext.txt I made in Step 2 of Task 1 and trying encryption using 3 cipher types.

```
[09/16/23]seed@VM:~/.../Files$ cat plaintext.txt
i am doing seed labs am i doing it right
[09/16/23]seed@VM:~/.../Files$
```

First, I encrypted using aes 128 cbc cipher and stored in encrypted.bin.

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in plaintext.txt -out encrypted.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$
```

Then I encrypted using aes 128 cbc cipher and stored in encrypted1.bin.

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -bf-cbc -e -in plaintext.txt -out encrypted1.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[09/16/23]seed@VM:~/.../Files$
```

Finally, I encrypted using aes 128 cbc cipher and stored in encrypted2.bin.

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cfb -e -in plaintext.txt -out encrypted2.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$
```

This is what my encrypted files look like using the mentioned ciphers above.

```
[09/16/23]seed@VM:~/.../Files$ hexdump -C encrypted.bin
00000000 c3 3b 30 55 cc ce 1e 15 52 74 d5 f4 cf 66 91 e1 |.;0U....Rt...f..|
00000010 60 33 ee 35 4c c1 cc 89 c8 1a cc 11 82 25 6b 0d |`3.5L.....%k..|
00000020 8b 98 67 7b 31 45 f5 b0 bc 9f 53 d0 9b 46 bb 00 |..g{1E....S..F..|
00000030
[09/16/23]seed@VM:~/.../Files$ hexdump -C encrypted1.bin
00000000 34 e3 97 7d 43 d5 3a 84 2f f5 fd db 26 f8 6c e9 |4..}C.../...&.l.|
00000010 0f 28 79 55 67 e5 2d ec 0b dd 7b eb 9a e4 ce 69 |.(yUg.-...{....i|
00000020 04 a2 3c 7c 7e 88 13 30 54 dc df 06 45 f5 60 b6 |..<|~..0T...E.`.|
00000030
[09/16/23]seed@VM:~/.../Files$ hexdump -C encrypted2.bin
00000000 ee a6 ee 48 e1 5b d5 b8 c1 f9 9f 6e 10 75 38 4d |...H.[.....n.u8M|
00000010 93 5d 39 a1 1b 4c 10 f9 72 2a 5e 7b 9d fb f4 bf |.]9..L..r*^{....|
00000020 b9 51 28 ad d0 97 30 af 3d |.Q(...0.=|
00000029
[09/16/23]seed@VM:~/.../Files$
```

Decrypting the encrypted text for good measure to check if it comes back to the original text which verifies the success of encryption and decryption.

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -d -in encrypted.bin -out decrypt.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$ openssl enc -bf-cbc -d -in encrypted1.bin -out decrypt1.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cfb -d -in encrypted2.bin -out decrypt2.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

The result for the cipher aes 128 cb, bf cbc and aes 128 cfb are stored in decrypt.txt, decrypt1.txt and decrypt2.txt, respectively. And the result matches for each cipher encrypted file after decryption with the original plain text.

```
[09/16/23]seed@VM:~/.../Files$ cat decrypt.txt
i am doing seed labs am i doing it right
[09/16/23]seed@VM:~/.../Files$ cat decrypt1.txt
i am doing seed labs am i doing it right
[09/16/23]seed@VM:~/.../Files$ cat decrypt2.txt
i am doing seed labs am i doing it right
[09/16/23]seed@VM:~/.../Files$ █
```

Task 3

Step 1

Encrypting original image provided in Labsetup with aes 128 ecb and cbc ciphers and it is worth mentioning aes 128 ecb doesn't require Initialization Vector known as iv.

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out pic_ecb_encrypted.bmp -K 00112233445566778889aabbccddeeff
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out pic_cbc_encrypted.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$ █
```

Extracting the Header of the original image file.

```
[09/16/23]seed@VM:~/.../Files$ head -c 54 pic_original.bmp > header
[09/16/23]seed@VM:~/.../Files$ █
```

Extracting Data from the encrypted images.

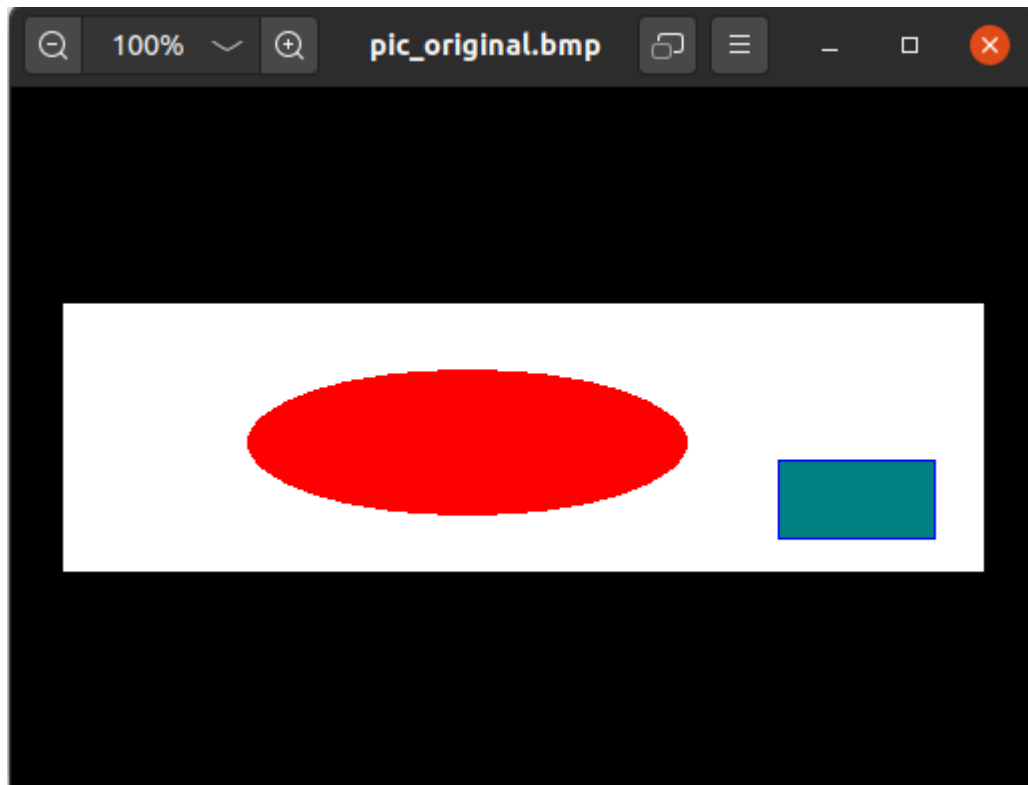
```
[09/16/23]seed@VM:~/.../Files$ tail -c +55 pic_ecb_encrypted.bmp > body_ecb
[09/16/23]seed@VM:~/.../Files$ tail -c +55 pic_cbc_encrypted.bmp > body_cbc
[09/16/23]seed@VM:~/.../Files$
```

Combining header and body to create a viewable image.

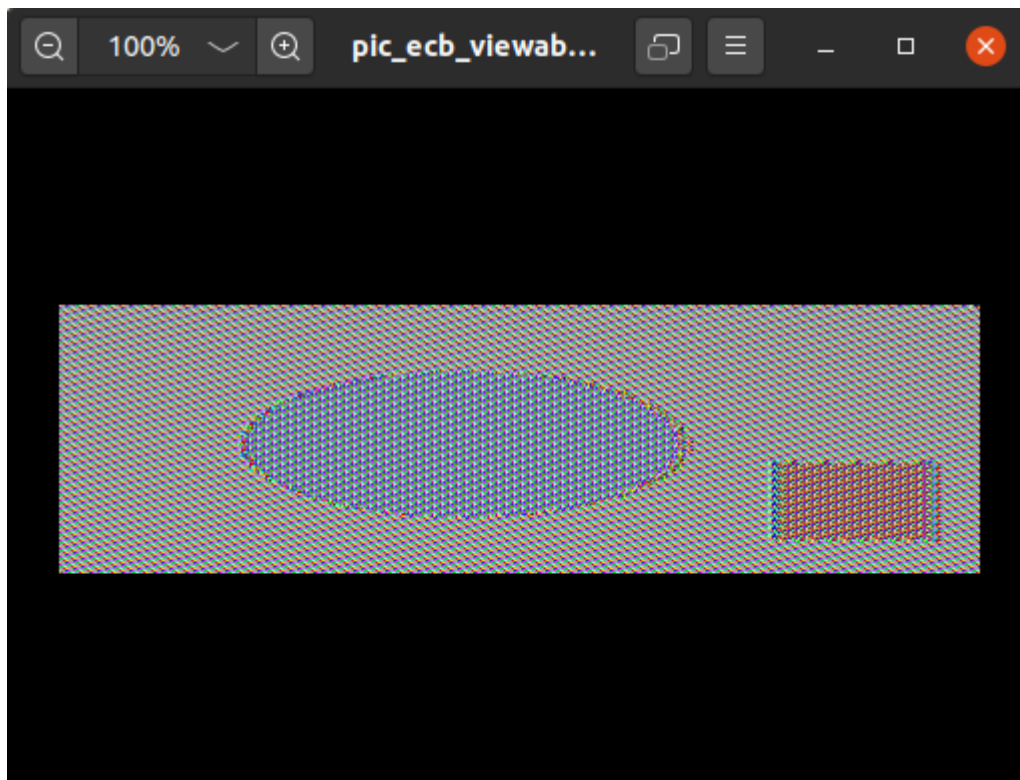
```
[09/16/23] seed@VM:~/.../Files$ cat header body_ecb > pic_ecb_viewable.bmp  
[09/16/23] seed@VM:~/.../Files$ cat header body_cbc > pic_cbc_viewable.bmp  
[09/16/23] seed@VM:~/.../Files$ █
```

Step 2

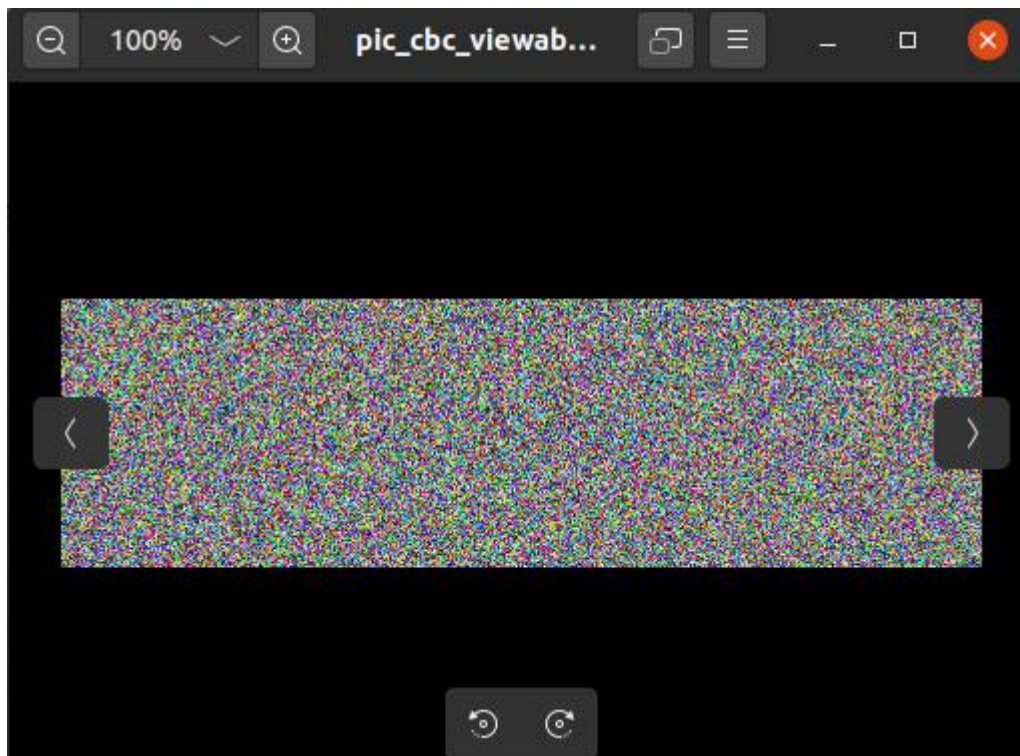
This is the original image.



Now the aes 128 ecb image is like this which is blur but reveals the contents of the real image using eob with command *"eog pic_ecb_viewable.bmp"*.



Now the aes 128 cbc image is like this which doesn't reveal any details regarding the original image as seen by the naked eye by using eob with command *"eog pic_cbc_viewable.bmp"*.



Task 4

Step 1

I'll be using again plaintext.txt I created earlier.

```
[09/16/23]seed@VM:~/.../Files$ cat plaintext.txt
i am doing seed labs am i doing it right
[09/16/23]seed@VM:~/.../Files$
```

Now I will be encrypting in different modes.

ECB Mode

ECB Mode doesn't require iv.

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-ecb -e -in plaintext.txt -out ciphertext_ecb.bin -K 00112233445566778889aabbccddeeff
```

Padding is not done in this mode as size is a multiple of 16 bytes which is 128 bits in AES. Moreover, this mode uses padding because it uses fixed size.

```
[09/16/23]seed@VM:~/.../Files$ ls -lh ciphertext_ecb.bin
-rw-rw-r-- 1 seed seed 48 Sep 16 05:28 ciphertext_ecb.bin
```

CBC Mode

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in plaintext.txt -out ciphertext_cbc.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

Like ECB this mode also requires padding due to requirement of fixed size.

```
[09/16/23]seed@VM:~/.../Files$ ls -lh ciphertext_cbc.bin
-rw-rw-r-- 1 seed seed 48 Sep 16 05:28 ciphertext_cbc.bin
```

CFB Mode

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cfb -e -in plaintext.txt -out ciphertext_cfb.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

Padding isn't done as this mode turns block cipher into a stream cipher, so it doesn't require padding. It operates on smaller units than the block size and can handle plaintext of any length.

```
[09/16/23]seed@VM:~/.../Files$ ls -lh ciphertext_cfb.bin
-rw-rw-r-- 1 seed seed 41 Sep 16 05:28 ciphertext_cfb.bin
```

OFB Mode

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-ofb -e -in plaintext.txt -out ciphertext_ofb.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

OFB also turns a block cipher into a stream cipher and doesn't require padding. It generates a stream of key bits that XOR with the plaintext, so the length of the plaintext can vary.

```
[09/16/23]seed@VM:~/.../Files$ ls -lh ciphertext_ofb.bin
-rw-rw-r-- 1 seed seed 41 Sep 16 05:28 ciphertext_ofb.bin
```


Step 2

I created three files of size 5, 10 and 15 bytes respectively.

```
[09/16/23]seed@VM:~/.../Files$ echo -n "12345" > f1.txt
[09/16/23]seed@VM:~/.../Files$ echo -n "1234567890" > f2.txt
[09/16/23]seed@VM:~/.../Files$ echo -n "123456789012345" > f3.txt
[09/16/23]seed@VM:~/.../Files$
```

I encrypted all of the created files with AES 128 CBC cipher.

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in f1.txt -out f1_
encrypted.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in f2.txt -out f2_
encrypted.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in f3.txt -out f3_
encrypted.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$
```

Decrypted the file with -nopad to prevent automatic padding removal during decryption and stored in the files p1.txt, p2.txt and p3.txt, respectively.

```
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -d -nopad -in f1_encrypt
ed.bin -out p1.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -d -nopad -in f2_encrypt
ed.bin -out p2.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -d -nopad -in f3_encrypt
ed.bin -out p3.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/16/23]seed@VM:~/.../Files$
```

Now checking the data in all files using hexdump.

```
[09/16/23]seed@VM:~/.../Files$ hexdump -C p1.txt
00000000 31 32 33 34 35 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b |12345.....|
00000010
[09/16/23]seed@VM:~/.../Files$ hexdump -C p2.txt
00000000 31 32 33 34 35 36 37 38 39 30 06 06 06 06 06 06 |1234567890.....|
00000010
[09/16/23]seed@VM:~/.../Files$ hexdump -C p3.txt
00000000 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 01 |123456789012345.|
00000010
[09/16/23]seed@VM:~/.../Files$
```

Checking the data in all files using xxd.

```
[09/16/23] seed@VM:~/.../Files$ xxd p1.txt
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b 12345.....
[09/16/23] seed@VM:~/.../Files$ xxd p2.txt
00000000: 3132 3334 3536 3738 3930 0606 0606 0606 1234567890.....
[09/16/23] seed@VM:~/.../Files$ xxd p3.txt
00000000: 3132 3334 3536 3738 3930 3132 3334 3501 123456789012345.
[09/16/23] seed@VM:~/.../Files$ █
```

NOTE: I used hexdump and xxd tools to display the results and padding represented by dots which as whole output is not representable in output files normally as shown in the screenshot below the padding is not displayed without hex tools.

```
[09/16/23] seed@VM:~/.../Files$ cat p1.txt
12345
```

```

[09/16/23] seed@VM:~/.../Files$ cat p2.txt
1234567890[09/16/23] seed@VM:~/.../Files$ cat p3.txt
123456789012345[09/16/23] seed@VM:~/.../Files$ █
```
