EXPERIMENT 3

Name: Abhishek Chopra **UID: 2019130009**

Batch A **Subject: CSS**

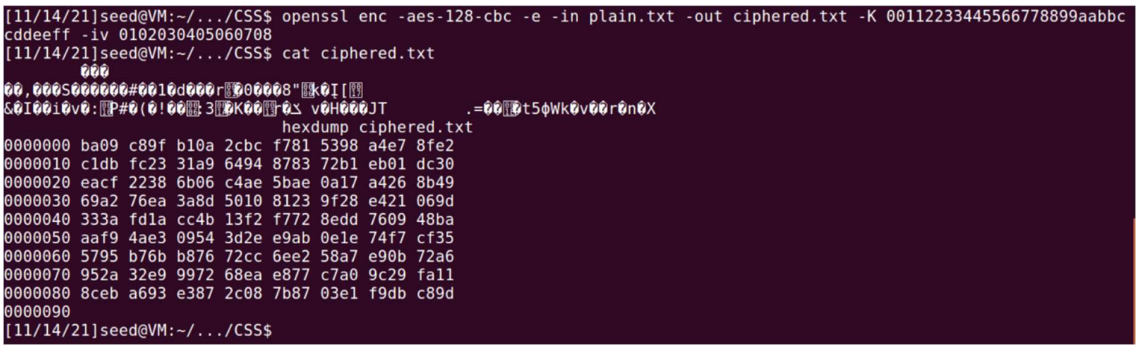
Task 1: Encryption using different ciphers and modes

-aes-128-ecb

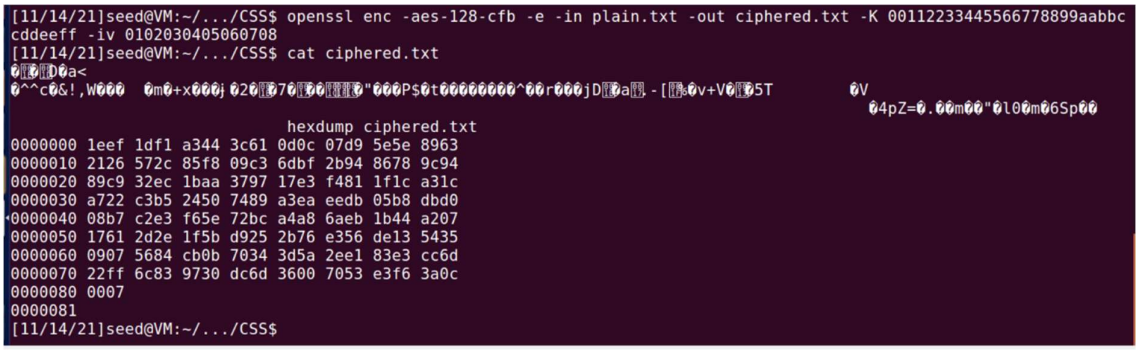
```
[11/14/21]seed@VM:~/.../CSS$ openssl enc -aes-128-ecb -e -in plain.txt -out ciphered.txt -K 00112233445566778899aabbc
cddeeff
[11/14/21]seed@VM:~/.../CSS$ cat ciphered.txt
H
yx(@@H@_@H@@@@@w@@H@@H@]@@8@@@h}@H@@@@@?
                                                H@@l@+@H@-@@@@H@@H@@G@@1@5jO@/-fOR@4@@@@H@-5`@@T@X@i=`        @@Y@@@w_@@@
                                ~CH@K@`@@@@@2{A_@
n@f@(-a@@@@%@@H@H@u@@ BD@@H@@l{H@_.H@#@M@qQ@@>I@@@H@@@Tr@@@@@@&@G3ju_@@M@@@@@@@H@*@;@H@_8@H@@+gbH@@VH@I@h@^@@,w@Zo#@@@@[@
CUH@^@p@@@sy
@[@GX@ @@@H@@B  @z2@@GRH@@MI@@`'@M@Y@@b@@76@,u@@x@g@'@x@@@H@zs6@@>H@h(@t7áz@B@@@ca^ o@H@@H@H@@@@H@U@?(@@@Ì)3@@@H@H@@@
H@pD9lfH@N@S@@5@9\j@@H@_+@@@u@@@@@@L$@~@@@o;@09`w+
                                @@@@@D@@H@@@H\?b@@A@@"@3?@v@o9@@@
:@@Y@_r@=@%H@@{@H@L=TƒØZ@Q@@
@λ@@@q@@H@u4扡@p@@:@@rrj@@w@)H@~@@@-FH@@@@@x@7H@3O@@@Vs@XH@@%@<@@H@@@H@9H@@.@@)V@@v|@05\}z@@ć@{H@-@@@@@LG@@@@@c@@X@@@;
]@,@n@@@H@@o兯 @dqXE
@@@@@V*.@N@@H@0OYH@@@%@~H@@H@@5@]@@ηQVK@<5@=%@@Gu@\a@3@@@ı1@@|
                @@@@cE;@@@6xŸ5%o@@@@@d@@Q@@@PH@@H;=c@@@pV@@ @(@@KK@@N@@@@@@@>H@@i@@@@u-NZ|Z@@(@@I>@_5J"@@@@<@@h@0-D
_;H}8O@@@@@?M@5@@@H@xO:λ@EY@H@@BU@>>95@@@V(@H@@@@@@v@@@@@x@b@\3.
$@@H@d@@~@@   +@H7X @3H@9@DM@H@B@X@H@H@P@H@@@@b@@$k@.@H@@@B@@@{@@~R@@@@@K@@IR@\/@3@@@sHH@2剖 '@@@@@b#@xj7@H@r@j@@
H@@@@@twO@r@h4@@u@@-#@@H@j@@@,?w@UH @@Pi@iWw@
                                H@.@ô>@Q@R@>[H@@@}@@@H@@|@H@404H@@HH @I@H@@200[11/14/21]seed@VM:~/.../CSS$ hexdump
ciphered.txt
0000000 0a1b 7879 cd28 10f1 5f13 06ea f571 80ab
0000010 f0a7 ca77 1efa fe25 16ef 5dee c9f5 f138
0000020 b9e7 7d68 149b e3b5 d1c7 919d 0c3f fc1e
0000030 6cf5 2bc8 1de3 2d74 94a6 c998 af06 01b5
0000040 81a3 ef47 31b8 5396 4f6a 2fd7 662d 524f
0000050 34d5 88e7 b4fc 7e16 6035 8684 db54 cc58
```

```
Terminal
0000320 63e6 3b45 7fd4 9bba 7836 b2c8 2535 6f05
0000330 b3a9 9c9b 64eb ef0f 3789 4812 fc64 510d
0000340 c1de 5089 4012 1200 3d3b a9c2 e9bb a8db
0000350 70c9 b856 aa7f ab20 9028 4b89 914b 4ed4
0000360 c0d6 960e 0eb5 e1ff 3e93 e003 d407 f069
0000370 c9a1 75e5 4e2d 7c5a d05a 28f0 939a 3e49
0000380 5f99 4a35 e622 f3dc 3ce9 c6a2 ee68 2d30
0000390 5f44 013b 86d0 387d df4f e005 b89d 3fc5
00003a0 fc4d f435 feda a9ea e09f 4f78 ce3a adbb
00003b0 5945 0188 ce86 5542 3ed3 393e b035 8ca3
00003c0 2856 17bf d6d2 85fe a4fd 9f76 e5f8 ceb6
00003d0 db78 ad62 da5c 2e8b db0a 24ad e8c0 a61f
00003e0 ff64 7ea5 e892 2b09 1296 5837 be5f 0333
00003f0 3954 108b 4d44 d7c7 b5b5 ba42 580e 01b7
0000400 d3be 0f87 d415 dc50 b910 9bb1 62f3 96e9
0000410 6b24 ded7 de8a 1719 f497 42af bdb6 7b87
0000420 8988 527e 9788 c38e 4bf4 ddef 5249 5cf5
0000430 fa2f a433 c990 4873 ad1e e532 9e89 9ad5
0000440 c3df b8e1 62d1 c623 6a78 c237 2b1f 72df
0000450 6afd ebf4 1920 f7f8 f58c 7f96 7774 ad4f
0000460 fc72 3468 d9d8 ef75 2de1 9223 0682 6aa7
0000470 a8b8 da98 3fbc fb77 ea55 9dbc d886 6950
0000480 e505 5769 9177 140c 2ea7 c5f7 3e8d 5186
0000490 d30e bf52 5b3e b106 b9a3 fe7d c180 fd17
00004a0 7cab 07d9 dc1a d834 4834 f6c9 1e04 9d27
00004b0 c549 7111 e5f6 aa32 dcf7 613d 8527 f604
00004c0 1770 9ba2 495f e3e0 0800 3e9c 4180 c191
00004d0
[11/14/21]seed@VM:~/.../CSS$
```

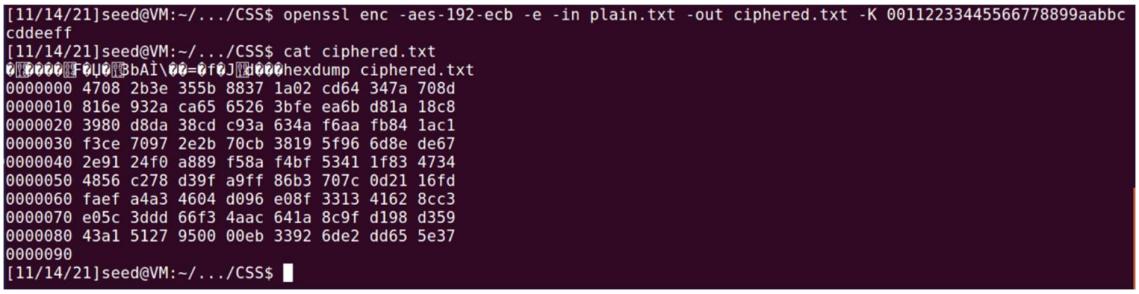
-aes-128-cbc



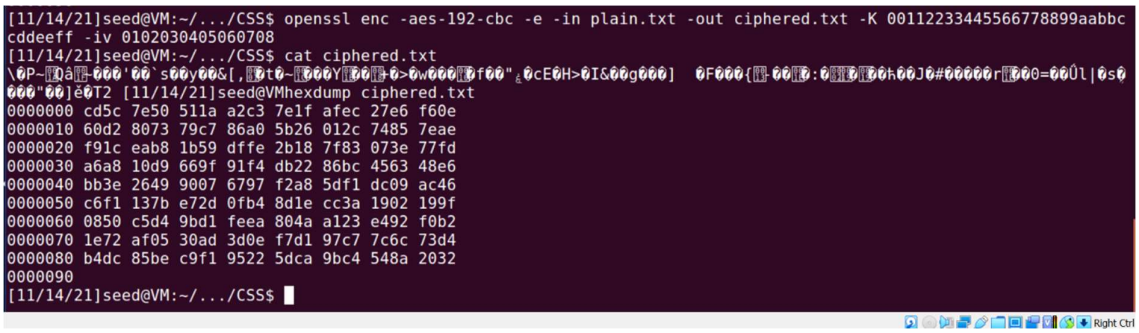
-aes-128-cfb



-aes-192-ecb



-aes-192-cbc



-aes-192-cfb

```
[11/14/21]seed@VM:~/.../CSS$ openssl enc -aes-192-cfb -e -in plain.txt -out ciphered.txt -K 00112233445566778899aabbc
cddeeff -iv 0102030405060708
[11/14/21]seed@VM:~/.../CSS$ cat ciphered.txt
¿Yw@\Ss(@▓@-@ufMe@▓5@▓@@@OhN@;07C@@@#@R@iw@@▓▓@@@▓@▓@@@L@
@S@@m=@@맂 @@@@L@▓@@@@v  s@@▓Lu@
                          hexdump ciphered.txt
0000000 21e9 d35b e250 94b4 bdeb 20d2 f0bc 66a6
0000010 8c29 ea78 b7dc 3515 14d8 cdf0 b243 520e
0000020 30d9 3104 4337 8ffc 23c1 529b 69a3 0777
0000030 c1ae 0116 ba00 a09c e204 f610 e0a4 4ca4
0000040 0d96 0d0d 84da 0759 bd77 535c 2873 1998
0000050 2dbf 7599 4d66 9f65 f009 684f 954e 303b
0000060 eb0a ee53 6daf c83d ebd7 979d ebd4 c4e2
0000070 834c e118 9cf6 76ef 7309 40f7 6c16 fd75
0000080 000c
0000081
[11/14/21]seed@VM:~/.../CSS$
```

-des-ecb

```
[11/14/21]seed@VM:~/.../CSS$ openssl enc -des-ecb -e -in plain.txt -out ciphered.txt -K 00112233445566778899aabbccdde
eff
hex string is too long
invalid hex key value
[11/14/21]seed@VM:~/.../CSS$ openssl enc -des-ecb -e -in plain.txt -out ciphered.txt -K 0123456789abcdef
[11/14/21]seed@VM:~/.../CSS$ cat ciphered.txt
@@\t@@<@@lr@@@KD@▓@!@@@@@>¥-yhexdump ciphered.txt
0000000 29b5 5521 025f 2b51 59b3 4d64 ffd1 c4e8
0000010 3cd8 9851 aecd d0c4 f2ce d067 ff56 c63b
0000020 ec34 03c9 9fdd bf54 e388 c988 22a0 f648
0000030 23cf ad8f 21bb 7345 c9c7 2992 91e8 309c
0000040 d8af 15f7 be5f 9822 2dce 48ba 21a2 e5a5
0000050 7341 98dc 2b1c 4c82 4d35 0dcc 9587 745c
0000060 8d8e 833c 6cc7 c872 87f2 444b 13fc 21df
0000070 099d 3ea8 a5c2 792d 303c 1d01 30f2 940e
0000080 6be0 296b 9a06 c4f5
0000088
[11/14/21]seed@VM:~/.../CSS$
```

-des-cfb

```
[11/14/21]seed@VM:~/.../CSS$ openssl enc -des-cfb -e -in plain.txt -out ciphered.txt -K 0123456789abcdef
iv undefined
[11/14/21]seed@VM:~/.../CSS$ openssl enc -des-cfb -e -in plain.txt -out ciphered.txt -K 0123456789abcdef -iv 01020304
05060708
[11/14/21]seed@VM:~/.../CSS$ cat ciphered.txt
@@;@s@@▓@R@@@@@@@@@6@@@@▓@i@@@@@FL@@@@@@@[@k@$▓@2X@@@@J@▓@
                                                          >@@@@@@@@M▓▓'@)@@@@ #h▓▓@▓@▓Z@@@@@@@@@OK/@E@@▓Z!▓▓ok~1Uog
@[11/14/21]seed@VM:~/.../CSS$hexdump ciphered.txt
0000000 eaa8 3b0e 73ef 8eb8 bfdf c62a aa52 0fb0
0000010 8581 e6f2 dfa7 36c7 bcb3 c8c7 a801 96d1
0000020 facc f1f8 46a4 a24c b8ba 818b 8593 055b
0000030 6b9d 2484 db03 5832 8a93 d9ba e94a f804
0000040 3e0b fcac e5be d1f0 b4f3 1f4d 0474 a622
0000050 f529 82f3 dca7 2383 0268 046c b9ae c51c
0000060 81b9 dedc adf8 cd8a a6e3 4b4f c22f a545
0000070 15c8 215a 6906 08f1 6f00 7e6b 5531 676f
0000080 00b4
0000081
[11/14/21]seed@VM:~/.../CSS$
```

-des-cbc

```
[11/14/21]seed@VM:~/.../CSS$ openssl enc -des-cbc -e -in plain.txt -out ciphered.txt -K 0123456789abcdef -iv 01020304
05060708
[11/14/21]seed@VM:~/.../CSS$ cat ciphered.txt
@f0@q@▓ 3▓@(@▓h@@;@
                  @@0(@@@t
@T@@@@SG@i@@r*@yC@s@@@~@u@@@@@@@▓:@@QJ@@_f@X@@G@)       v@5*@V?m@@@C@M▓uOZ-@▓@
                          hexdump ciphered.txt
0000000 66b3 cc4f ac71 92eb 3392 fd12 2807 13f4
0000010 8b6e 3b87 0c94 90ea 2830 83a0 74b1 e90a
0000020 f954 c0fe 53a7 8047 e569 72d5 eb2a 4379
0000030 738f 9085 decf e199 9b75 ace7 81c1 e687
0000040 1590 de3a 51d5 d74a 5fea 8c66 a658 ab00
0000050 f347 0929 8376 2a35 5690 0f3f 886d ddeb
0000060 e443 114d 4f75 2d5a 1785 0bfe e2e5 3d27
0000070 f09f b0a5 16d1 c8e5 7782 b6a4 fff9 78e5
0000080 621f 1fce 0e1a 4e24
0000088
[11/14/21]seed@VM:~/.../CSS$
```

Task 2: Encryption Mode – ECB vs. CBC

original picture:

image encrypted using ecb:

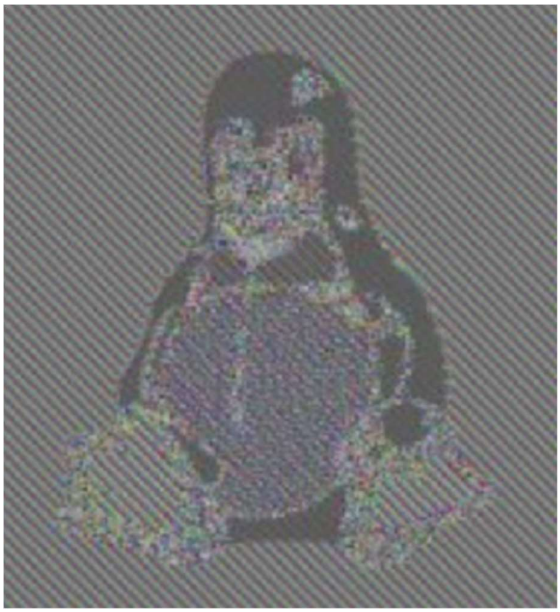
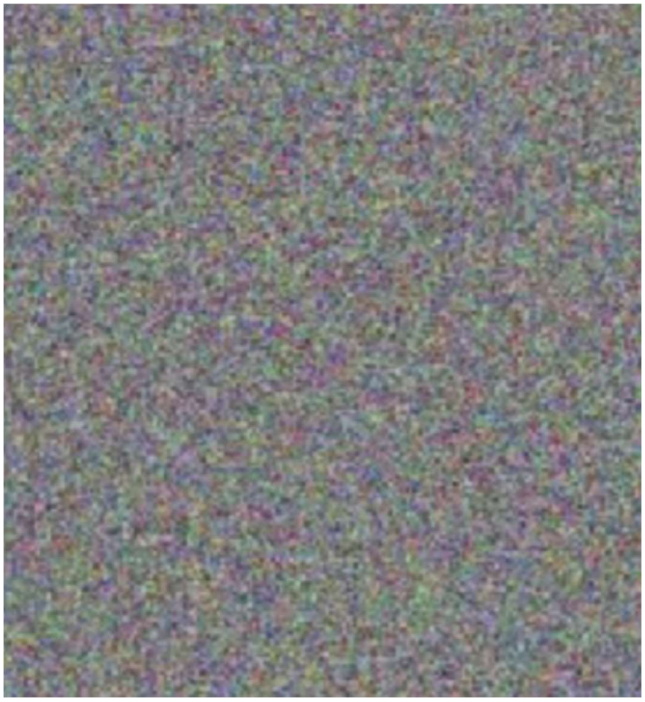


image encrypted using cbc:



Inference:
In the image encrypted using ECB, we can still see a set of certain colours and shapes from which we can try to figure out what the original picture could be. In the image encrypted using CBC the entire images is distorted and there is no way by which we can try to figure out the original picture.

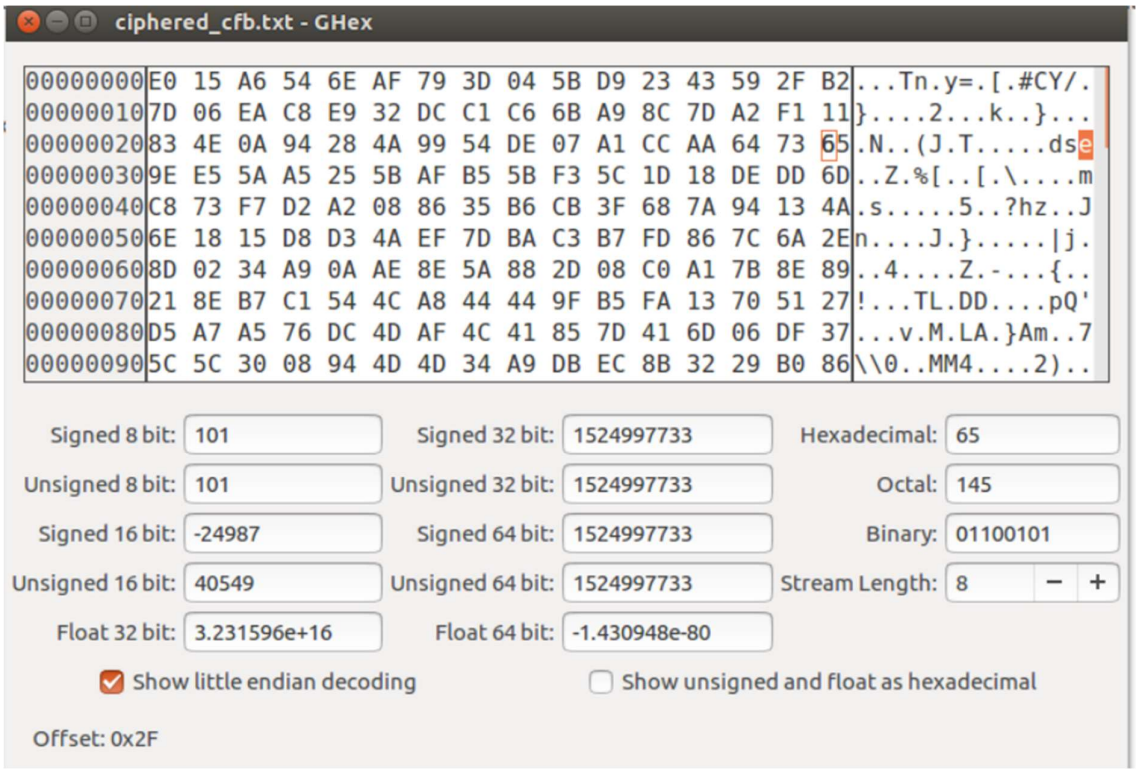
Task 3: Encryption Mode – Corrupted Cipher Text

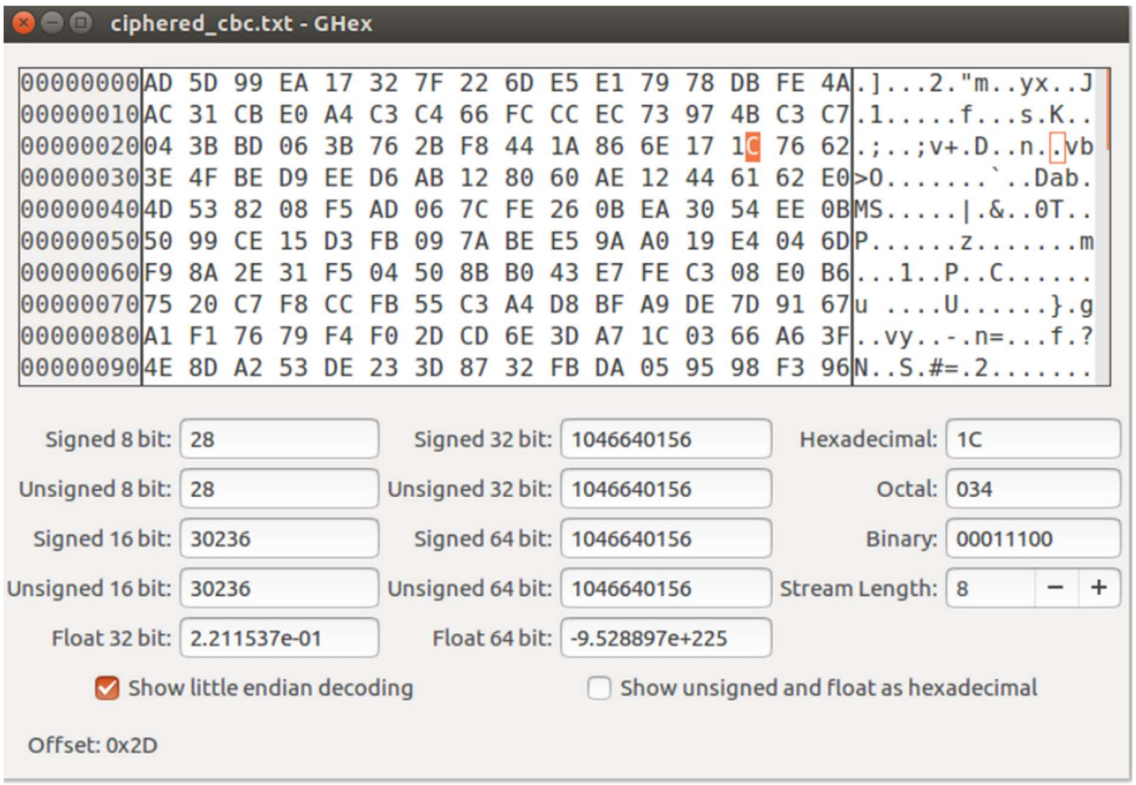
Original plain text:

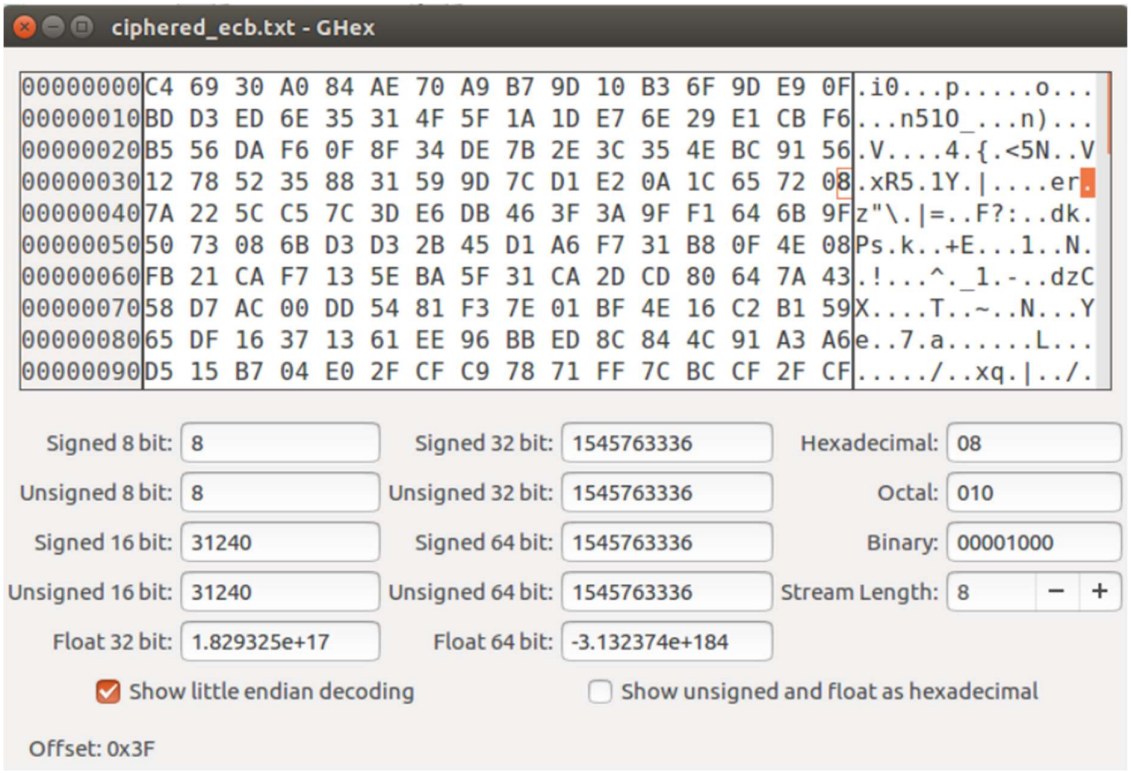
plain.txt × | Untitled Document × | ciphered_cbc_o.txt × | ciphered_cfb_o.txt × | ciphered_ecb_o.txt × | ciphered_ofb_o.txt × | hello.txt ×

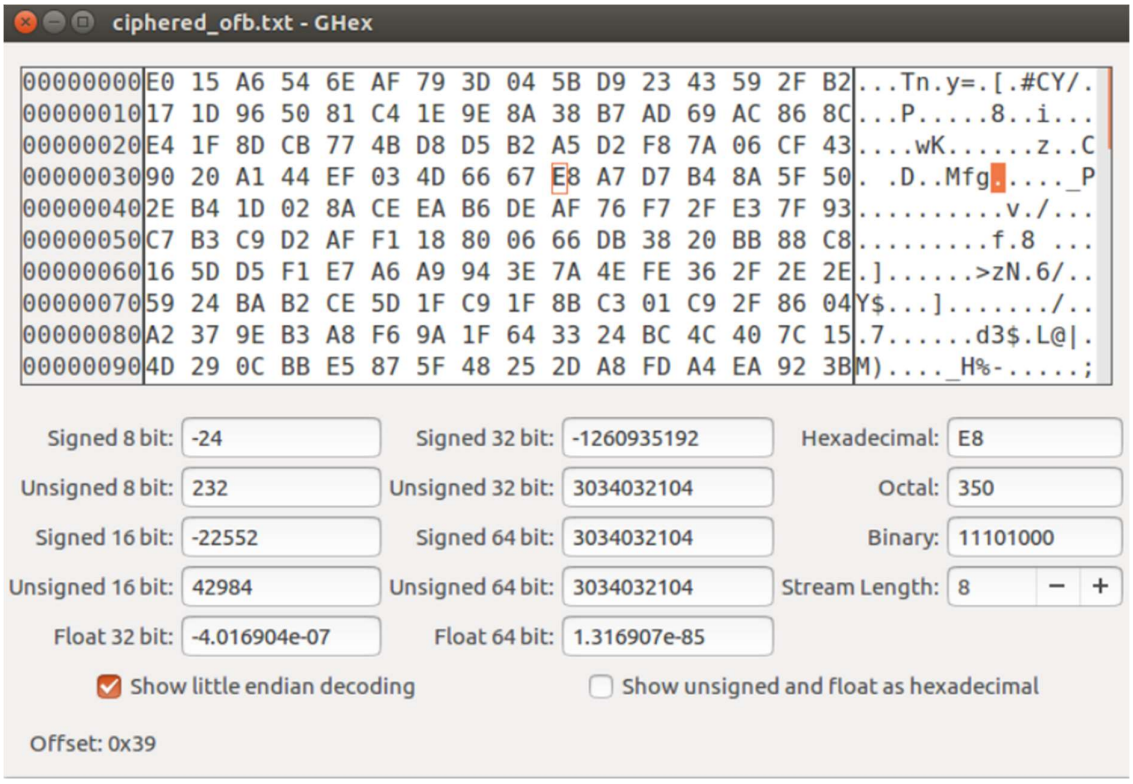
```
An initial public offering (IPO) refers to the process of offering shares of a private corporation to the public in a new stock issuance.
Companies must meet requirements by exchanges and the Securities and Exchange Commission (SEC) to hold an IPO.
IPOs provide companies with an opportunity to obtain capital by offering shares through the primary market.
Companies hire investment banks to market, gauge demand, set the IPO price and date, and more.
An IPO can be seen as an exit strategy for the company's founders and early investors, realizing the full profit from their private investment.
```

The above text is encrypted using aes-128 in 4 different modes: cbc, cfb, ofb and ecb.
After corrupting 30th byte of each cbc, cfb, ofb and ecb:


ciphered_cfb.txt - GHex
00000000 E0 15 A6 54 6E AF 79 3D 04 5B D9 23 43 59 2F B2 ...Tn.y=.[.#CY/.
00000010 7D 06 EA C8 E9 32 DC C1 C6 6B A9 8C 7D A2 F1 11 }....2...k..}...
00000020 83 4E 0A 94 28 4A 99 54 DE 07 A1 CC AA 64 73 65 .N..(J.T.....dse
00000030 9E E5 5A A5 25 5B AF B5 5B F3 5C 1D 18 DE DD 6D ..Z.%[..[.\....m
00000040 C8 73 F7 D2 A2 08 86 35 B6 CB 3F 68 7A 94 13 4A .s.....5..?hz..J
00000050 6E 18 15 D8 D3 4A EF 7D BA C3 B7 FD 86 7C 6A 2E n....J.}.....|j.
00000060 8D 02 34 A9 0A AE 8E 5A 88 2D 08 C0 A1 7B 8E 89 ..4....Z.-...{..
00000070 21 8E B7 C1 54 4C A8 44 44 9F B5 FA 13 70 51 27 !...TL.DD....pQ'
00000080 D5 A7 A5 76 DC 4D AF 4C 41 85 7D 41 6D 06 DF 37 ...v.M.LA.}Am..7
00000090 5C 5C 30 08 94 4D 4D 34 A9 DB EC 8B 32 29 B0 86 \\0..MM4....2)..
Signed 8 bit: 101
Signed 32 bit: 1524997733
Hexadecimal: 65
Unsigned 8 bit: 101
Unsigned 32 bit: 1524997733
Octal: 145
Signed 16 bit: -24987
Signed 64 bit: 1524997733
Binary: 01100101
Unsigned 16 bit: 40549
Unsigned 64 bit: 1524997733
Stream Length: 8
Float 32 bit: 3.231596e+16
Float 64 bit: -1.430948e-80
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x2F


ciphered_cbc.txt - GHex
00000000 AD 5D 99 EA 17 32 7F 22 6D E5 E1 79 78 DB FE 4A .]...2."m..yx..J
00000010 AC 31 CB E0 A4 C3 C4 66 FC CC EC 73 97 4B C3 C7 .1.....f...s.K..
00000020 04 3B BD 06 3B 76 2B F8 44 1A 86 6E 17 1C 76 62 .;..;v+.D..n..vb
00000030 3E 4F BE D9 EE D6 AB 12 80 60 AE 12 44 61 62 E0 >O.......`..Dab.
00000040 4D 53 82 08 F5 AD 06 7C FE 26 0B EA 30 54 EE 0B MS.....|.&..0T..
00000050 50 99 CE 15 D3 FB 09 7A BE E5 9A A0 19 E4 04 6D P......z.......m
00000060 F9 8A 2E 31 F5 04 50 8B B0 43 E7 FE C3 08 E0 B6 ...1..P..C......
00000070 75 20 C7 F8 CC FB 55 C3 A4 D8 BF A9 DE 7D 91 67 u ....U......}.g
00000080 A1 F1 76 79 F4 F0 2D CD 6E 3D A7 1C 03 66 A6 3F ..vy..-.n=...f.?
00000090 4E 8D A2 53 DE 23 3D 87 32 FB DA 05 95 98 F3 96 N..S.#=.2.......
Signed 8 bit: 28
Signed 32 bit: 1046640156
Hexadecimal: 1C
Unsigned 8 bit: 28
Unsigned 32 bit: 1046640156
Octal: 034
Signed 16 bit: 30236
Signed 64 bit: 1046640156
Binary: 00011100
Unsigned 16 bit: 30236
Unsigned 64 bit: 1046640156
Stream Length: 8
Float 32 bit: 2.211537e-01
Float 64 bit: -9.528897e+225
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x2D


ciphered_ecb.txt - GHex
00000000 C4 69 30 A0 84 AE 70 A9 B7 9D 10 B3 6F 9D E9 0F .i0...p.....o...
00000010 BD D3 ED 6E 35 31 4F 5F 1A 1D E7 6E 29 E1 CB F6 ...n51O_...n)...
00000020 B5 56 DA F6 0F 8F 34 DE 7B 2E 3C 35 4E BC 91 56 .V....4.{.<5N..V
00000030 12 78 52 35 88 31 59 9D 7C D1 E2 0A 1C 65 72 08 .xR5.1Y.|....er.
00000040 7A 22 5C C5 7C 3D E6 DB 46 3F 3A 9F F1 64 6B 9F z"\.|=..F?:..dk.
00000050 50 73 08 6B D3 D3 2B 45 D1 A6 F7 31 B8 0F 4E 08 Ps.k..+E...1..N.
00000060 FB 21 CA F7 13 5E BA 5F 31 CA 2D CD 80 64 7A 43 .!...^._1.-..dzC
00000070 58 D7 AC 00 DD 54 81 F3 7E 01 BF 4E 16 C2 B1 59 X....T..~..N...Y
00000080 65 DF 16 37 13 61 EE 96 BB ED 8C 84 4C 91 A3 A6 e..7.a......L...
00000090 D5 15 B7 04 E0 2F CF C9 78 71 FF 7C BC CF 2F CF ...../..xq.|../.
Signed 8 bit: 8
Signed 32 bit: 1545763336
Hexadecimal: 08
Unsigned 8 bit: 8
Unsigned 32 bit: 1545763336
Octal: 010
Signed 16 bit: 31240
Signed 64 bit: 1545763336
Binary: 00001000
Unsigned 16 bit: 31240
Unsigned 64 bit: 1545763336
Stream Length: 8
Float 32 bit: 1.829325e+17
Float 64 bit: -3.132374e+184
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x3F


ciphered_ofb.txt - GHex
00000000 E0 15 A6 54 6E AF 79 3D 04 5B D9 23 43 59 2F B2 ...Tn.y=.[.#CY/.
00000010 17 1D 96 50 81 C4 1E 9E 8A 38 B7 AD 69 AC 86 8C ...P.....8..i...
00000020 E4 1F 8D CB 77 4B D8 D5 B2 A5 D2 F8 7A 06 CF 43 ....wK......z..C
00000030 90 20 A1 44 EF 03 4D 66 67 E8 A7 D7 B4 8A 5F 50 . .D..Mfg.....P
00000040 2E B4 1D 02 8A CE EA B6 DE AF 76 F7 2F E3 7F 93 ....v./...
00000050 C7 B3 C9 D2 AF F1 18 80 06 66 DB 38 20 BB 88 C8 ....f.8 ...
00000060 16 5D D5 F1 E7 A6 A9 94 3E 7A 4E FE 36 2F 2E 2E .]......>zN.6/..
00000070 59 24 BA B2 CE 5D 1F C9 1F 8B C3 01 C9 2F 86 04 Y$...]......./..
00000080 A2 37 9E B3 A8 F6 9A 1F 64 33 24 BC 4C 40 7C 15 .7......d3$.L@|.
00000090 4D 29 0C BB E5 87 5F 48 25 2D A8 FD A4 EA 92 3B M)...._H%-.....;
Signed 8 bit: -24
Signed 32 bit: -1260935192
Hexadecimal: E8
Unsigned 8 bit: 232
Unsigned 32 bit: 3034032104
Octal: 350
Signed 16 bit: -22552
Signed 64 bit: 3034032104
Binary: 11101000
Unsigned 16 bit: 42984
Unsigned 64 bit: 3034032104
Stream Length: 8
Float 32 bit: -4.016904e-07
Float 64 bit: 1.316907e-85
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x39

```
[11/15/21]seed@VM:~/.../CSS$ openssl enc -aes-128-cbc -e -in hello.txt -out ciphered_cbc.txt -K 00112233445566778899a
abbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../CSS$ openssl enc -aes-128-cfb -e -in hello.txt -out ciphered_cfb.txt -K 00112233445566778899a
abbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../CSS$ openssl enc -aes-128-ofb -e -in hello.txt -out ciphered_ofb.txt -K 00112233445566778899a
abbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../CSS$ openssl enc -aes-128-ecb -e -in hello.txt -out ciphered_ecb.txt -K 00112233445566778899a
abbccddeeff -iv 0102030405060708
warning: iv not use by this cipher
[11/15/21]seed@VM:~/.../CSS$ openssl enc -aes-128-cbc -d -in ciphered_cbc.txt -out ciphered_cbc_o.txt -K 001122334455
66778899aabbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../CSS$ openssl enc -aes-128-cfb -d -in ciphered_cfb.txt -out ciphered_cfb_o.txt -K 001122334455
66778899aabbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../CSS$ openssl enc -aes-128-ecb -d -in ciphered_ecb.txt -out ciphered_ecb_o.txt -K 001122334455
66778899aabbccddeeff -iv 0102030405060708
warning: iv not use by this cipher
[11/15/21]seed@VM:~/.../CSS$ openssl enc -aes-128-ofb -d -in ciphered_ofb.txt -out ciphered_ofb_o.txt -K 001122334455
66778899aabbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../CSS$
```

Right Ctrl

The decrypted text for the same is as follows:

cbc:

plain.txt | Untitled Document | ciphered_cbc_o.txt | ciphered_cfb_o.txt | ciphered_ecb_o.txt | ciphered_ofb_o.txt | hello.txt

```
An initial public offering (IPO) refers to the pa□□oo□□□□lu□µ□□òA□ËOng shares of □□□private corporation to the public in a new stock issuance.
Companies must meet requirements by exchanges and the Securities and Exchange Commission (SEC) to hold an IPO.
IPOs provide companies with an opportunity to obtain capital by offering shares through the primary market.
Companies hire investment banks to market, gauge demand, set the IPO price and date, and more.
An IPO can be seen as an exit strategy for the companyâ□□□s founders and early investors, realizing the full profit from their private investment.
```

cfb:

plain.txt | Untitled Document | ciphered_cbc_o.txt | ciphered_cfb_o.txt | ciphered_ecb_o.txt | ciphered_ofb_o.txt | hello.txt

```
An initial public offering (IPO) refers to thñ!p
□□Aþð□□=N0'□□□□Œæ∪Tqc€ÛBiÇžOÂ·âa□□Y□rivate corporation to the public in a new stock issuance.
Companies must meet requirements by exchanges and the Securities and Exchange Commission (SEC) to hold an IPO.
IPOs provide companies with an opportunity to obtain capital by offering shares through the primary market.
Companies hire investment banks to market, gauge demand, set the IPO price and date, and more.
An IPO can be seen as an exit strategy for the companyâ□□□s founders and early investors, realizing the full profit from their private investment.
```

ecb:

plain.txt | Untitled Document | ciphered_cbc_o.txt | ciphered_cfb_o.txt | ciphered_ecb_o.txt | ciphered_ofb_o.txt | hello.txt

```
An initial public offering (IPO) refers to the pikîŽ~□□òMÙ)□□v□□oN¿ng shares of a private corporation to the public in a new stock issuance.
Companies must meet requirements by exchanges and the Securities and Exchange Commission (SEC) to hold an IPO.
IPOs provide companies with an opportunity to obtain capital by offering shares through the primary market.
Companies hire investment banks to market, gauge demand, set the IPO price and date, and more.
An IPO can be seen as an exit strategy for the companyâ□□□s founders and early investors, realizing the full profit from their private investment.
```

ofb:

plain.txt | Untitled Document | ciphered_cbc_o.txt | ciphered_cfb_o.txt | ciphered_ecb_o.txt | ciphered_ofb_o.txt | hello.txt

```
An initial public offering (IPO) refers to the process Ý
 offering shares of a private corporation to the public in a new stock issuance.
Companies must meet requirements by exchanges and the Securities and Exchange Commission (SEC) to hold an IPO.
IPOs provide companies with an opportunity to obtain capital by offering shares through the primary market.
Companies hire investment banks to market, gauge demand, set the IPO price and date, and more.
An IPO can be seen as an exit strategy for the companyâ□□□s founders and early investors, realizing the full profit from their private investment.
```

Inference:

In the case of ecb mode encryption since we know that each plaintext block is encrypted separately similarly decrypted separately therefore only the block containing the corrupted byte gets corrupted there is no difference in the rest of the text. An advantage of this mode is that since there is no dependency upon other blocks, the encryption and decryption can be carried out by many threads simultaneously.

As we know in the cbc mode the chaining between input and output takes place , the block of plain text is XOR ed with the encrypted block of the previous pass and thus the chain continues. So I inferred and understood that if one bit of the actual plain block is corrupted then the entire chain will have corrupted bits leading to a totally corrupted text, but if say only one bit of the ciphertext is damaged only two received plaintext blocks will be damaged hence making it possible to recover the original data.

The cfb mode is similar to the cbc mode but the only difference being that the ciphertext from the previous round needs to be encrypted and then added to the plaintext bits. Here the same encryption algorithm needs to be used for both encryption and decryption. I observed that after corrupting one ciphertext bit only the two consecutive plaintext blocks will be damaged.

In case of ofb mode the keystream bits are created that are used for the encryption of subsequent data blocks and due to this the working of this mode is similar to a typical stream cipher. In this case I observed that if one bit of a plaintext or ciphertext message is damaged, only one corresponding ciphertext or respectively plaintext bit is damaged as well.

Task4: Padding

Two files namely large.txt and small.txt are made with each having 32 and 20 bytes respectively.
The files are encrypted using aes-128 in 4 modes namely: ecb, cbc, cfb, ofb

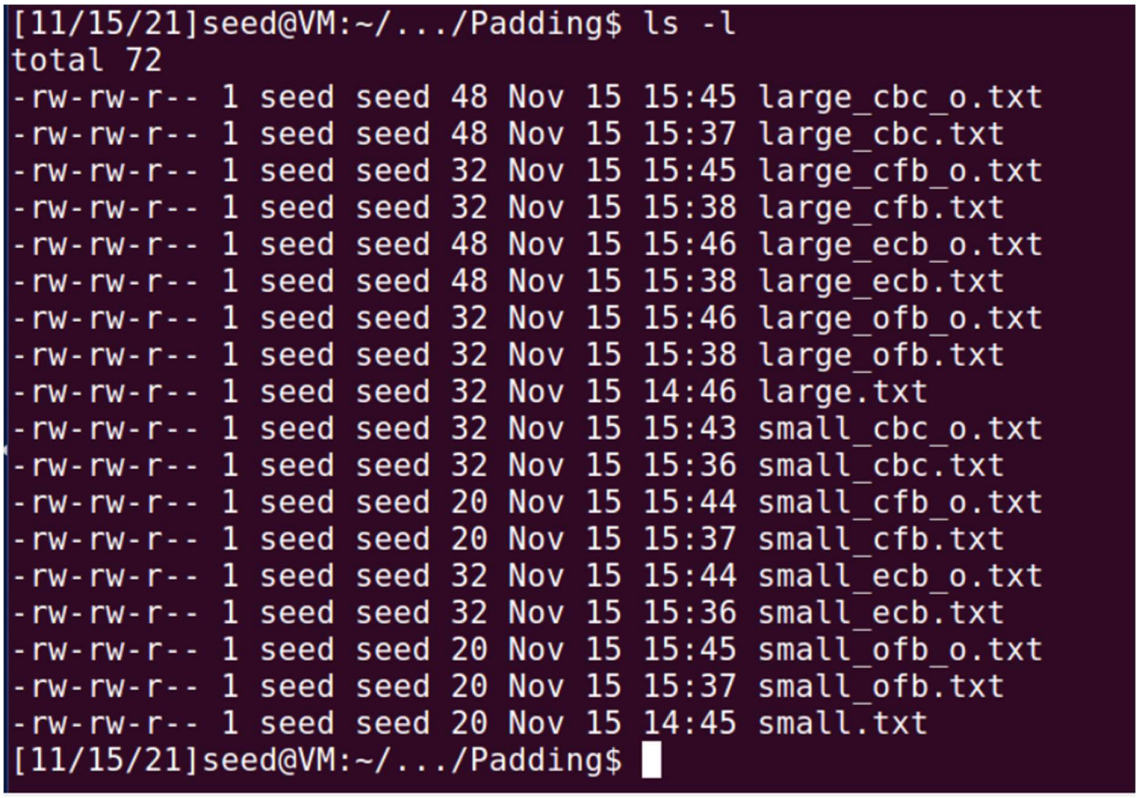
```
[11/15/21]seed@VM:~/.../Padding$ ls -l
total 8
-rw-rw-r-- 1 seed seed 32 Nov 15 14:46 large.txt
-rw-rw-r-- 1 seed seed 20 Nov 15 14:45 small.txt
[11/15/21]seed@VM:~/.../Padding$
```

To confirm that openssl uses PKCS5 padding, decrypt the encrypted file with option–nopad. This option turns off the standard block padding. Normally, the padding is included by default during encryption, so if I use the nopad option, I can see the padding in the decrypted file.

```
[11/15/21]seed@VM:~/.../Padding$ openssl enc -aes-128-cbc -d -nopad -in small_cbc.txt -out small_cbc_o.txt -K 0011223
3445566778899aabbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../Padding$ openssl enc -aes-128-cfb -d -nopad -in small_cfb.txt -out small_cfb_o.txt -K 0011223
3445566778899aabbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../Padding$ openssl enc -aes-128-ecb -d -nopad -in small_ecb.txt -out small_ecb_o.txt -K 0011223
3445566778899aabbccddeeff -iv 0102030405060708
warning: iv not use by this cipher
[11/15/21]seed@VM:~/.../Padding$ openssl enc -aes-128-ofb -d -nopad -in small_ofb.txt -out small_ofb_o.txt -K 0011223
3445566778899aabbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../Padding$ openssl enc -aes-128-cbc -d -nopad -in large_cbc.txt -out large_cbc_o.txt -K 0011223
3445566778899aabbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../Padding$ openssl enc -aes-128-cfb -d -nopad -in large_cfb.txt -out large_cfb_o.txt -K 0011223
3445566778899aabbccddeeff -iv 0102030405060708
[11/15/21]seed@VM:~/.../Padding$ openssl enc -aes-128-ecb -d -nopad -in large_ecb.txt -out large_ecb_o.txt -K 0011223
3445566778899aabbccddeeff -iv 0102030405060708
warning: iv not use by this cipher
[11/15/21]seed@VM:~/.../Padding$ openssl enc -aes-128-ofb -d -nopad -in large_ofb.txt -out large_ofb_o.txt -K 0011223
3445566778899aabbccddeeff -iv 0102030405060708
```

```
[11/15/21]seed@VM:~/.../Padding$ ls -l
total 40
-rw-rw-r-- 1 seed seed 48 Nov 15 15:37 large_cbc.txt
-rw-rw-r-- 1 seed seed 32 Nov 15 15:38 large_cfb.txt
-rw-rw-r-- 1 seed seed 48 Nov 15 15:38 large_ecb.txt
-rw-rw-r-- 1 seed seed 32 Nov 15 15:38 large_ofb.txt
-rw-rw-r-- 1 seed seed 32 Nov 15 14:46 large.txt
-rw-rw-r-- 1 seed seed 32 Nov 15 15:36 small_cbc.txt
-rw-rw-r-- 1 seed seed 20 Nov 15 15:37 small_cfb.txt
-rw-rw-r-- 1 seed seed 32 Nov 15 15:36 small_ecb.txt
-rw-rw-r-- 1 seed seed 20 Nov 15 15:37 small_ofb.txt
-rw-rw-r-- 1 seed seed 20 Nov 15 14:45 small.txt
[11/15/21]seed@VM:~/.../Padding$
```

Finally we can see the decrypted and encrypted files as follows:



Inference:

The screenshot above shows that the size of CBC and ECB encrypted files with the nopad option is 12 bytes more for the 20 bytes file and 16 bytes larger for the 32 bytes file, however the size of OFB and CFB decrypted files is the same.

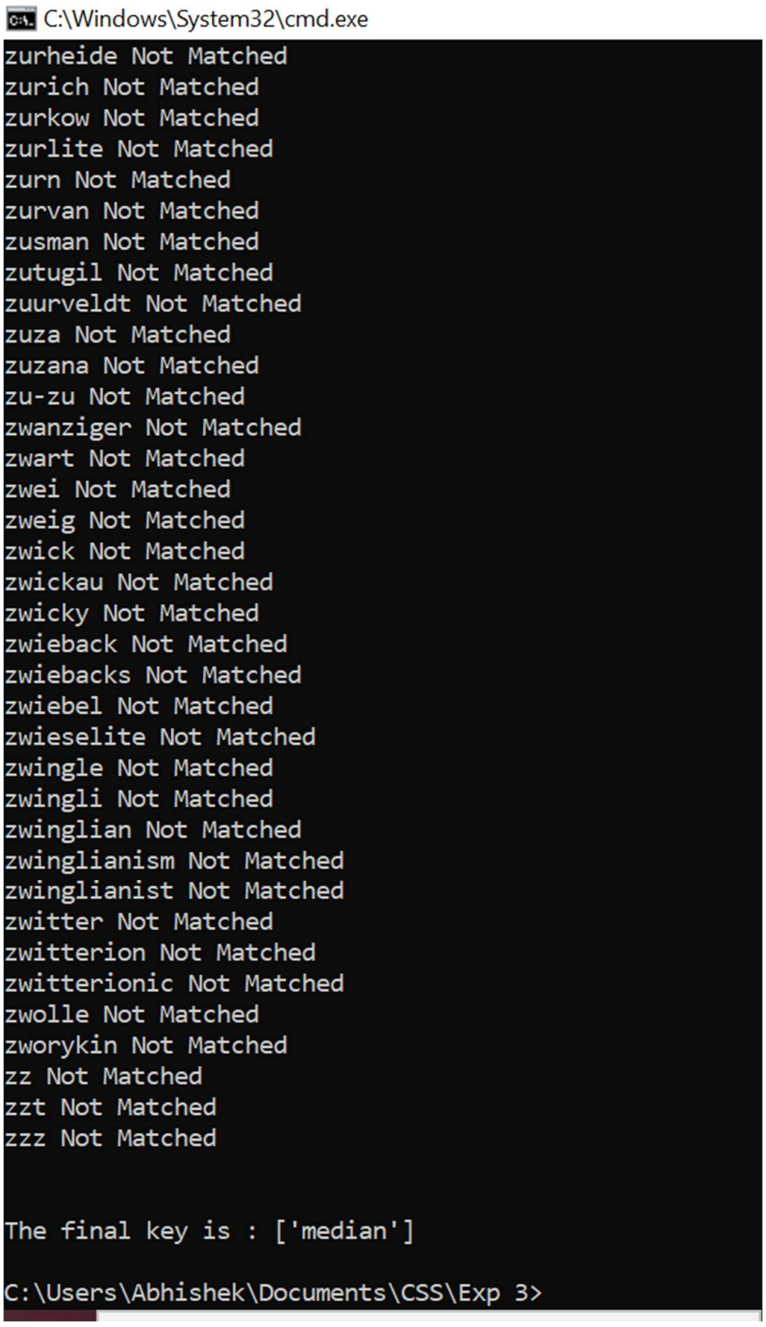
Result:
1. The experiment shows that padding is needed for ECB and CBC encryption modes. This can be because ECB and CBC are block ciphers and for a block cipher length of input must be an exact multiple of block length. If this is not the case then padding must be added to make it so. This padding is removed after decrypting.
2. In OFB and CFB, the padding is not required because they are stream ciphers and the ciphertext is always the same length as plain text.

Task 5: Programming using the Crypto Library

Code:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
plaintText = b"This is a top secret."
cipherText = "8d20e5056a8d24d0462ce74e4904c1b513e10d1df4a2ef2ad4540fae1ca0aaf9"
myFile = open('words.txt', 'r')
lines = myFile.readlines()
words = [str.strip(line) for line in lines]
arr = []
for word in words:
    if len(word)<16:
        word=word.lower()
        key=word.encode()+b' '*(16-len(word))
        getCipher=AES.new(key, AES.MODE_CBC, iv=bytes.fromhex('0'*32))
        ciphertext=getCipher.encrypt(pad(plaintText, AES.block_size))
        match="Not Matched"
        if bytes.hex(ciphertext)==cipherText:
            match="Matched"
            arr.append(word)
        print(word,match)
    print("\n\nThe final key is :",arr)
```

Output:



The key used to encrypt is median.
Here I observed that using the pycryptodome library present in python and with given plain text, cipher text and the iv used I will be able to find the key by brute force approach.

Conclusion:

AES, DES are symmetric key algorithms using the same keys to encrypt and decrypt the data.
ECB mode of encryption is the weakest form of encryption in comparison to CBC, CFB and OFB.
I could conclude from the experiment that ECB and CBC use padding while encryption while the other two don't. This proves that ECB and CBC are block ciphers while CFB and OFB are stream ciphers.
I learned how different modes react to a corrupted bit of a cipher text. The best decryption in such a case is provided by OFB where only the corrupted bit of cipher text is affected while encrypting.
I could conclude from this experiment that, If I have the plaintext, ciphertext and iv known, I can easily find the key using brute force method

Github Link:

https://github.com/AbhishekC20001/CSS-Lab-2019130009