Experiment 4: Crypto Lab – Secret-Key Encryption

Name: Deep Nayak **UID**: 2019130045 TE COMPS

OBJECTIVE

The learning objective of this lab is for students to get familiar with the concepts in the secret-key encryption. After finishing the lab, students should be able to gain a first-hand experience on encryption algorithms, encryption modes, paddings, and initial vector (IV). Moreover, students will be able to use tools and write programs to encrypt/decrypt messages.

LAB TASKS

Task 1: Encryption using different ciphers and modes

In this task, we will play with various encryption algorithms and modes. You can use the following openssl enc command to encrypt/decrypt a file. To see the manuals, you can type man openssl and man enc.

Encryption using different cipher algorithms:

The plain text that is encrypted in every scenario is as follows:

plain_text_1.txt U

CSS-Lab-Deep-Nayak > Experiment 3 > plain_text_1.txt

```
1  Base to all engaged units.
2  Eagle is going to attack land of the rising sun with deadliest weapons in existence.
3  All units are ordered to stand down and stay on alert.
4  In case of any new information, encrypt the same and send it via secure channel.
5  Destroy this message once it has served its purpose.
6  Over and Out.
```

Fig 1.1

1. Encryption using ARIA-128-CBC

Encrypted Text:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
 openssl enc -aria-128-cbc -e -in plain_text_1.txt -out cipher_1.txt -K 00112233445566778899aabb
ccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
 cat cipher_1.txt
<binary garbage>
J+)<binary garbage>s
<binary garbage>
<binary garbage>deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task
```

Fig 1.2

Hex Dump:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task
 hexdump cipher_1.txt
0000000 0c82 aa5c 76ff b93b 6e9c fc3c 210e 54e7
0000010 e405 b071 b0eb 1c54 3d58 2e03 e184 8826
0000020 b725 04c5 6c54 f1a1 1240 14d2 da6e 6e17
0000030 b555 5945 6b49 398b cda2 5faa 4510 8463
0000040 73a1 0592 2778 0e42 ffe3 132d a65e 80a3
0000050 dd71 ec4f 8e8e f2ca 7aeb 9ea1 b46f ef0f
0000060 5c4c 9522 87f6 875e f0e8 ae8f c365 f497
0000070 a29d 4a3c 292b 7385 500c 5fb1 74d6 6a75
0000080 d9a2 3c9c b6e9 a74a 9f7c 4d8b f8da e3c5
0000090 5886 ab0e c46f 5b4b bc3d 029d 4a0f bf08
00000a0 a56d 51b5 0cf9 372b 54ee baa7 f5c2 8c59
00000b0 bec3 8de2 b0d7 4ec7 90c8 b2c2 dc7f 120d
00000c0 0de2 2482 b894 6114 0d69 a5aa 6bde 1274
00000d0 5565 d59e 871c 687c 7241 82d0 4418 f998
00000e0 386a daa7 d5d8 96bd 21e1 1a7c 437f 2681
00000f0 3c20 4374 15bb 17d6 3a4f b52a 6346 1717
0000100 a8b1 626d 4c6a 2c73 f774 3e35 36f7 7d4b
0000110 72a5 8c15 2d6a cc62 3699 2608 8d9f a78f
0000120 52b8 b546 23c9 b5ed 871d cd00 9cf6 51d5
0000130 ebc3 424a 3688 64c1 e05e c28e 1ffa 1b95
0000140
```

Fig 1.3

2. Encryption using ARIA-128-ECB

Encrypted Text:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
openssl enc -aria-128-ecb -e -in plain_text_1.txt -out cipher_1.txt -K 00112233445566778899aabbcc
ddeeff
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
cat cipher_1.txt
o|>N   f e7<0t [_y$m X n7i%g{BGFr   fx-p Z#H AW7IZ5%f6
S!+:o U aS9Xx"S Y Z tV %A 7( k3D<
                          O+l7@+
                                sucT\rflS"5 F8 Y#<l:n_ x gu kGUX` )#+gd2 "E2
)&W x- deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experime
```

Fig 1.4

Hex Dump:

```
hexdump cipher_1.txt
0000000 6fdc 7ce9 a103 8198 bc3e 06ec da96 4e83
0000010 e109 6699 c6bb 3765 d33c a4d0 b413 de74
0000020 b3a2 5f5b 7915 248c 0f1e 6dfa 5897 6e83
0000030 37b3 2569 c79a 7b67 4742 46fa 0972 fc66
0000040 78b6 ea02 a8a4 1092 f8b7 e1bd 2df3 bc7f
0000050 fe91 d770 5ae2 4823 e49a f3cb 5741 37fb
0000060 b28a a31e 1e49 d65a 2535 bc06 c0b7 3666
0000070 962a f847 a529 1a9a db89 f2f5 d867 ab91
0000080 d28c fe47 6fd3 ff40 991b 35c5 8ae6 15d8
0000090 5371 530e 5997 e20e 5ad8 b1f2 b811 5674
00000a0 a31e 25d1 ed41 b61a 3798 2898 e3e5 cff8
00000b0 6b0e d9db d933 449f 113c 0d7f a1b1 53ce
00000c0 2b21 84ed 3afc 6fbd b6bc ef84 ab55 d6dc
00000d0 6100 3953 7858 f0e0 9022 860b 2b4f 186c
00000e0 0f37 d140 0c2b 7573 5463 ca5c 6672 6ca6
00000f0 2253 9eda 15ca 35c8 960f 3846 ad1b f08a
0000100 cb18 59b2 3c23 dfc4 9387 6e3a 9e5f 1dcd
0000110 78ca cc13 7567 8d02 476b 5855 ff60 2988
0000120 e8a6 de23 672b 80d4 8fed 3205 c5de 4522
0000130 2932 a7c1 b3e8 2603 b457 bc78 2dfe abfc
0000140
```

Fig 1.5

3. Encryption using ARIA-128-OFB

Encrypted Text:

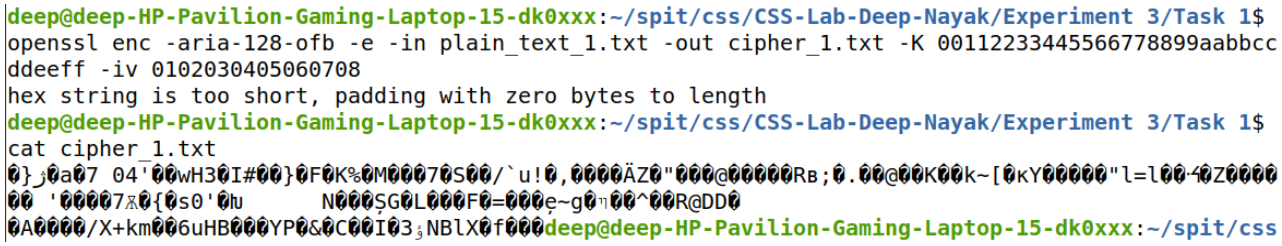


Fig 1.6

Hex Dump:

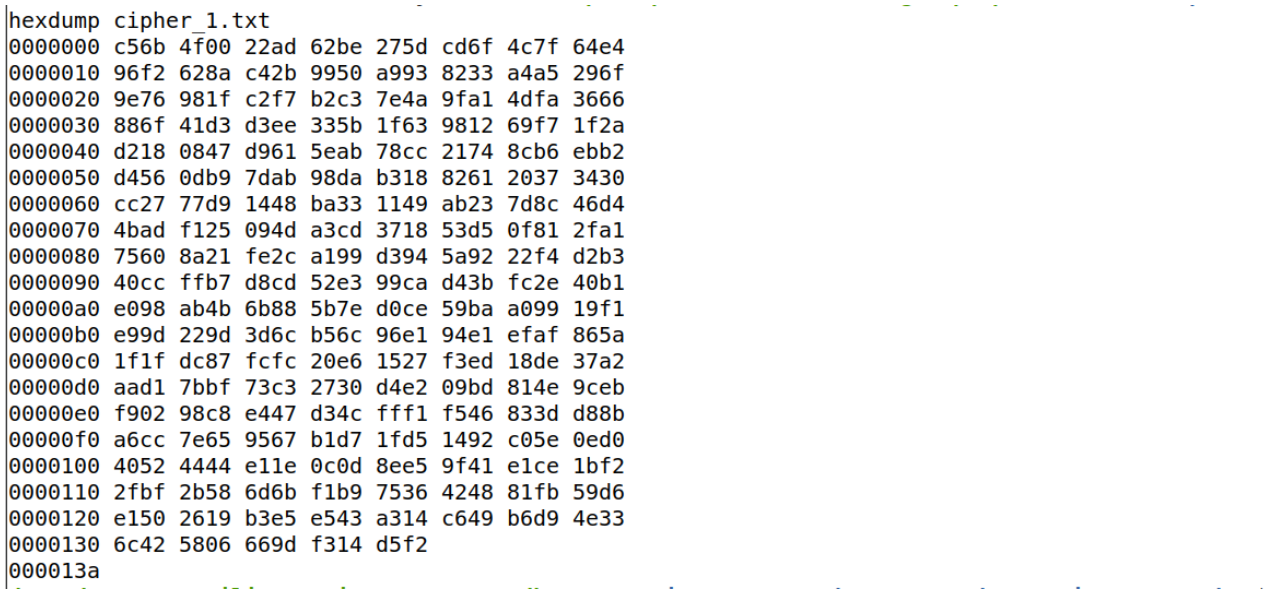


Fig 1.7

4. Encryption using AES-192-CBC

Encrypted Text:

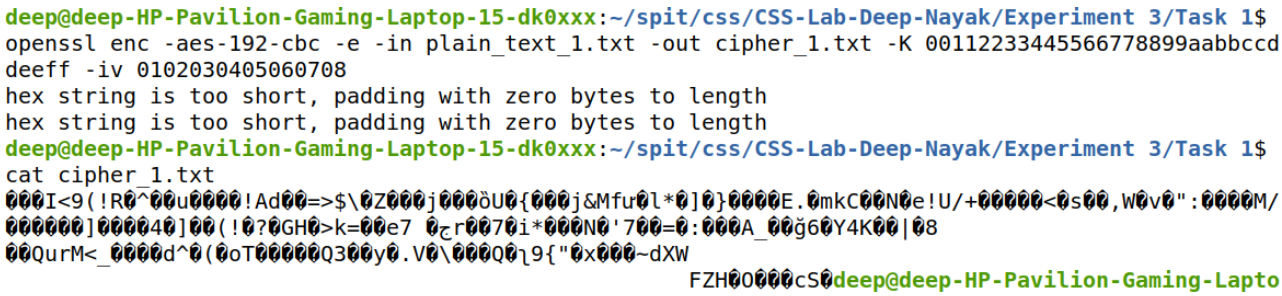


Fig 1.8

Hex Dump:

```
hexdump cipher_1.txt
0000000 9d13 93b5 3c49 2839 1721 1a52 ee5e 75c6
0000010 f780 e78c 4121 6400 c89c 3e3d 5c24 1406
0000020 5aef a814 0839 96f6 8c6a c8e2 8dc8 f455
0000030 fe7b 0ede 6af9 4d26 661c b0c6 6ce6 2a18
0000040 5d9b 7d87 eac9 f1f1 ae45 2e08 6df3 6b00
0000050 ee43 ee06 4e14 1707 81dd 84ee 2165 5518
0000060 2b2f 90f5 1aef 3cc5 73b1 efb1 572c 76b7
0000070 22dc 9a3a c7a3 4ded 192f 6e9f 7285 3774
0000080 aefc 193b 6444 2674 161d f806 1583 1815
0000090 79ab bc27 434a 2b70 534a f212 a70d a785
00000a0 f5b7 a61e a85d f298 34c6 1df8 935d 28b6
00000b0 c921 8d3f 4847 3ef8 3d6b d5d6 3765 a420
00000c0 acd8 a672 37e0 69f2 2a18 aeee 0ec5 4ec4
00000d0 27a2 9c37 c718 933d a13a d0cd 5f41 f219
00000e0 1da4 cc8f 6786 e336 5918 4b34 d2d7 927c
00000f0 0a38 f381 0751 1817 7275 3c4d 1b5f d1e5
0000100 fcda 64e7 ce5e c728 546f b90f ae8a d803
0000110 06c8 3351 9900 7989 2ee9 e056 805c a786
0000120 9f51 85ca 3905 227b 78ba d8b6 f500 7e7f
0000130 5864 5715 460c 485a 4fa4 f0c6 63e0 f453
0000140
```

Fig 1.9

5. Encryption using AES-192-ECB

Encrypted Text:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
openssl enc -aes-192-ecb -e -in plain_text_1.txt -out cipher_1.txt -K 00112233445566778899aabbccd
deeff
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
cat cipher_1.txt
NVz.@$@@nn@)@@@i@+@"g@@C@@@e@@.@@@@@yq4@@x@Os&/$@*M@wx@x%PWJ沦a@@h@@@Dd@@U@@SXmx@b/4D@@@@@@"@@ù!@
^~3?@@;@@@)I@@@1@@@@%n@(@b5@gG@@S>&@
@e@deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task
```

Fig 1.10

Hex Dump:

```
hexdump cipher_1.txt
0000000 fc2f 5811 1fb4 5a74 7b02 899a 7694 6fc3
0000010 b51c 12b2 99f9 24d1 77a4 65a4 40bd 3472
0000020 47dc 830e 6f4e 0d44 ce20 8e68 f16d 1e06
0000030 36c3 3730 0909 2137 0d70 c930 5a78 8bee
0000040 bb45 d2c3 775a 7342 d540 3669 b989 b764
0000050 f7a2 eb1f 6401 5925 12f9 67c1 7ee5 f312
0000060 e604 615a adc2 cd51 b47c 9046 2028 0393
0000070 6f7b 119c a768 f633 ad4e b5de af14 0612
0000080 7631 1f29 b96e 52e7 0d2d 564e 2e7a 24f1
0000090 db9d 6e6e 29ce cb97 f909 f82b 6722 ef05
00000a0 43fb d9ad 8e7f 6505 cb94 82db b7a7 b914
00000b0 98eb 79f4 3471 f088 780f 4f97 2673 242f
00000c0 2aa9 9cd0 77e3 b778 2578 5750 4a00 b2e6
00000d0 61a6 1488 68cd 95b2 44cd e564 11e4 8f55
00000e0 53bd 6d58 c778 a6cf 342f 9a44 bce6 ba1b
00000f0 ce9a 9582 22d6 f0f9 9cc7 db21 5e03 7e14
0000100 3f33 cd9b 993b f502 298f fa49 9702 3188
0000110 a3ea 40e2 2598 d519 ffa8 2812 62cb a635
0000120 1067 a347 53a2 1d3e ea26 0a18 a6ff 1db4
0000130 0310 e73f c114 e42b 3ca3 b8ee c50d c565
0000140
```

Fig 1.11

6. Encryption using AES-192-OFB

Encrypted Text:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
openssl enc -aes-192-ofb -e -in plain_text_1.txt -out cipher_1.txt -K 00112233445566778899aabbccd
deeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
cat cipher_1.txt
[illegible]
[illegible]
T[illegible]r[illegible]2Q=9y][illegible]*[illegible]u[illegible]7[illegible]*L[illegible]*[illegible]vAC~[illegible]Rp[illegible]_*[illegible]Ac[illegible]!;gM[illegible]94[illegible]
[illegible]L[illegible]?[illegible]!uM2[illegible]gp[illegible]Rũ[illegible]=[illegible]r> wi,)Q[illegible],5[illegible]4 Fdeep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS
```

Fig 1.12

Hex Dump:

```
hexdump cipher_1.txt
0000000 25e5 965f f334 dcb7 ebe3 549e fbb1 55e1
0000010 9181 a95b 108a 9bb2 e607 9c25 fca3 71a6
0000020 5c9f 39ca 8525 d7fc 4c3d d8b8 2cf7 a949
0000030 a0d5 ceb0 1065 6aca 4568 3efa 48ea e89a
0000040 fcc6 fb30 0afa 8860 a694 a520 3d5d 1d8e
0000050 163e e19a 3ff0 bb93 c9d7 413a 830c 387c
0000060 678a 0947 b28b 7045 f33f 841d c624 f6c7
0000070 f5ec 34cb c58e 6802 99ce a4f7 a95b 682e
0000080 e905 68f6 7b61 fd8c 8e32 e49d ae87 a43d
0000090 f84d fb78 af40 1c5f 6835 1092 ee0c f710
00000a0 c9e8 8d3f 41eb 9478 0e34 1a34 2b89 c3b7
00000b0 6120 76c6 5133 393d 5d79 b8b3 d02a fd75
00000c0 8583 a4aa 7fed c437 2af1 e74c 2a16 11ed
00000d0 96b8 4176 4317 9c7e cee5 7052 a106 2a5f
00000e0 bc95 6341 90ea 9221 673b 804d 3918 b834
00000f0 8e0d b9ba 540d 72d8 ba5b 0b32 d894 df74
0000100 90a4 c28e e6f5 7a84 887b 7f5e 0de4 4c8e
0000110 d8fe 823f 1c21 17fa 0375 324d 6780 b170
0000120 a3bf c552 f1a9 3d83 f112 3e72 7720 1569
0000130 292c c351 352c 34a6 4620
000013a
```

Fig 1.13

7. Encryption using CAMELLIA-256-CBC

Encrypted Text:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
openssl enc -camellia-256-cbc -e -in plain_text_1.txt -out cipher_1.txt -K 001122334455667788 99aa
bbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
cat cipher_1.txt
[illegible]
[illegible]
[illegible]
```

Hex Dump:

```
hexdump cipher_1.txt
0000000 4586 2e69 d38b db51 b8f6 11a9 29cf b473
0000010 990a fc97 6c3f dd31 32b5 f24e fd0e 747a
0000020 ba7e a57f c57c 651e c76d fb63 7ba0 abbc
0000030 752e c89c f1b7 f890 78ba f592 8e99 df00
0000040 ef70 59fc ae9c 4533 bca8 0709 c8bc e7fa
0000050 178e d3d6 2116 f9d1 f970 624e 7856 9c9a
0000060 94d4 0374 ffba d686 75e1 9708 d8a4 acc1
0000070 c95f 9167 2a90 1f12 5239 494e 4c48 ca32
0000080 9638 15b0 919c 9a4d 0e08 0f24 e28b 09cf
0000090 51df 1d60 bf58 e3d1 6b8c 139c af05 70b8
00000a0 c817 197e a646 f2a6 6ba7 2576 ca35 30ed
00000b0 5af6 fbcf b882 7f93 8af8 4701 e5cf 7570
00000c0 e6b5 2731 4019 a473 4595 4743 618d 609b
00000d0 f9ad 2da4 adb0 90c6 2c0c 2fe1 f33f 18e8
00000e0 9586 b8a4 f6e3 5709 bcfe 9f73 a9d2 77f2
00000f0 4f7e 5e06 b682 5655 103a fcb1 0456 78f1
0000100 db19 cefc 9e8b ba35 0476 5ae3 89cf d4c5
0000110 b28f 2fd9 d51c 955b b3c9 c2c7 f859 f61c
0000120 2527 6362 8b28 ebbe 4759 4e72 9fc4 2efb
0000130 6ba2 29bd 72c9 e445 e344 f0ef 0828 7f59
0000140
```

Fig 1.15

8. Encryption using CAMELLIA-256-ECB

Encrypted Text:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
openssl enc -camellia-256-ecb -e -in plain_text_1.txt -out cipher_1.txt -K 00112233445566778899aa
bbccddeeff
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
cat cipher_1.txt
_ ??????+???T=;j6???U??t??
m??3[oì???1????QS=??/DpK????rs?Y`?? :|????y!?L?ğ?>F?W\P?{?YZ-]??{
?R@3?[???mq?   x?dj{??G???c^Y???O?éf?I?U???    ?yX?
?o2???/??*L?t??y?Wc?.#?.IELd?^??XmFw,3???_??^??z?????y???Y??n????3A?D??
                                                                        ]U??????F?????P???=5H?
aj2G?2*?%??l??*eè??`?deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/
```

Fig 1.16

Hex Dump:

```
hexdump cipher_1.txt
0000000 b1ef 8f47 9fac bae7 3caf 5e58 b788 5f0d
0000010 11b3 93bc 9c03 ce1a 9c2b c10f b710 fa54
0000020 3d08 6a3b 3614 c6b9 55d0 f1fa bc74 0a8d
0000030 f76d 3381 6f5b c302 82ac fb12 31b6 00c5
0000040 a205 b7bb 5351 c53d 2fda 7044 164b 96a9
0000050 f71d dbdd 73b2 59e4 8e60 20a0 7c3a f2d0
0000060 8ffc 2179 4ce9 8c1f 9fc4 3ee4 fb46 5c57
0000070 d350 7b06 59d9 2d5a 9f5d 7bdd d70a e392
0000080 4052 331c 5b83 e098 6d9b 0671 09fe be78
0000090 6a64 c37b 17d4 f647 c1cd 5e63 b659 b8ed
00000a0 f14f bbac a9c3 9b66 bc49 a455 bc82 8d09
00000b0 5879 0bab 2786 948f 0da9 b219 de6f cab8
00000c0 f015 ccaa 2f9e a218 ddd1 8232 4c2a 74c0
00000d0 b7ab ef79 6357 2ecb af23 492e 4c45 c964
00000e0 9c5e 58a6 056d 7746 332c cd9b 5fd6 89e0
00000f0 125e e29c b47a 89f9 8a91 7906 ee95 59db
0000100 db83 d36e d5d1 3fcc 4133 44d6 9087 5d0b
0000110 e055 eefb bba5 d281 9a11 db46 e7f5 e7e0
0000120 9fde 8050 e192 353d a648 6a61 4732 3281
0000130 972a ba25 6cc1 ffe0 652a 90d1 e0ec e160
0000140
```

Fig 1.17

9. Encryption using CAMELLIA-256-OFB

Encrypted Text:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
openssl enc -camellia-256-ofb -e -in plain_text_1.txt -out cipher_1.txt -K 00112233445566778899aa
bbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 1$
cat cipher_1.txt
"}��z��]5|
��1Q��D$$�]Yba���7���e�|�~[<<M@z@@'ڰ`(@@i@@z@,@
g@@@nC@yb @L@n@@4@@@@Z;@f3@@&_@@@@vl@@@@X'o@@@_@$@J@@ ;@@e@~I@@Q@@svB@)@@@~`'u8w@d@@@Ii@@@@6@C@}@
```

Fig 1.18

Hex Dump:

```
hexdump cipher_1.txt
0000000 7d22 7a9a 15ed 5d80 7c35 b90b 6a25 41ba
0000010 004c 23bf 0662 3a63 5399 138a 3a22 4b96
0000020 1bcb 50f7 7ab6 fba5 d827 c9ab 2860 a3a6
0000030 9e69 7ad6 2cea 0dc2 fcf4 aee5 832f 1216
0000040 3b4f 515a 1eec a2b1 379b 630d 65f7 559b
0000050 481b 3ba0 4ea1 20fc 6c12 76ca ce42 e58d
0000060 f313 598f f62f 1bd5 053f 05a0 7a6c 7fcc
0000070 b215 d95c 518e 980d 82ba 65cd 95b6 df5f
0000080 595d 6162 86b0 37d0 dadb 910e fe65 a47c
0000090 5b7e ddf4 d683 e24d b30d 31fc 9ad4 4496
00000a0 24a1 0b89 efb8 0df2 6fee 529a 268d 580f
00000b0 8941 f1fa 764b c556 64be 0d64 af67 b7bd
00000c0 436e 79cb 2062 c4f9 17b9 6eaf e4d3 f334
00000d0 f885 b591 3b5a 6699 e833 26db d95f e1d5
00000e0 76ff b56c adf2 f39e 58ec 2703 976f d9b4
00000f0 cf5f 1f15 0224 0fa4 b64a 20d8 3b10 82a5
0000100 cc65 497e e1d1 0551 d807 73c5 4276 de10
0000110 f029 8b1a 027e 2760 3875 c577 e464 96e7
0000120 49ca f969 e919 dbc8 ec36 4380 7dbf 3ec5
0000130 7592 bdfe ac7f 5a51 0d07
000013a
```

Fig 1.19

Observations:

1. Camellia is Japan's first 128 bit block cipher. In terms of security it is equivalent or even more secure than triple DES. A single DES encryption is vulnerable to brute force attacks however since Camellia matches the standards of triple DES and AES it is very secure. Encryption processing and decryption processing are achieved using the same structure in Camellia, hence, it exhibits superior performance particularly in IC cards that have a low capacity memory or compact hardware. It has either 18 rounds (when using 128-bit keys) or 24 rounds (when using 192- or 256-bit keys)
2. Aria is also a 128 bit block cipher developed by Korean scientists. It uses a substitution-permutation network (SPN) structure based on AES. ARIA uses two 8×8-bit S-boxes and their inverses in alternate rounds. The number of rounds is 12, 14, or 16, depending on the key size
3. AES comprises a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys.

Task 2: Encryption Mode – ECB vs. CBC

The file pic original.bmp contains a simple picture. We would like to encrypt this picture, so people without the encryption keys cannot know what is in the picture. Please encrypt the file using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes, and then do the following:

1. Let us treat the encrypted picture as a picture, and use a picture viewing software to display it. However, For the .bmp file, the first 54 bytes contain the header information about the picture, we have to set it correctly, so the encrypted file can be treated as a legitimate .bmp file. We will replace the header of the encrypted picture with that of the original picture. You can use the ghex tool to directly modify binary files.
2. Display the encrypted picture using any picture viewing software. Can you derive any useful infor-mation about the original picture from the encrypted picture? Please explain your observations.

Original Image:

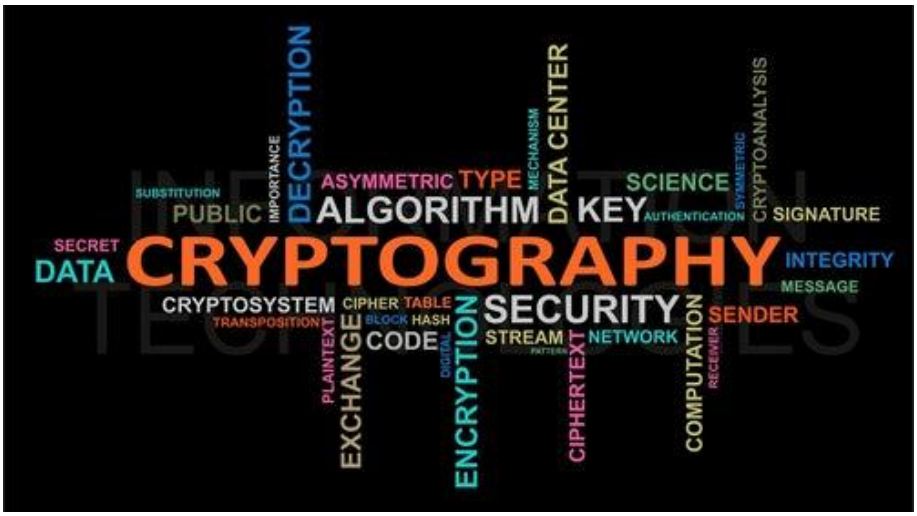


Fig 2.1

Method used to encrypt bmp file using AES-128-ECB and copy valid header:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 2$
head -c 54 cryptography.bmp > enc1.bmp
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 2$
openssl enc -aes-128-ecb -e -in cryptography.bmp -out temp.bin -K 00112233445566778899aabbccddeef
f -iv 0102030405060708
warning: iv not used by this cipher
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 2$
ls -l
total 1120
-rw-rw-r-- 1 deep deep 556138 Nov  8 14:43 cryptography.bmp
-rw-rw-r-- 1 deep deep  26086 Nov  8 14:42 cryptography.jpeg
-rw-rw-r-- 1 deep deep     54 Nov  8 15:06 enc1.bmp
-rw-rw-r-- 1 deep deep 556144 Nov  8 15:07 temp.bin
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 2$
tail -c 556084 temp.bin >> enc1.bmp
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 2$
```

Fig 2.2

Encrypted Image using ECB:

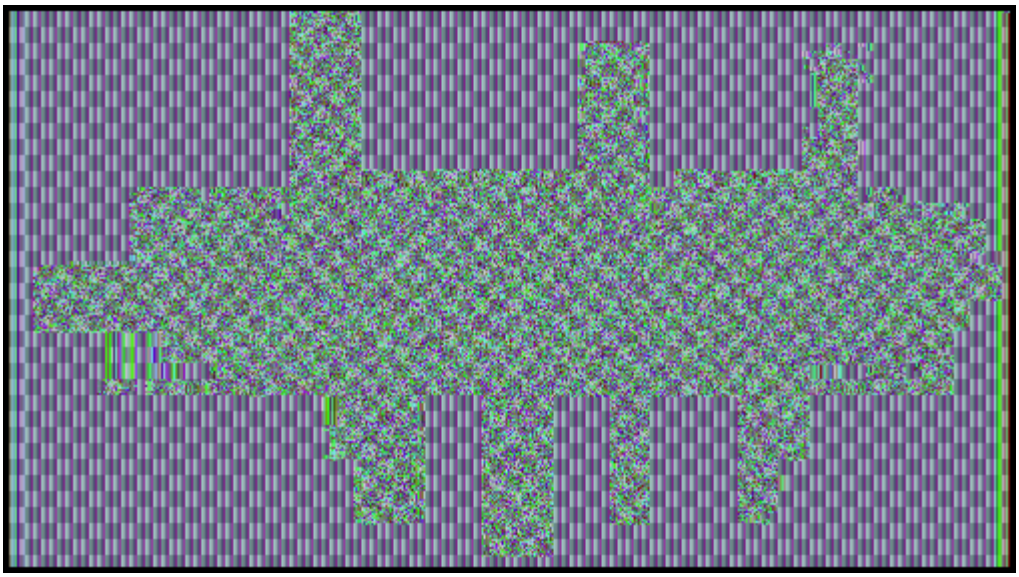


Fig 2.3

Method used to encrypt bmp file using AES-128-ECB and copy valid header:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 2$
head -c 54 cryptography.bmp > enc2.bmp
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 2$
openssl enc -aes-128-cbc -e -in cryptography.bmp -out temp.bin -K 00112233445566778899aabbccddeef
f -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 2$
tail -c 556084 temp.bin >> enc2.bmp
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 2$
```

Fig 2.4

Encrypted image using CBC:

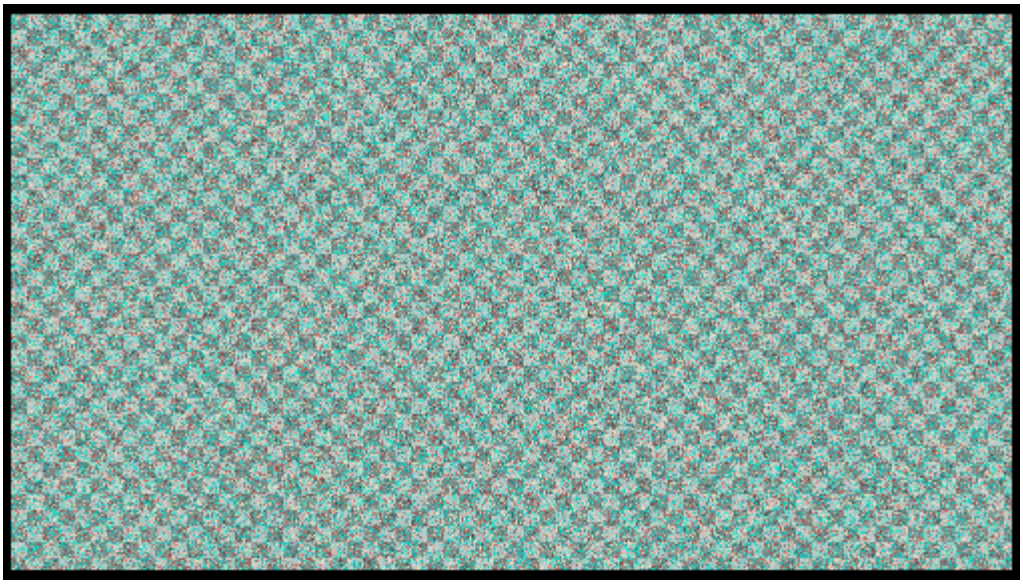


Fig 2.5

Observations:

1. In ECB, the overall structure of the original image is still maintained in the encrypted image. This means that the original image can be guessed to some extent when ECB is used. This mainly happens because we use the same key on each and every block and get the output for each of them.

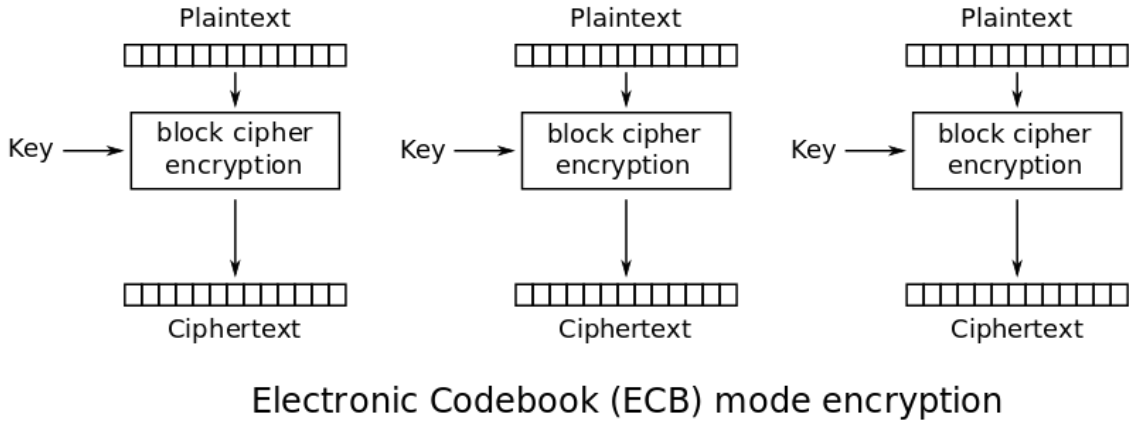


Fig 2.6 | *Credits: Wikipedia*

2. In CBC, the overall structure of the original image is not transferred to the encrypted image. This means that it is impossible to guess the original image based on the encrypted image. This mainly happens because in CBC, we XOR every block with the last encrypted block and then encrypt the current block, this helps in breaking patterns in the data and makes it more secure.

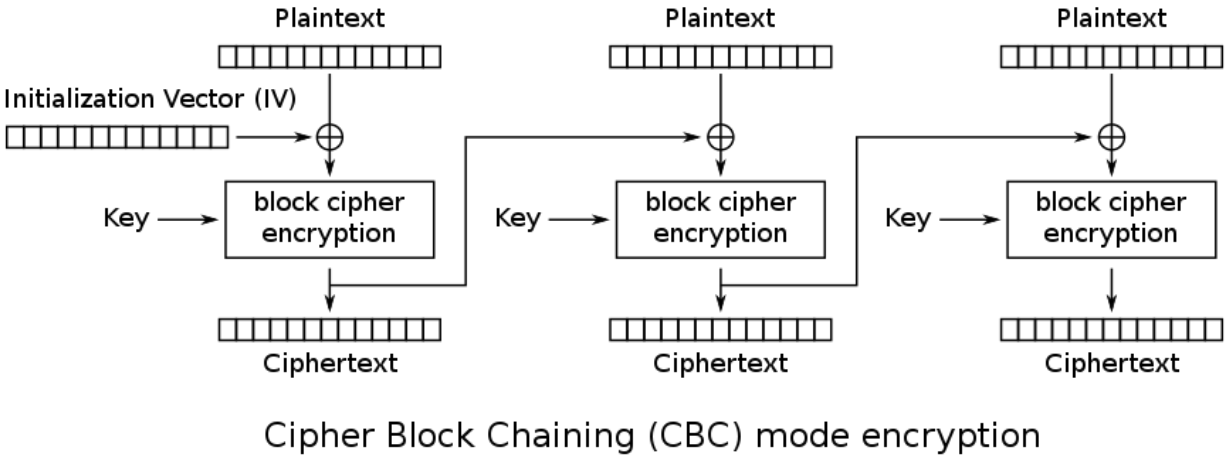


Fig 2.6 | *Credits: Wikipedia*

Task 3: Encryption Mode – Corrupted Cipher Text

To understand the properties of various encryption modes, we would like to do the

following exercise:

1. Create a text file that is at least 64 bytes long.
2. Encrypt the file using the AES-128 cipher.
3. Unfortunately, a single bit of the 30th byte in the encrypted file got corrupted. You can achieve this corruption using ghex.
4. Decrypt the corrupted file (encrypted) using the correct key and IV.

Please answer the following questions: (1) How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively? Please answer this question before you conduct this task, and then find out whether your answer is correct or wrong after you finish this task. (2) Please explain why. (3) What are the implications of these differences?

Original Text:

plain_text_1.txt U ×

CSS-Lab-Deep-Nayak > Experiment 3 > plain_text_1.txt

```
1  Base to all engaged units.
2  Eagle is going to attack land of the rising sun with deadliest weapons in existence.
3  All units are ordered to stand down and stay on alert.
4  In case of any new information, encrypt the same and send it via secure channel.
5  Destroy this message once it has served its purpose.
6  Over and Out.
```

Fig 3.1

AES-128-ECB

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$
openssl enc -aes-128-ecb -e -in plain_text_1.txt -out encrypted_ecb.txt -K 00112233445566778899aa
bbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$
bless encrypted_ecb.txt
```

Fig 3.2

Corrupting the 30th Byte

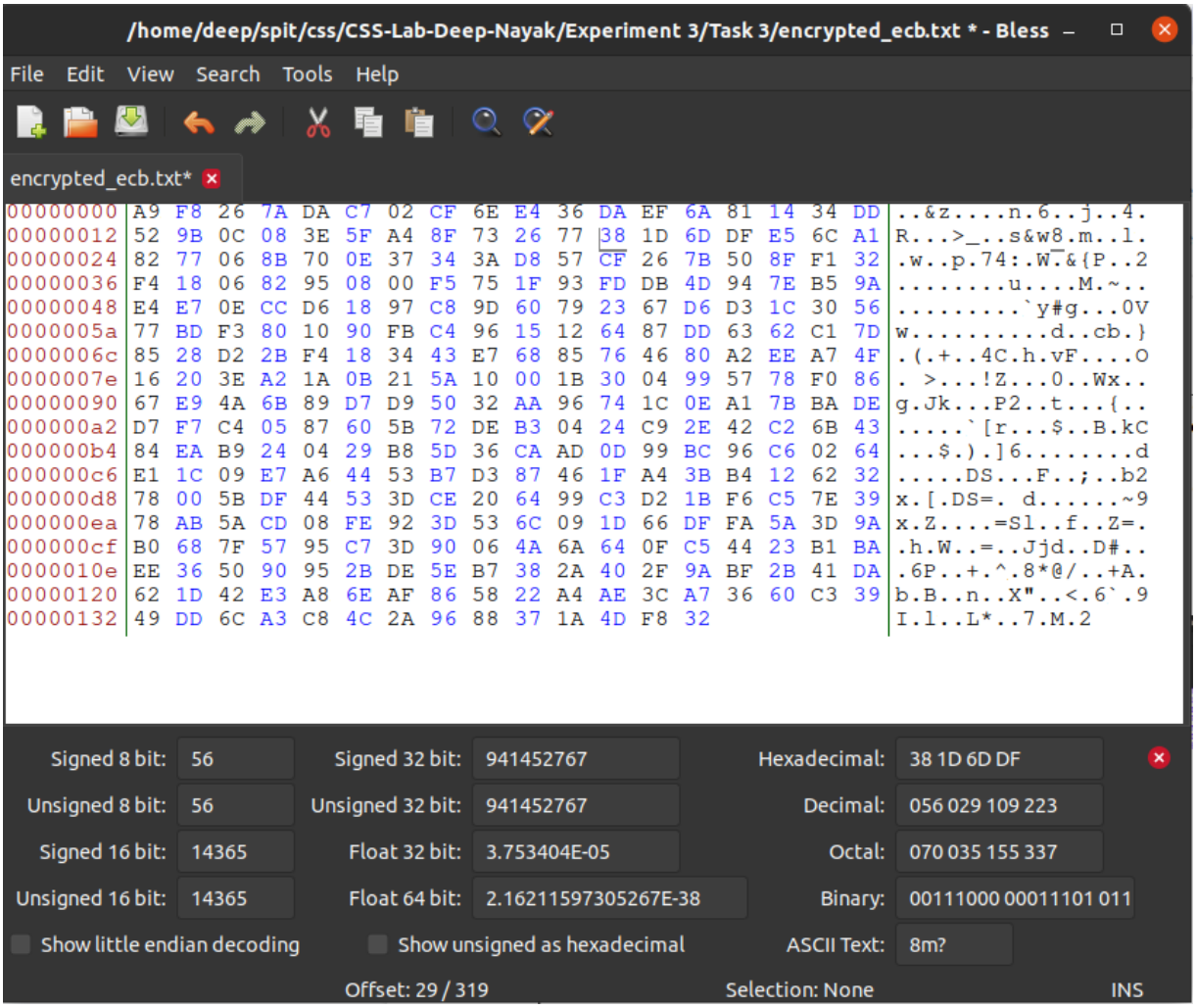


Fig 3.3

Decrypted text:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$ openssl enc -aes-128-ecb
 -d -in encrypted_ecb.txt -out decrypted_ecb.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$ cat decrypted_ecb.txt
Base to all enga▯▯3▯▯,▯udX▯▯!▯ is going to attack land of the rising sun with deadliest weapons in existence.
All units are ordered to stand down and stay on alert.
In case of any new information, encrypt the same and send it via secure channel.
Destroy this message once it has served its purpose.
Over and Out.deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$
```

Fig 3.4

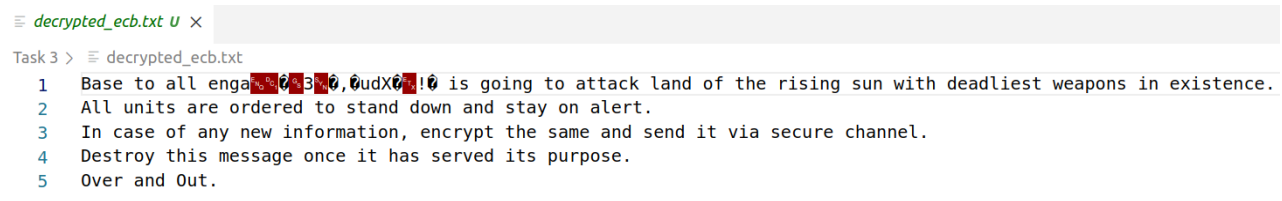
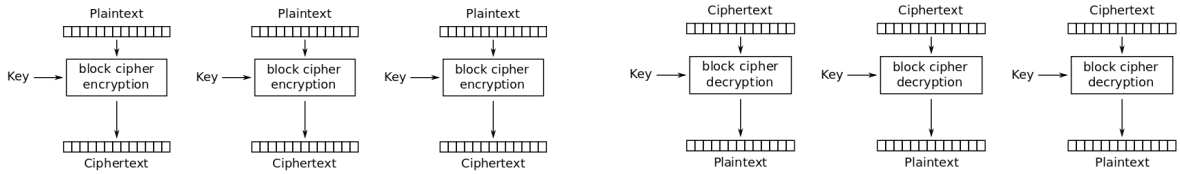


Fig 3.5

Observations:



Electronic Codebook (ECB) mode encryption

Electronic Codebook (ECB) mode decryption

Fig 3.6 | *Credits: Wikipedia*

Since every block cipher is encrypted separately using the key, when one byte is corrupted and then decrypted, it only affects the corresponding block of data while the other blocks of data are unaffected.

AES-128-CBC

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$ openssl enc -aes-128-cbc
 -e -in plain_text_1.txt -out encrypted_cbc.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$ bless encrypted_cbc.txt
```

Fig 3.7

Corrupting the 30th Byte

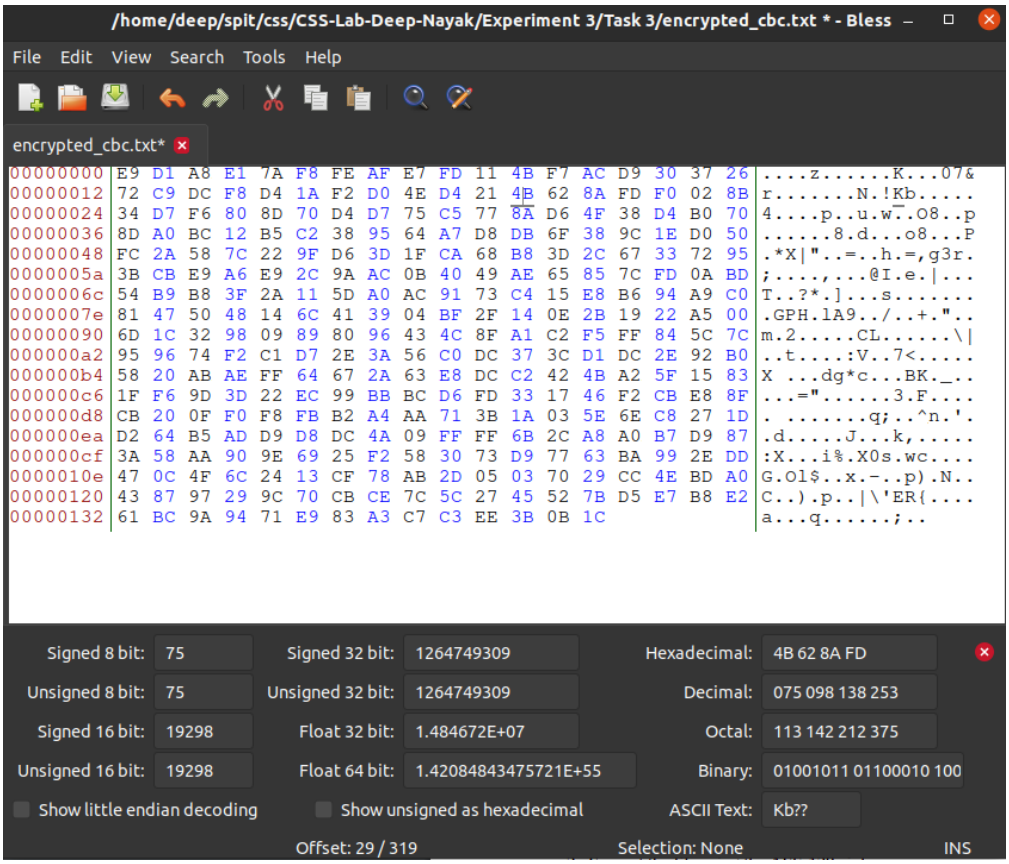


Fig 3.8

Decrypted text:

decrypted_cbc.txt U

Task 3 > decrypted_cbc.txt

```
1  Base to all engaX6q5I L0k is going to
2  ttack land of the rising sun with deadliest weapons in existence.
3  All units are ordered to stand down and stay on alert.
4  In case of any new information, encrypt the same and send it via secure channel.
5  Destroy this message once it has served its purpose.
6  Over and Out.
```

Fig 3.9

Observation:

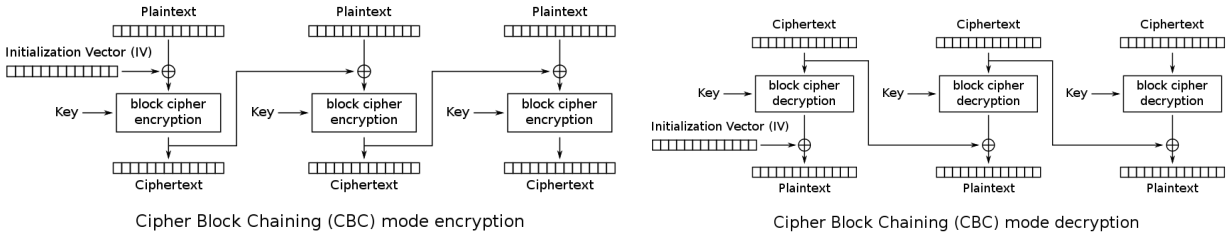


Fig 3.10 | *Credits: Wikipedia*

In CBC every encrypted block cipher depends on the previous encrypted block cipher. However while decrypting, we perform XOR on the decrypted block by using the previous cipher text. Hence a one-bit change to the ciphertext causes complete corruption of the corresponding block of plaintext, and inverts the corresponding bit in the following block of plaintext, but the rest of the blocks remain intact.

AES-128-CFB

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$ openssl enc -aes-128-cfb -e -in
 plain_text_1.txt -out encrypted_cfb.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$ bless encrypted cfb.txt
```

Fig 3.11

Corrupting the 30th Byte

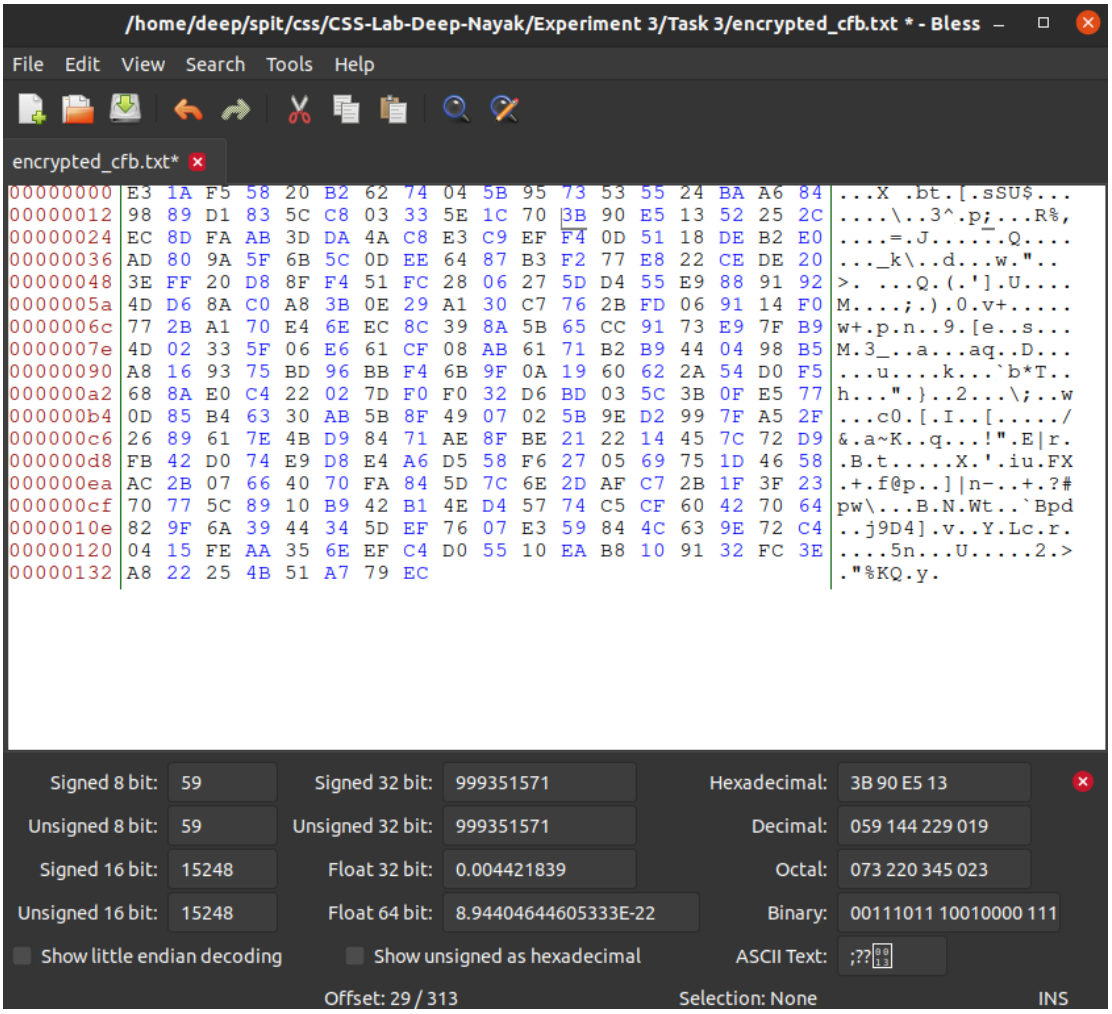


Fig 3.12

Decrypted text:

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$ openssl enc -aes-128-cfb -d -in
 encrypted_cfb.txt -out decrypted_cfb.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

Fig 3.13

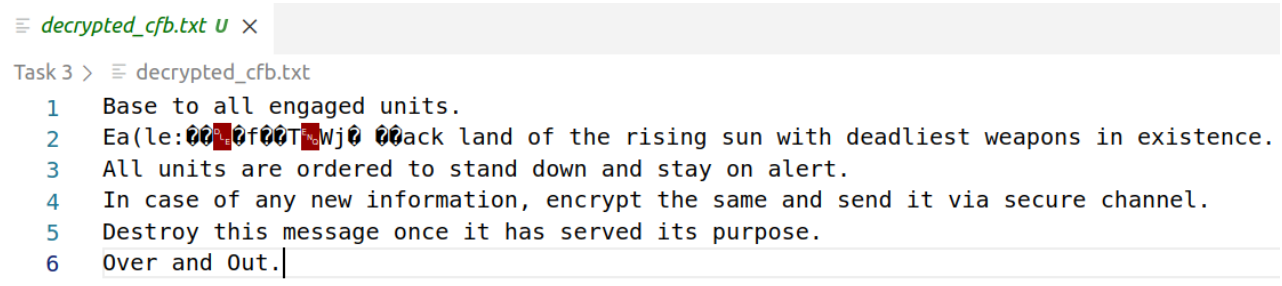


Fig 3.14

Observation:

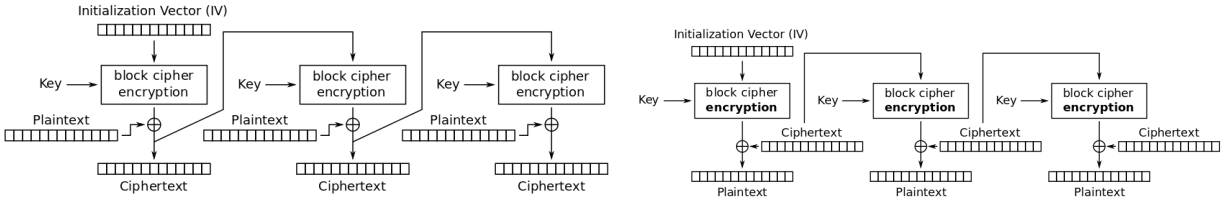


Fig 3.15 | *Credits: Wikipedia*

Loss in CFB is much less as compared to ECB and CBC. This is because the same encryption algorithm is used for decryption and the corrupted cipher text for a block is used for two corresponding blocks hence limiting the damage to two blocks of cipher text.

AES-128-OFB

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$ openssl enc -aes-128-ofb -e -in
 plain_text_1.txt -out encrypted_ofb.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 3$ bless encrypted_ofb.txt
```

Fig 3.16

Corrupting the 30th Byte

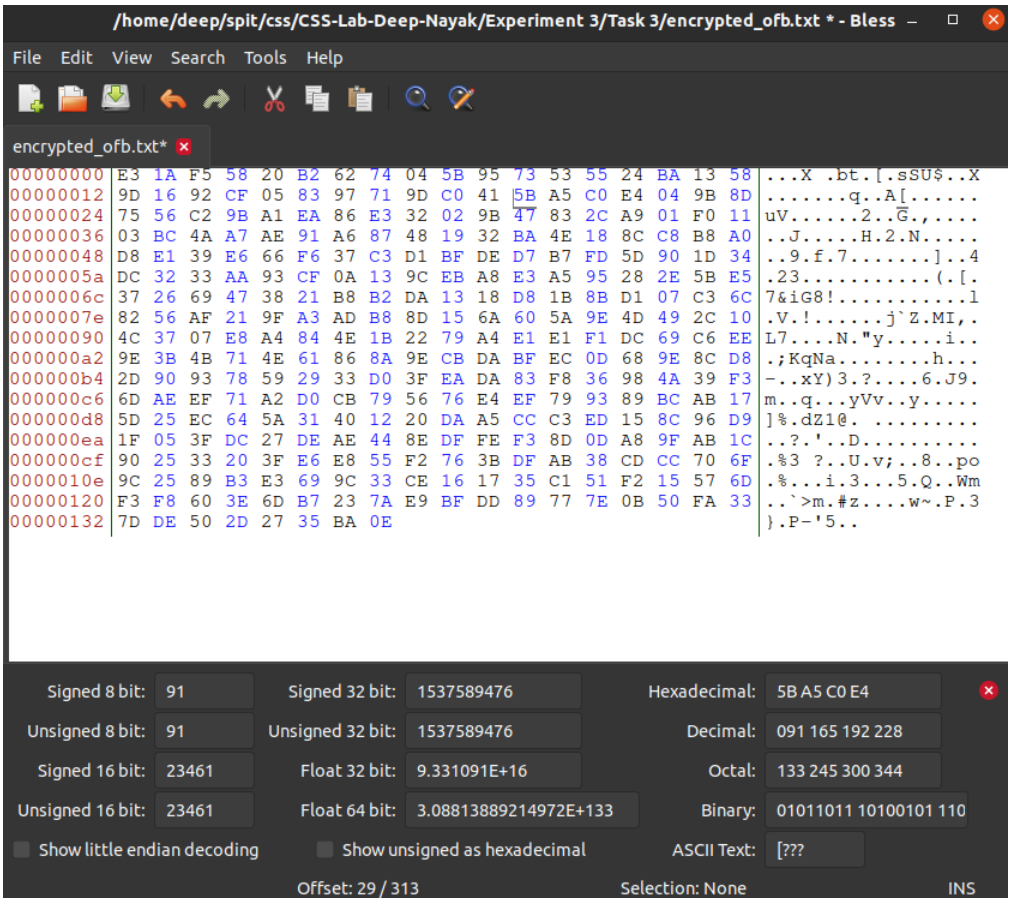


Fig 3.17

Decrypted text:

decrypted_ofb.txt U

Task 3 > decrypted_ofb.txt

```
1  Base to all engaged units.
2  Eagle is going to attack land of the rising sun with deadliest weapons in existence.
3  All units are ordered to stand down and stay on alert.
4  In case of any new information, encrypt the same and send it via secure channel.
5  Destroy this message once it has served its purpose.
6  Over and Out.
```

Fig 3.18

Observation:

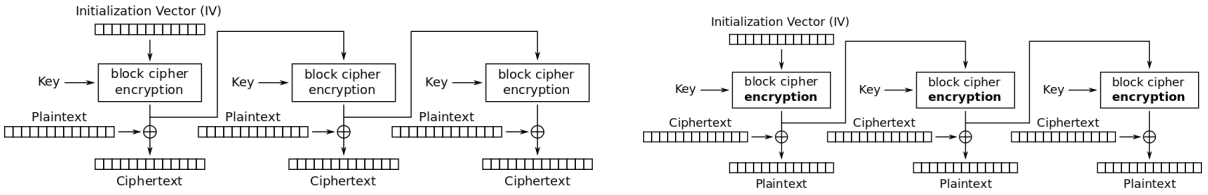


Fig 3.19 | *Credits: Wikipedia*

The data lost in OFB is the least as compared to other block cipher modes. This is mainly because flipping a bit in the ciphertext produces a flipped bit in the plaintext at the same location. This property allows many error-correcting codes to function normally even when applied before encryption.

Task4 : Padding

For block ciphers, when the size of the plain text is not the multiple of the block size, padding may be required. In this task, we will study the padding schemes. Please do the following exercises:

1. The openssl manual says that openssl uses PKCS5 standard for its padding. Please design an experiment to verify this. In particular, use your experiment to figure out the paddings in the AES encryption when the length of the plaintext is 20 octets and 32 octets.
2. Please use ECB, CBC, CFB, and OFB modes to encrypt a file (you can pick any cipher). Please report which modes have paddings and which ones do not. For those that do not need paddings, please explain why.

Original Files:

Small.txt

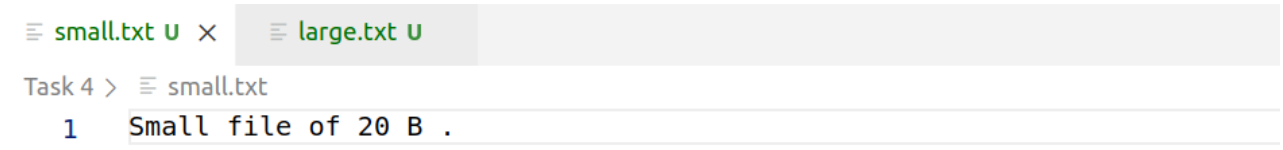


Fig 4.1

Large.txt

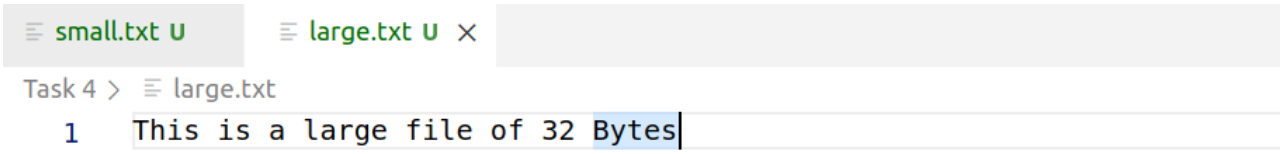


Fig 4.2

Encrypting using different Encryption methods

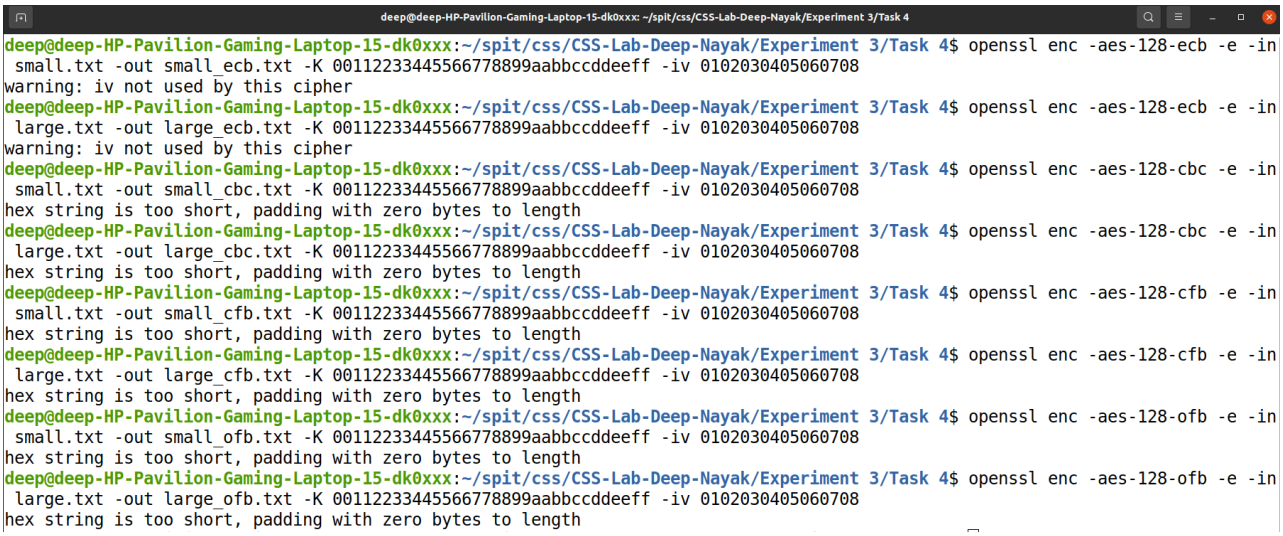


Fig 4.3

Decrypting using nopad option

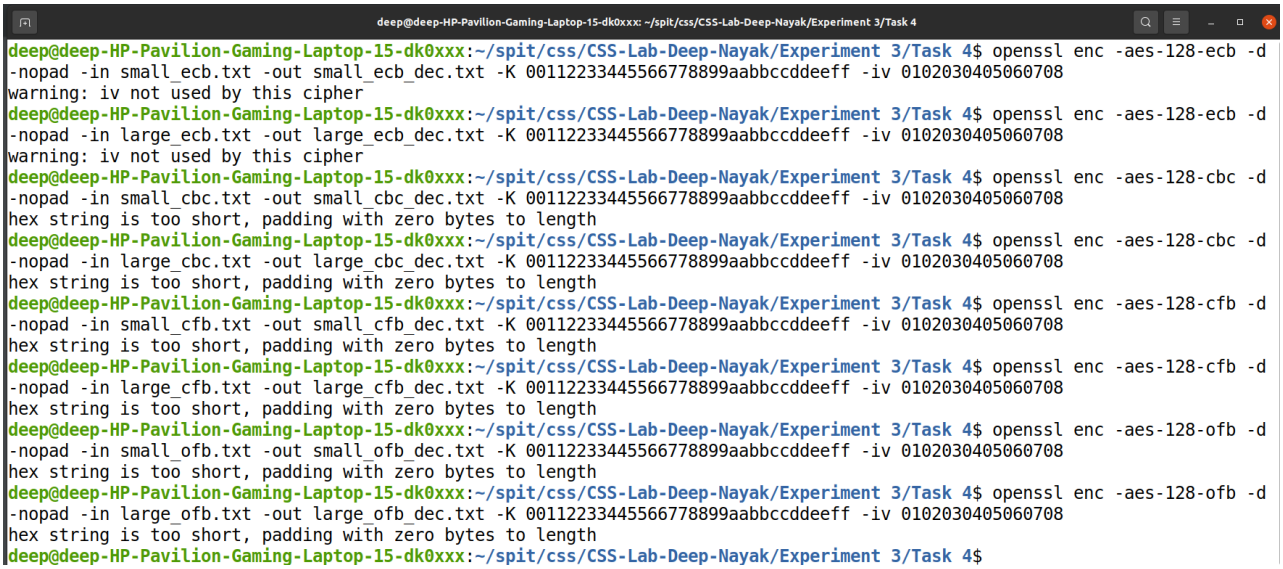


Fig 4.4

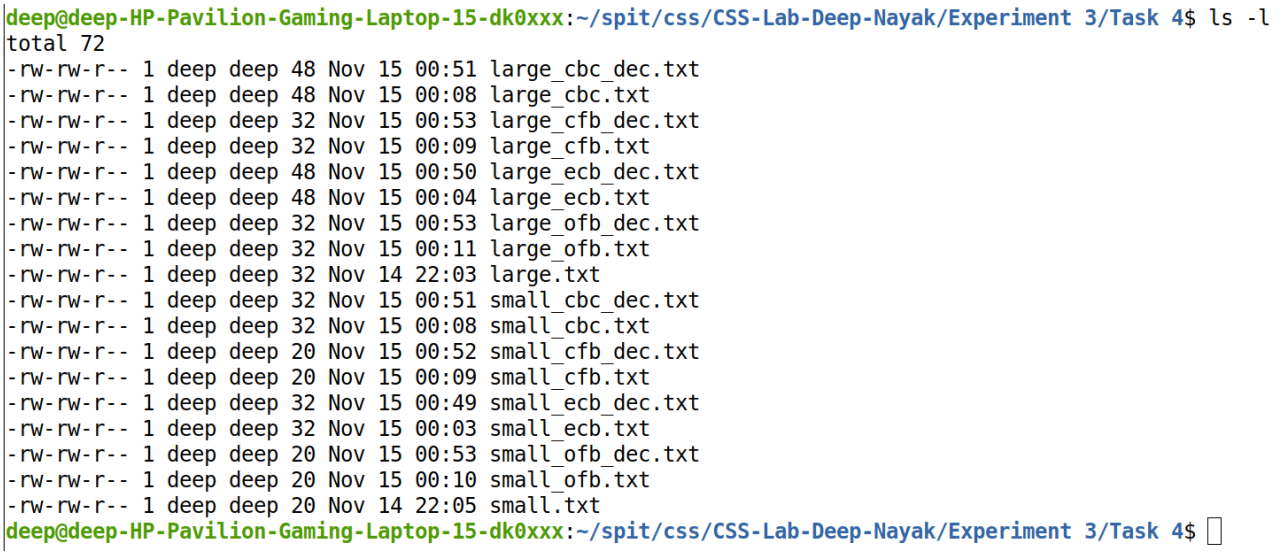


Fig 4.5

Observation:

1. The cipher modes which require encryption are CBC and ECB as we can see that for small as well as large text files, it is adding padding and hence increasing the size of the file to be an exact multiple of the block size, which is 16 Bytes for AES.

2. We can confirm that the padding standard is PKCS5 because, whenever a mode which requires padding is used, PKCS5 adds atleast 1 Byte of padding and at most n Bytes of padding where n is the block size. In short, it adds padding to the file so that its size becomes the ***next*** closest multiple of block size.

3. We can observe this in figure 4.5, as the size of large.txt is increased to 48 Bytes (next multiple of 16 after 32) and 32 Bytes (next multiple of 16 after 20) for small.txt

Encrypting using nopad option

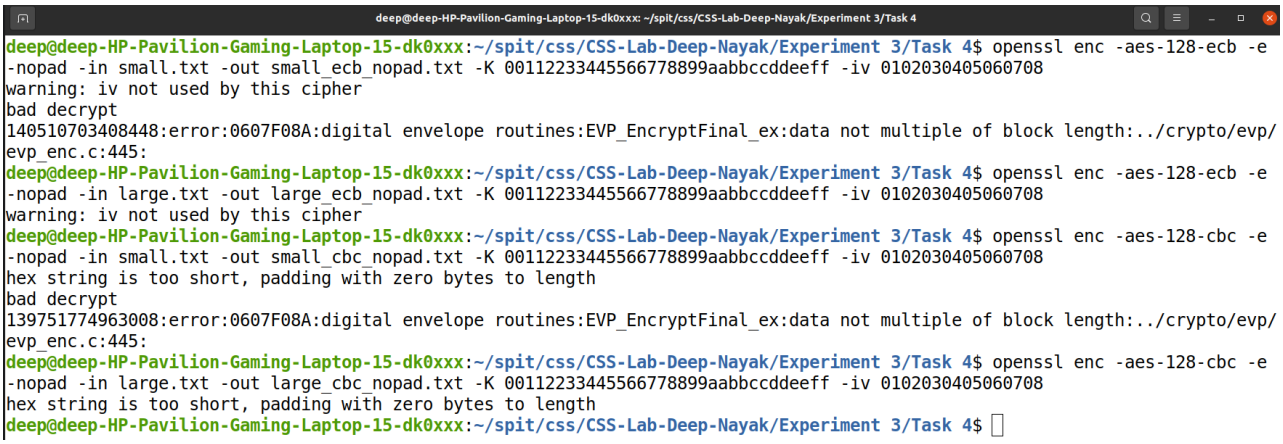


Fig 4.6

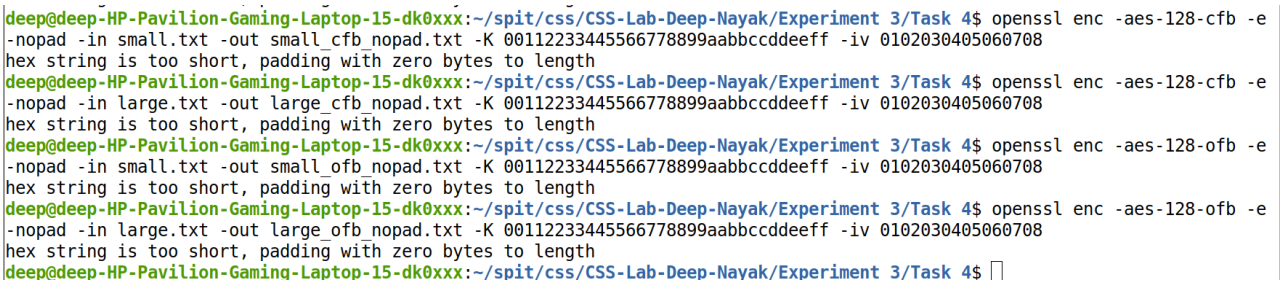


Fig 4.7

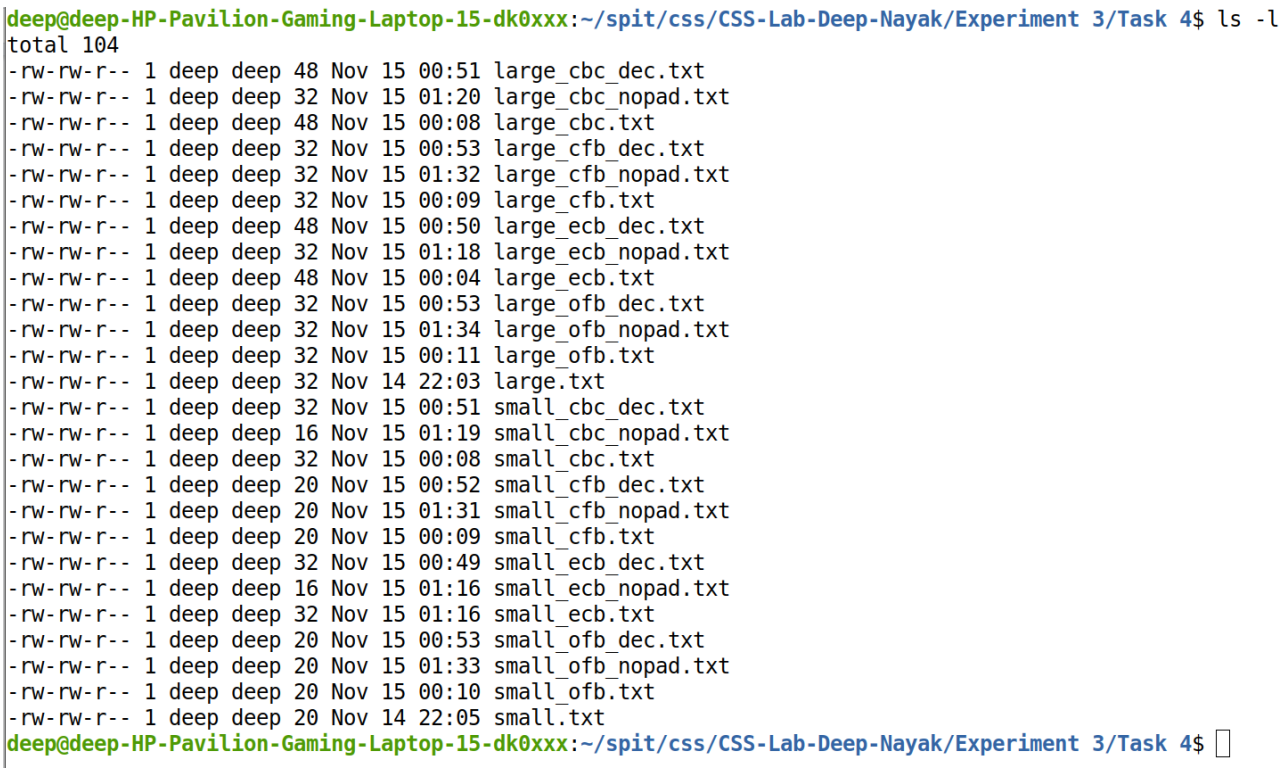


Fig 4.8

Decrypting the ciphertext that was encrypted using nopad

In every mode, we decrypt the encrypted small.txt and large.txt using the nopad option and not using the nopad option.

ECB

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-ecb -d
-in small_ecb_nopad.txt -out small_ecb_dec_nopad.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
bad decrypt
140083491927360:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:../crypto/evp/evp_enc.c:610:
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-ecb -d
-in large_ecb_nopad.txt -out large_ecb_dec_nopad.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
bad decrypt
139713682199872:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:../crypto/evp/evp_enc.c:610:
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-ecb -d
-nopad -in large_ecb_nopad.txt -out large_ecb_dec_nopad_1.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-ecb -d
-nopad -in small_ecb_nopad.txt -out small_ecb_dec_nopad.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$
```

Fig 4.9

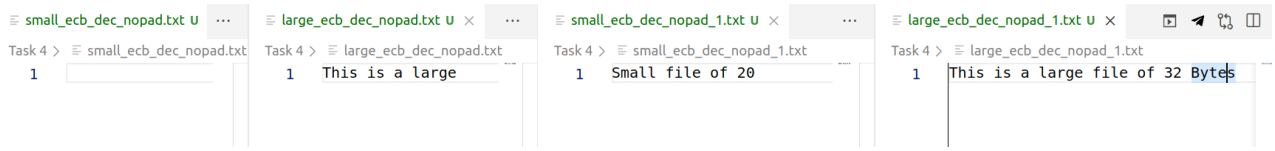


Fig 4.10

CBC

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-cbc -d
-nopad -in small_cbc_nopad.txt -out small_cbc_dec_nopad_1.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-cbc -d
-in small_cbc_nopad.txt -out small_cbc_dec_nopad.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
bad decrypt
140437280417088:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:../crypto/evp/evp_enc.c:610:
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-cbc -d
-nopad -in large_cbc_nopad.txt -out large_cbc_dec_nopad_1.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-cbc -d
-in large_cbc_nopad.txt -out large_cbc_dec_nopad.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
bad decrypt
140067518010688:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:../crypto/evp/evp_enc.c:610:
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$
```

Fig 4.11

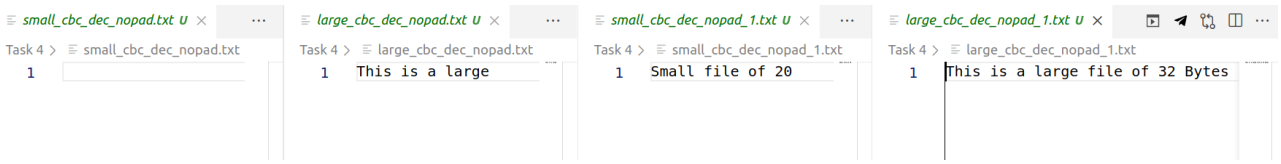


Fig 4.12

CFB

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-cfb -d
-nopad -in small_cfb_nopad.txt -out small_cfb_dec_nopad_1.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-cfb -d
-in small_cfb_nopad.txt -out small_cfb_dec_nopad.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-cfb -d
-nopad -in large_cfb_nopad.txt -out large_cfb_dec_nopad_1.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-cfb -d
-in large_cfb_nopad.txt -out large_cfb_dec_nopad.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$
```

Fig 4.13

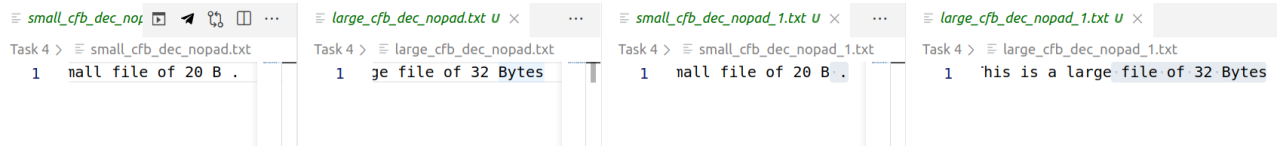


Fig 4.14

OFB

```
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-ofb -d
-nopad -in small_ofb_nopad.txt -out small_ofb_dec_nopad_1.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-ofb -d
-in small_ofb_nopad.txt -out small_ofb_dec_nopad.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-ofb -d
-nopad -in large_ofb_nopad.txt -out large_ofb_dec_nopad_1.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 4$ openssl enc -aes-128-ofb -d
-in large_ofb_nopad.txt -out large_ofb_dec_nopad.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

Fig 4.15

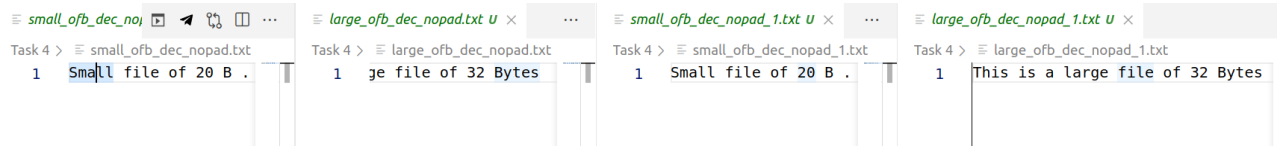


Fig 4.16

Observation:

1. Padding is not used by CFB and OFB. This is evident from the fact that the plain text is unaffected even if we use nopad option or not.

2. For ECB and CBC mode, while decrypting the cipher text which was encrypted using nopad option, the following is observed:

   a. If nopad is not used while decrypting, openssl assumes that padding was added while encrypting and hence tries to remove it. This results in loss of data in the decrypted text.
   b. If nopad is used, openssl does not try to remove the padding and since it was not added while encrypting, the original text is retrieved after decryption.

Task 5: Programming using the Crypto Library

So far, we have learned how to use the tools provided by openssl to encrypt and decrypt messages. In this task, we will learn how to use openssl's crypto library to encrypt/descrypt messages in programs.

OpenSSL provides an API called EVP, which is a high-level interface to cryptographic functions. Al though OpenSSL also has direct interfaces for each individual encryption algorithm, the EVP library pro vides a common interface for various encryption algorithms. To ask EVP to use a specific algorithm, we simply need to pass our choice to the EVP interface. A sample code is given in http://www.openssl. org/docs/crypto/EVP_EncryptInit.html. Please get yourself familiar with this program, and then do the following exercise.

You are given a plaintext and a ciphertext, and you know that aes-128-cbc is used to generate the ciphertext from the plaintext, and you also know that the numbers in the IV are all zeros (not the ASCII character '0'). Another clue that you have learned is that the key used to encrypt this plaintext is an English word shorter than 16 characters; the word that can be found from a typical English dictionary. Since the word has less than 16 characters (i.e. 128 bits), space characters (hexadecimal value 0x20) are appended to the end of the word to form a key of 128 bits. Your goal is to write a program to find out this key. You can download a English word list from the Internet. We have also linked one on the web page of this lab. The plaintext and ciphertext is in the following:

```
Plaintext (total 21 characters): This is a top secret. Ciphertext (in hex format):
8d20e5056a8d24d0462ce74e4904c1b5
                                    13e10d1df4a2ef2ad4540fae1ca0aaf9
```

Note 1: If you choose to store the plaintex message in a file, and feed the file to your program, you need to check whether the file length is 21. Some editors may add a special character to the end of the file. If that happens, you can use the ghex tool to remove the special character.

Note 2: In this task, you are supposed to write your own program to invoke the crypto library. No credit will be given if you simply use the openssl commands to do this task.

Note 3: To compile your code, you may need to include the header files in openssl, and link to openssl libraries. To do that, you need to tell your compiler where those files are. In your Makefile, you may want to specify the following:

Crypto Encryption/PV

```
INC=/usr/local/ssl/include/
LIB=/usr/local/ssl/lib/

all:
        gcc -I$(INC) -L$(LIB) -o enc yourcode.c -lcrypto -ldl
```

Code:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad

plain_text = b"This is a top secret."
cipher_hex = "8d20e5056a8d24d0462ce74e4904c1b513e10d1df4a2ef2ad4540fae1ca0aaf9"

result_k = []
file = open('words.txt', 'r')
lines = file.readlines()
words = [str.strip(line) for line in lines]
for word in words:
   if len(word) >= 16:
       continue
   word = word.lower()
   key = word.encode() + b' '*(16-len(word))
   cipher = AES.new(key, AES.MODE_CBC, iv=bytes.fromhex('0'*32))
   ciphertext = cipher.encrypt(pad(plain_text, AES.block_size))
   is_matched = "NOT MATCHED"
   if bytes.hex(ciphertext) == cipher_hex:
       is_matched = "MATCHED"
       result_k.append(word)
   print(word, bytes.hex(ciphertext), is_matched)
print("\n\nResulting Key:",result_k)
```

Fig 5.1

Output:

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE

```
zurn 02474b9a2fc07dc7e5d6eb6a1ec2a088b9914ef5e566cfd96d608cd031c837f8 NOT MATCHED
zurvan 6b3e75619622e2f9afc4068b49bcc8311ace4b7b9a839eb81b98eb6322a339e7 NOT MATCHED
zusman 8cbd1f67bf3a5c87321c2a4ba468e606d0d60c9d733dc89dcde56ae0a43c496b NOT MATCHED
zutugil f5b9cc23ad3645da2bda13e1090cf84d2b5e945f9635a522957e4a2316940c36 NOT MATCHED
zuurveldt 7c76dcc12c049ed0ea34aaffbd02f6963fa8117c1382c99784205aa1e71b0d22 NOT MATCHED
zuza 49259e25ddfdeac0b070282f10d98b25bbdfc97080b1d85a3c6bf1e40d440a08 NOT MATCHED
zuzana 4b317a3a18dd0c2ac63191e6ae80397a1507f6f861a0e9e342930278d26b24bc NOT MATCHED
zu-zu 04eb3d47bcc2fce2df97928758cd4182f2fdbc4866ffd4ed6227c80ad861f8ef NOT MATCHED
zwanziger 70b52532147330fb9b718d64b0cd144300ad9e4fe5ab875cef8ac74d587621ff NOT MATCHED
zwart 7b8693c89e594c35fec42bd33ec5f6a318e4c6096c8f73b0ba65346d7ee79627 NOT MATCHED
zwei 965faf9245114fbd64c0ac1bc2f5213f69e12199a7ad6dea685b56d62ac5acb7 NOT MATCHED
zweig 43f25db77ff3a0363dff7617755855365cf2ff071b50729b97cc6ebf3e5618cc NOT MATCHED
zwick 96ea898d70c7212bb5d0ff7735c55bcc318cfd3efd71a5bde00f25c55e6501b6 NOT MATCHED
zwickau 7c1b8bfb63bfd5b6675224556182a5099835623f2f8debe2ec604f9d82cbd8e4 NOT MATCHED
zwicky c6c25a3e4cafbdedb440529f89f3f18ac2521d0b4e44afee0406173430b61f57 NOT MATCHED
zwieback 31b67ea013c06813e26e0466ef22aac0d0742615c3a8127e438e87b4b20c5464 NOT MATCHED
zwiebacks cf4df730792dbc47b427c4aee046a20f6f3362cc657e8ceb9515c1f61c34a8a7 NOT MATCHED
zwiebel 080d15f65c1fb580a38e0f99e565869cc89fbb3773d4ad0f2e05422ab31c41ee NOT MATCHED
zwieselite 2f2342c4e1c99673b54aca30655a35d89767864c3f5e9718f90365d92360dfa0 NOT MATCHED
zwingle c99210ecf7014a9a948d5270ae6d3cd99fdafbc8cf5a68ca2b81877bea74f39b NOT MATCHED
zwingli b05d6713ea431a27239a46d191cadc9d5521c2ae25ec2596a8bc1dbf2bfab102 NOT MATCHED
zwinglian 5f2b18d9f93d7fa27fcca4c27581a2c272e864cf25a1e71652b86979ee6b5bac NOT MATCHED
zwinglianism aedf7feabae8a15a22334d5cd43844b60f946f0382d13a8acd60b84dd80348b0 NOT MATCHED
zwinglianist 17158ad1efe7d3e6fcd751fc2d801019d22b3f64a37e2ddf68ae9d877cf8a084 NOT MATCHED
zwitter 6df12d857a5b3461a67a6f4548250d705407c73c48cd5242f226ef38077d18b9 NOT MATCHED
zwitterion 0b794ab2da476a76aeab642ad9fe582dfd79cc0042113790d30adef2da8b51b5 NOT MATCHED
zwitterionic 43d9a3b874200bc70e02e6b0143c3e9495c296487814a3c4d4c8747d630c7de3 NOT MATCHED
zwolle 327589a13f6940f1b17e3f3e85826724e8b2bb8cebcecf4c151e7f91e625c635 NOT MATCHED
zworykin 770797ec4f4bd24b34e54ad3f7e958c05cc673631b47623b372e4473519e1305 NOT MATCHED
zz 5d7a15c6b07909667c79feac6346d16c02204f1c63389721179c2015cb21c2bb NOT MATCHED
zzt 58dae019524c7d7d79ceaa3a43186ce4a5f64d9ad1817ac4b06a649500de2ef8 NOT MATCHED
zzz 2bfc0c1cb86d489616d5eaf811512b493869e2b14dfe420eaba6a4a4f431e94b NOT MATCHED


Resulting Key: ['median']
deep@deep-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/spit/css/CSS-Lab-Deep-Nayak/Experiment 3/Task 5$
```

Fig 5.2

Conclusion:

1. AES, DES are symmetric key algorithms using the same keys to encrypt and decrypt the data. ARIA and Camellia are modern encryption algorithms at par with AES and contain more rounds of computation, substitution and permutation before giving the final result.
2. ECB mode of encryption is the weakest form of encryption in comparison to CBC, CFB and OFB. If it is used to encrypt an image, an outline of the main contents of the image is still visible in the encrypted image.
3. 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 could also verify that openssl uses PKCS5 standard for padding as it adds padding till the file size becomes the next closest multiple of the block size.
4. 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.
5. I could conclude from this experiment that, If we know the plaintext, ciphertext and iv, I can easily find the key using brute force method. In this case, we tried out each word with less than 16 characters from the dictionary and found the key to be "median" as seen in figure 5.2.

Github Link: https://github.com/deepnayak/CSS-Lab-Deep-Nayak