Dilnoza Saidova, Nathan Mahnke
TCSS 481: Computer Security
October 27, 2022

Lab 1: Secret – Key Encryption Lab

Task 1. Frequency Analysis

Step 1: We used Python console to generate a permutation of a-z:

```
>>> import random
>>> s = "abcdefghijklmnopqrstuvwxyz"
>>> ''.join(random.sample(s,len(s)))
'gndskrxlhzqeyivmaojtpufwbc'
```

Step 2: We converted all upper cases to lower cases, removing punctuations and numbers:

```
[10/27/22]seed@VM:~/.../Files$ tr [:upper:] [:lower:] < words.txt > lowercase.tx
t
[10/27/22]seed@VM:~/.../Files$ tr -cd '[a-z][\n][:space:]' < lowercase.txt > wor
ds.txt
[10/27/22]seed@VM:~/.../Files$
```

Step 3: We used `tr` command to do encryption, encrypting letters and leaving spaces:

```
[10/27/22]seed@VM:~/.../Files$ tr 'abcdefghijklmnopqrstuvwxyz' 'sxtrwinqbedpvgkf
malhyojzc' \ < words.txt > ciphertext.txt
```

Using Online frequency analysis tool (https://www.dcode.fr/frequency-analysis), we were able to find frequencies of repeated characters in our ciphertext:

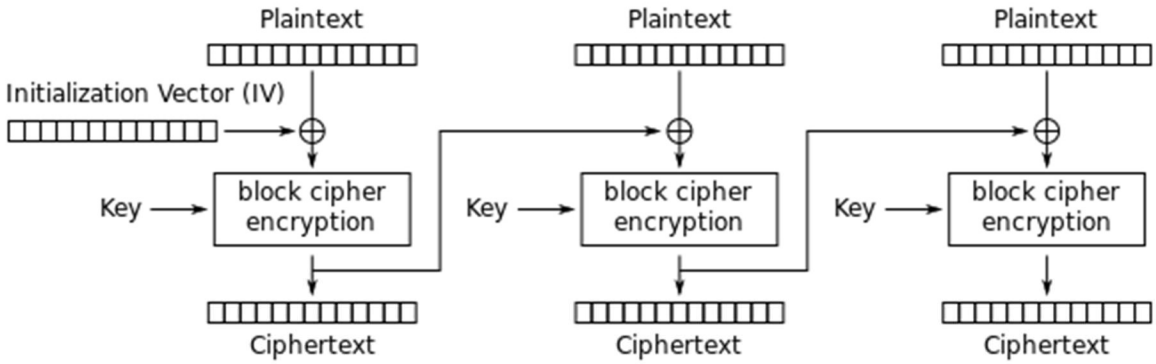
N	488×	12.41%
Y	373×	9.49%
V	348×	8.85%
X	291×	7.4%
U	280×	7.12%
Q	276×	7.02%
M	264×	6.72%
H	235×	5.98%
T	183×	4.66%
I	166×	4.22%
P	156×	3.97%
A	116×	2.95%
C	104×	2.65%
Z	95×	2.42%
L	90×	2.29%
G	83×	2.11%
B	83×	2.11%
R	82×	2.09%
E	76×	1.93%
D	59×	1.5%
F	49×	1.25%
S	19×	0.48%
K	5×	0.13%
J	5×	0.13%
O	4×	0.1%
W	1×	0.03%

By single letter frequency, the letters in ciphertext sorted by frequency are: `nyvxuqmhtipaczlgbredfskjow`. Compared to modern English, letter frequency found in Wikipedia is `eothasinrdluymwfgcbpkvjqxz.`

Task 2. Encryption using Different Ciphers and Modes

We replaced the `ciphertype` with -aes-128-cbc, -aes-128-cfb, -aes-128-ofb.

Cipher Block Chaining (CBC) – each block of plaintext gets XOR-ed with the preceding cipher block.



Cipher Block Chaining (CBC) mode encryption

Encryption:

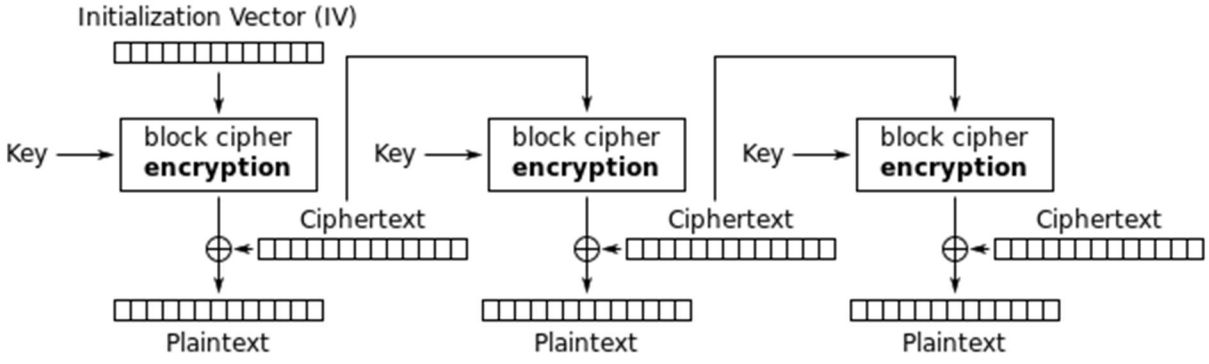
```
[10/27/22]seed@VM:~/Documents$ openssl enc -aes-128-cbc -e -in plaintext.txt -ou
t cbc_cipher.bin \
> -K 00112233445566778899aabbccddeeff \
> -iv 0102030405060708
```

Decryption:

```
[10/27/22]seed@VM:~/Documents$ openssl enc -aes-128-cbc -d -in cbc_cipher.bin -o
ut cbc_plain.txt \
> -K 00112233445566778899aabbccddff \
> -iv 0102030405060708
```

Validation: `$diff plaintext.txt cbc_plain.txt`

Cipher Feedback (CFB) – encryption of the block cipher that takes in the ciphertext of preceding block is XOR-ed with the plaintext, generating the final ciphertext.



Cipher Feedback (CFB) mode decryption

Encryption:

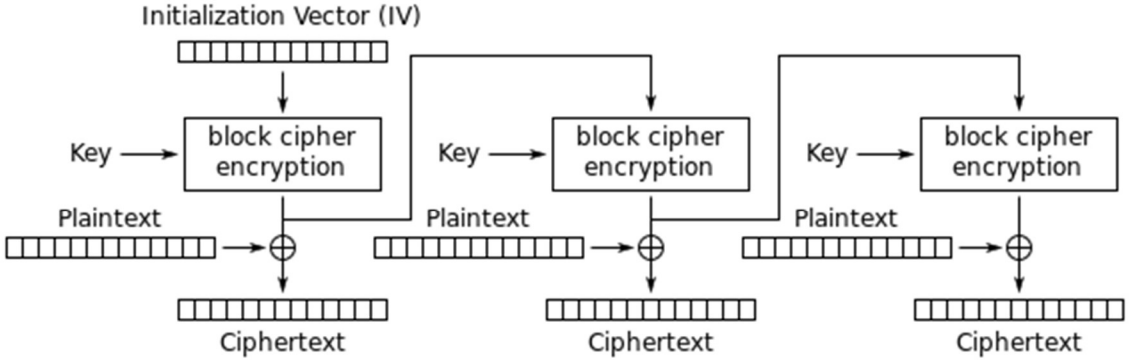
```
[10/27/22]seed@VM:~/Documents$ openssl enc -aes-128-cfb -e -in plaintext.txt -ou
t cfb_cipher.bin \
> -K 00112233445566778899aabbccddeeff \
> -iv 0102030405060708
```

Decryption:

```
[10/27/22]seed@VM:~/Documents$ openssl enc -aes-128-cfb -d -in cfb_cipher.bin -o
ut cfb_plain.txt \
> K 00112233445566778899aabbccddeeff \
> iv 0102030405060708
```

Validation: `$diff plaintext.txt cbc_plain.txt`

Output Feedback (OFB) – the block cipher that takes in the ciphertext of preceding block is XOR-ed with the plaintext, generating the final ciphertext after being encrypted.



Output Feedback (OFB) mode encryption

Encryption:
```
[10/27/22]seed@VM:~/Documents$ openssl enc -aes-128-ofb -e -in plaintext.txt -ou
t ofb_cipher.bin \
> -K 00112233445566778899aabbccddeeff \
> -iv 0102030405060708
```

Decryption:
```
[10/27/22]seed@VM:~/Documents$ openssl enc -aes-128-ofb -d -in ofb_cipher.bin -o
ut ofb_plain.txt \
> -K 00112233445566778899aabbccddeeff \
> -iv 0102030405060708
```

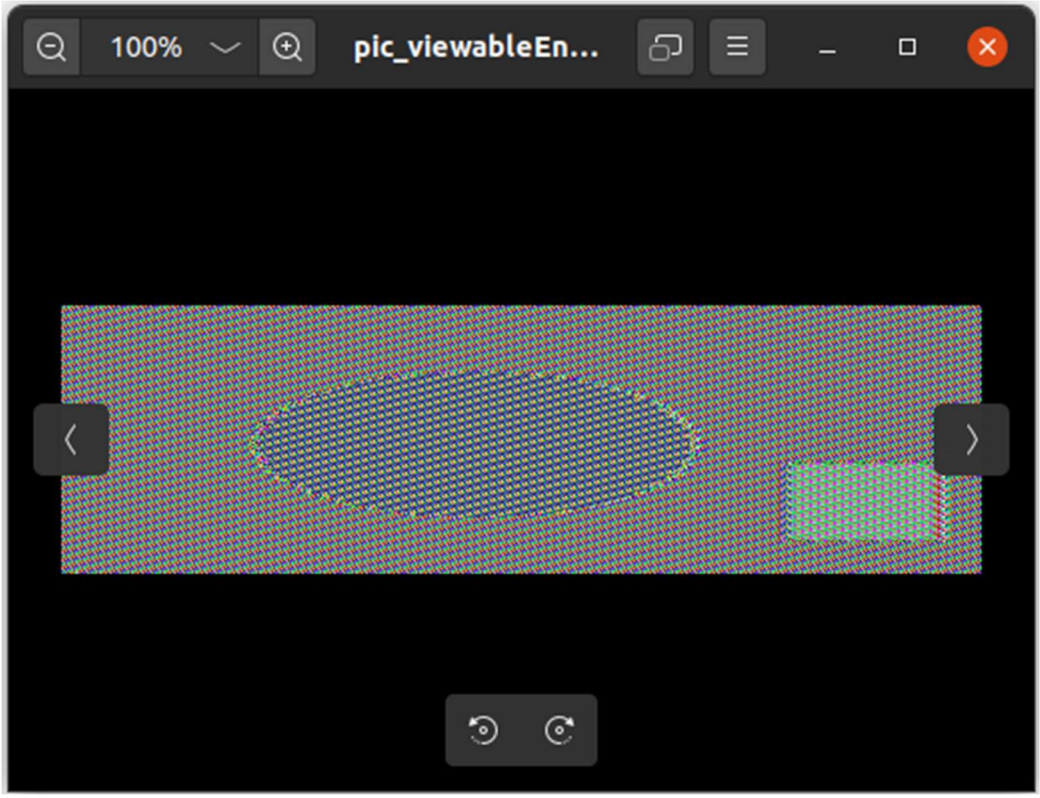
Validation: `$diff plaintext.txt cbc_plain.txt`

Task 3. Encryption Mode – ECB vs. CBC

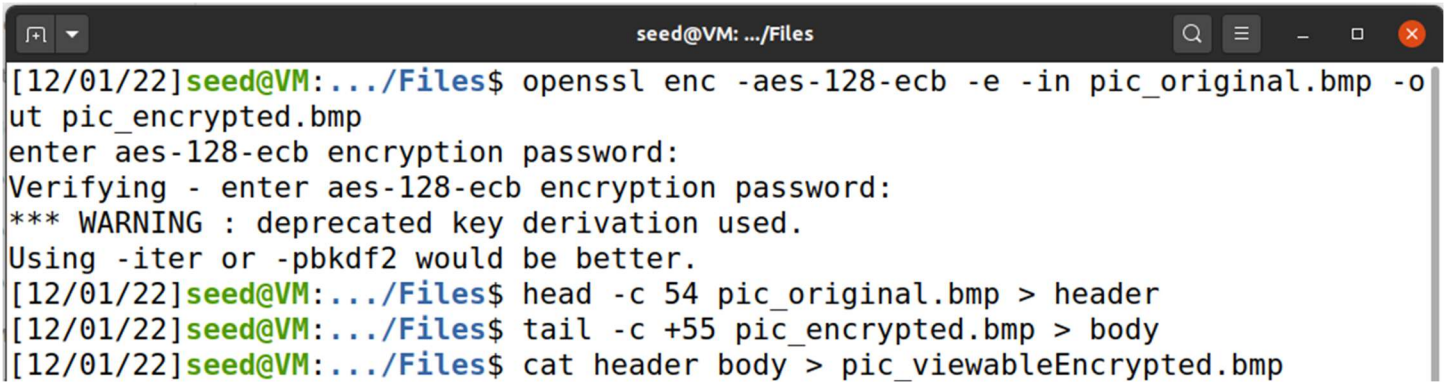
We noticed that with the ECB encryption, the image was still recognizable. We could gather what the initial shapes in the image were and get a general idea of their colors. With the CBC encryption, there was nothing but noise. The picture was unrecognizable, and we could gather no information from it regarding the original image. With our personal image (Great Wave Off Kanagawa), both encryption methods produced images consisting of pure noise and were unrecognizable.

```
seed@VM: .../Files
[12/01/22]seed@VM:.../Files$ openssl enc -aes-128-ecb -e -in GreatWaveOffKanagawa.bmp -out GreatWaveOffKanagawa_encrypted.bmp
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[12/01/22]seed@VM:.../Files$ head -c 54 GreatWaveOffKanagawa.bmp > header
[12/01/22]seed@VM:.../Files$ tail -c +55 GreatWaveOffKanagawa_encrypted.bmp > body
[12/01/22]seed@VM:.../Files$ cat header body > GreatWaveOffKanagawa viewableEncrypted.bmp
```

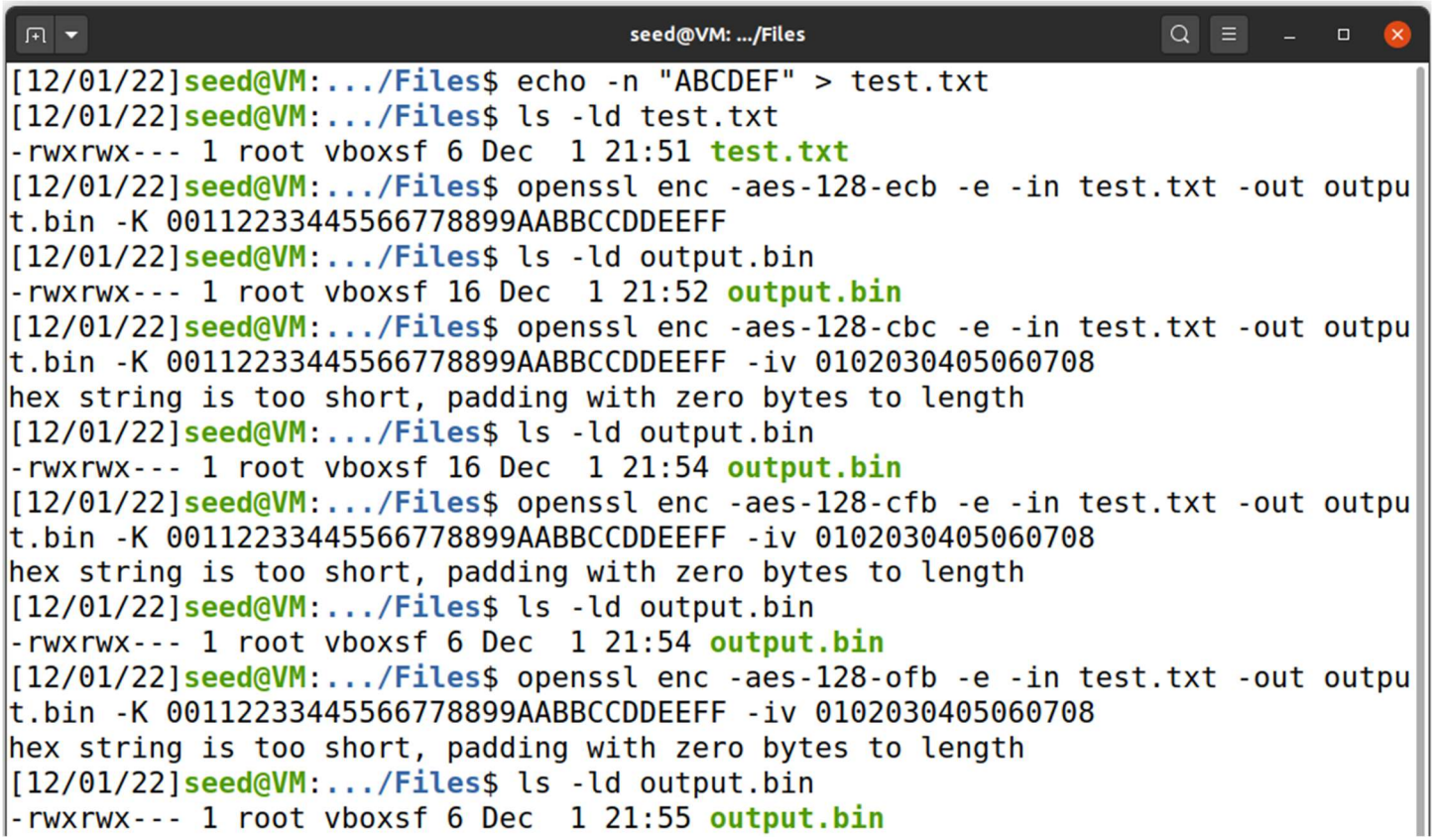
ECB Encryption:



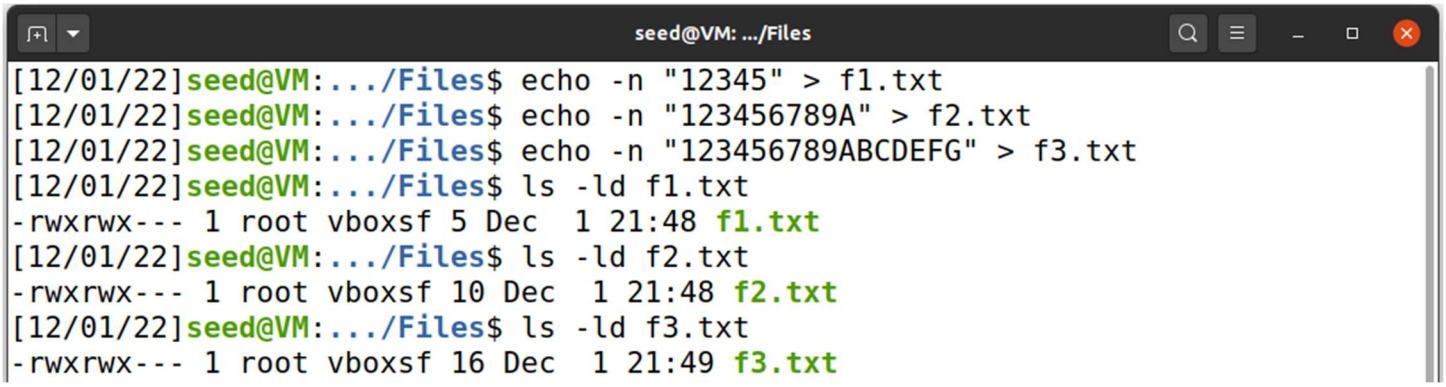
CBC Encryption:

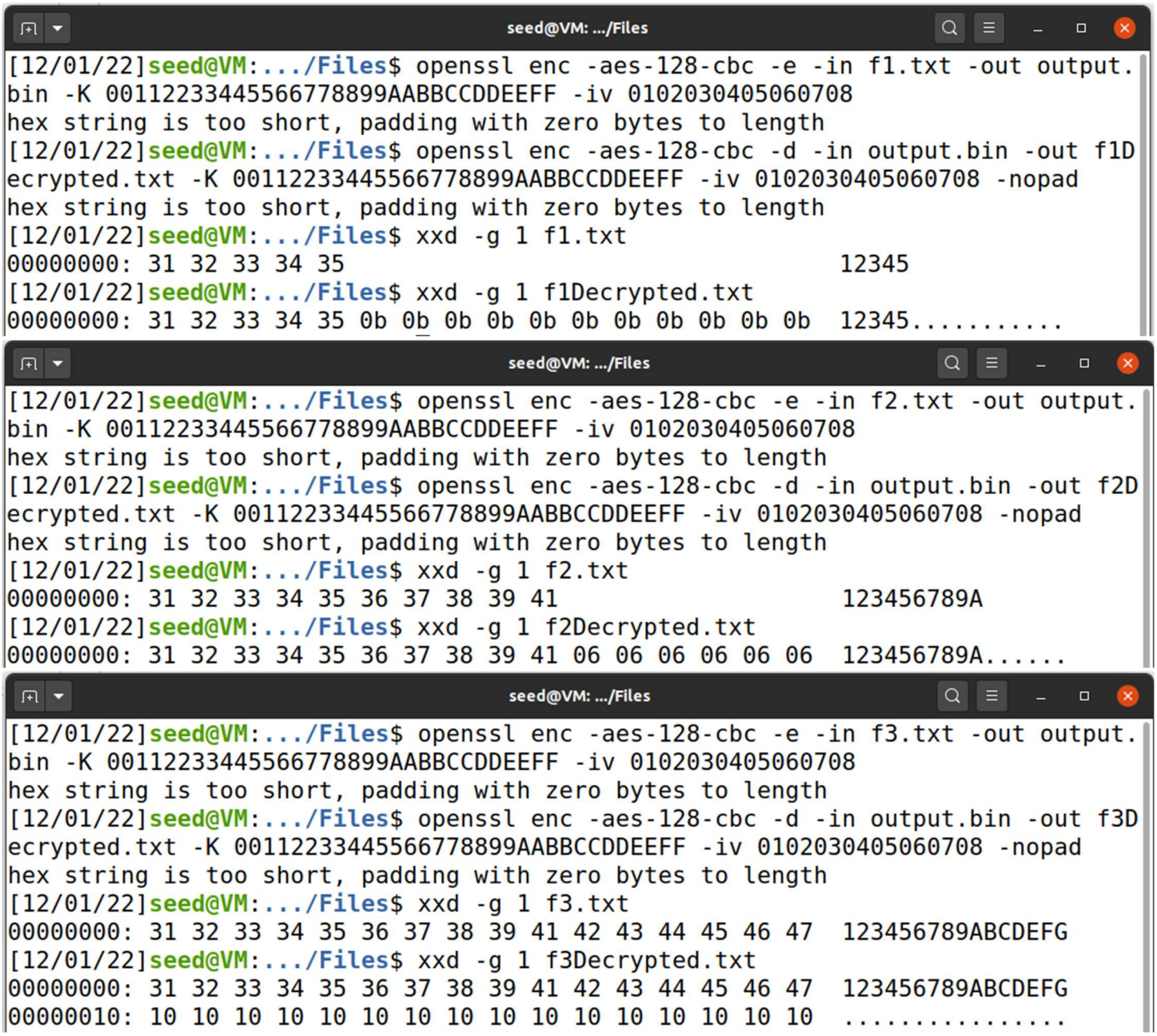


Task 4. Padding



CBC and ECB contain padding (16 bytes). This is because they are block ciphers; so, they use padding. CFB and OFB lacked padding. This is because they are stream ciphers; so, they don't use padding. Following are the screenshots we used for decrypting the padding:





Task 5. Error Propagation – Corrupted Cipher Text

Predictions:

- ECB: We would expect only the block containing the effected byte to be unrecoverable due to ECB encrypting block-by-block.
- CBC: Because CBC is probabilistic, we would expect all blocks after the effected block to be unrecoverable.
- CFB: Because every bit in CFB effects all others, we would expect the corrupted bit to propagate throughout the data during decryption, rendering it all corrupted.
- OFB: We would not expect to see the corruption propagate as bad as CFB, but it should be similar due to them both using stream encryption.

We made a file that was at least 1000 bytes:

```
seed@VM: .../Files
[12/01/22]seed@VM:.../Files$ python3 -c "print ('123456789a'*100)" > thousandByt
es.txt
[12/01/22]seed@VM:.../Files$ ls -ld thousandBytes.txt
-rwxrwx--- 1 root vboxsf 1001 Dec  1 22:25 thousandBytes.txt
[12/01/22]seed@VM:.../Files$ openssl enc -aes-128-ecb -e -in thousandBytes.txt -
out output.bin -K 00112233445566778899aabbccddeeff
```

We then used ECB encryption to encrypt the file and used bless to corrupt the 55th byte:

```
seed@VM: .../Files
[12/01/22]seed@VM:.../Files$ openssl enc -aes-128-ecb -e -in thousandBytes.txt -
out output.bin -K 00112233445566778899aabbccddeeff
[12/01/22]seed@VM:.../Files$ bless output.bin
```

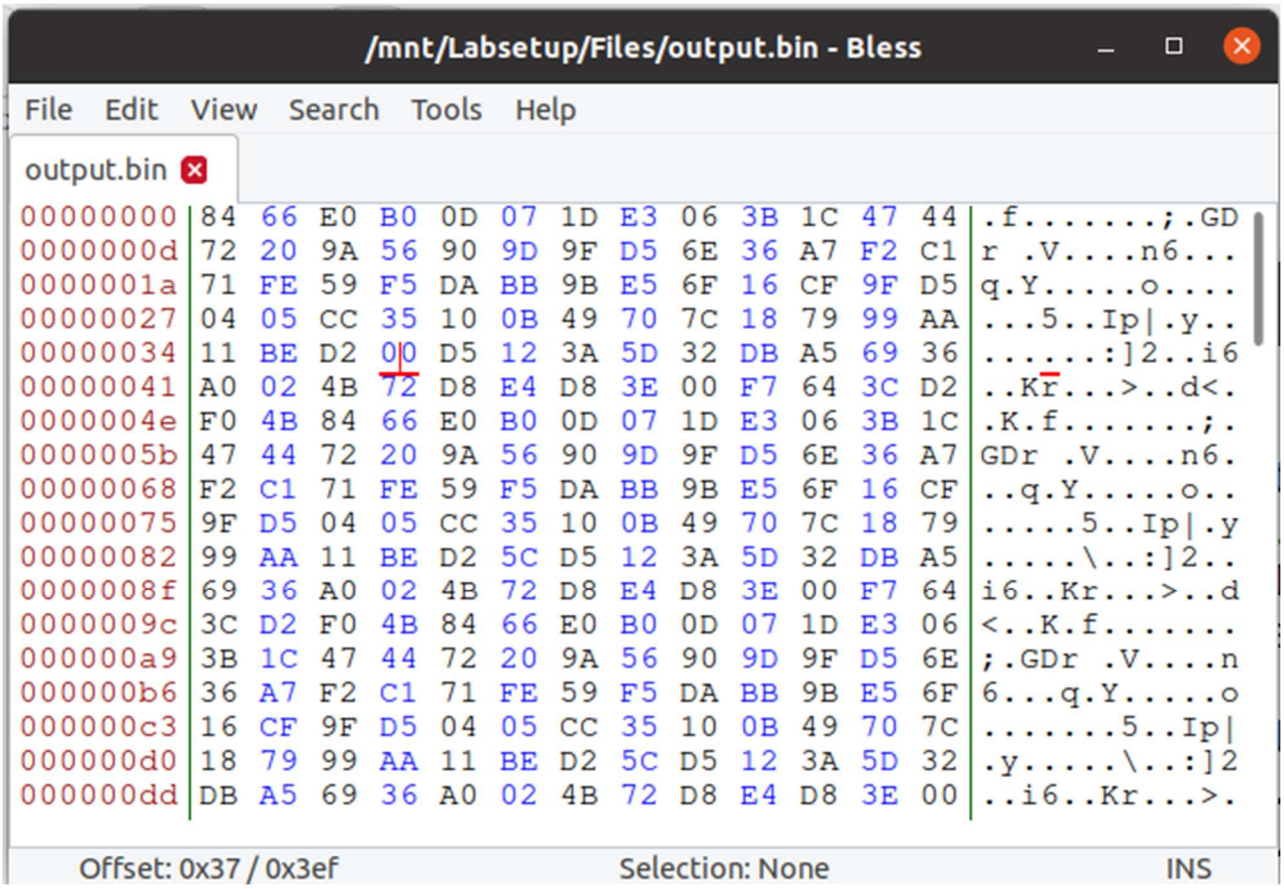
/mnt/Labsetup/Files/output.bin - Bless

File Edit View Search Tools Help

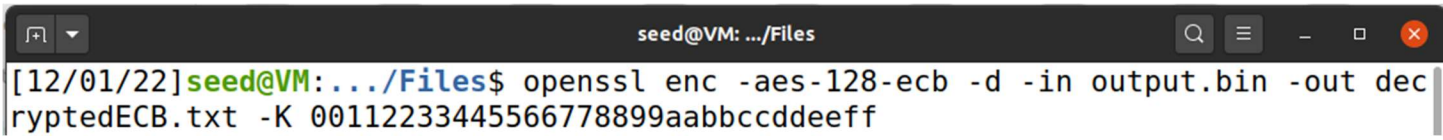
output.bin

```
00000000 84 66 E0 B0 0D 07 1D E3 06 3B 1C 47 44 .f.......;.GD
0000000d 72 20 9A 56 90 9D 9F D5 6E 36 A7 F2 C1 r .V....n6...
0000001a 71 FE 59 F5 DA BB 9B E5 6F 16 CF 9F D5 q.Y.....o....
00000027 04 05 CC 35 10 0B 49 70 7C 18 79 99 AA ...5..Ip|.y..
00000034 11 BE D2 5C D5 12 3A 5D 32 DB A5 69 36 ...\..:]2..i6
00000041 A0 02 4B 72 D8 E4 D8 3E 00 F7 64 3C D2 ..Kr...>..d<.
0000004e F0 4B 84 66 E0 B0 0D 07 1D E3 06 3B 1C .K.f.......;.
0000005b 47 44 72 20 9A 56 90 9D 9F D5 6E 36 A7 GDr .V....n6.
00000068 F2 C1 71 FE 59 F5 DA BB 9B E5 6F 16 CF ..q.Y.....o..
00000075 9F D5 04 05 CC 35 10 0B 49 70 7C 18 79 .....5..Ip|.y
00000082 99 AA 11 BE D2 5C D5 12 3A 5D 32 DB A5 .....\..:]2..
0000008f 69 36 A0 02 4B 72 D8 E4 D8 3E 00 F7 64 i6..Kr...>..d
0000009c 3C D2 F0 4B 84 66 E0 B0 0D 07 1D E3 06 <..K.f.......
000000a9 3B 1C 47 44 72 20 9A 56 90 9D 9F D5 6E ;.GDr .V....n
000000b6 36 A7 F2 C1 71 FE 59 F5 DA BB 9B E5 6F 6...q.Y.....o
000000c3 16 CF 9F D5 04 05 CC 35 10 0B 49 70 7C .......5..Ip|
000000d0 18 79 99 AA 11 BE D2 5C D5 12 3A 5D 32 .y.....\..:]2
000000dd DB A5 69 36 A0 02 4B 72 D8 E4 D8 3E 00 ..i6..Kr...>.
```

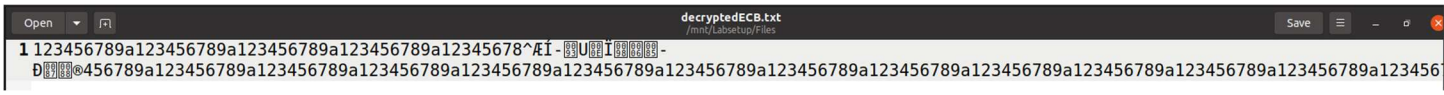
Offset: 067 / 01757 Selection: None INS



We decrypted the text:

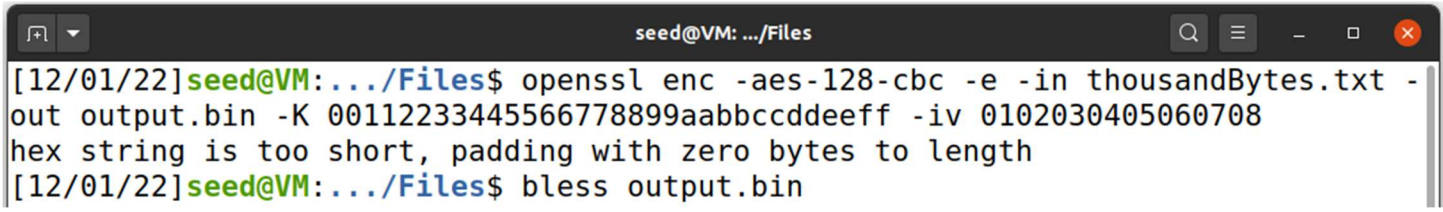


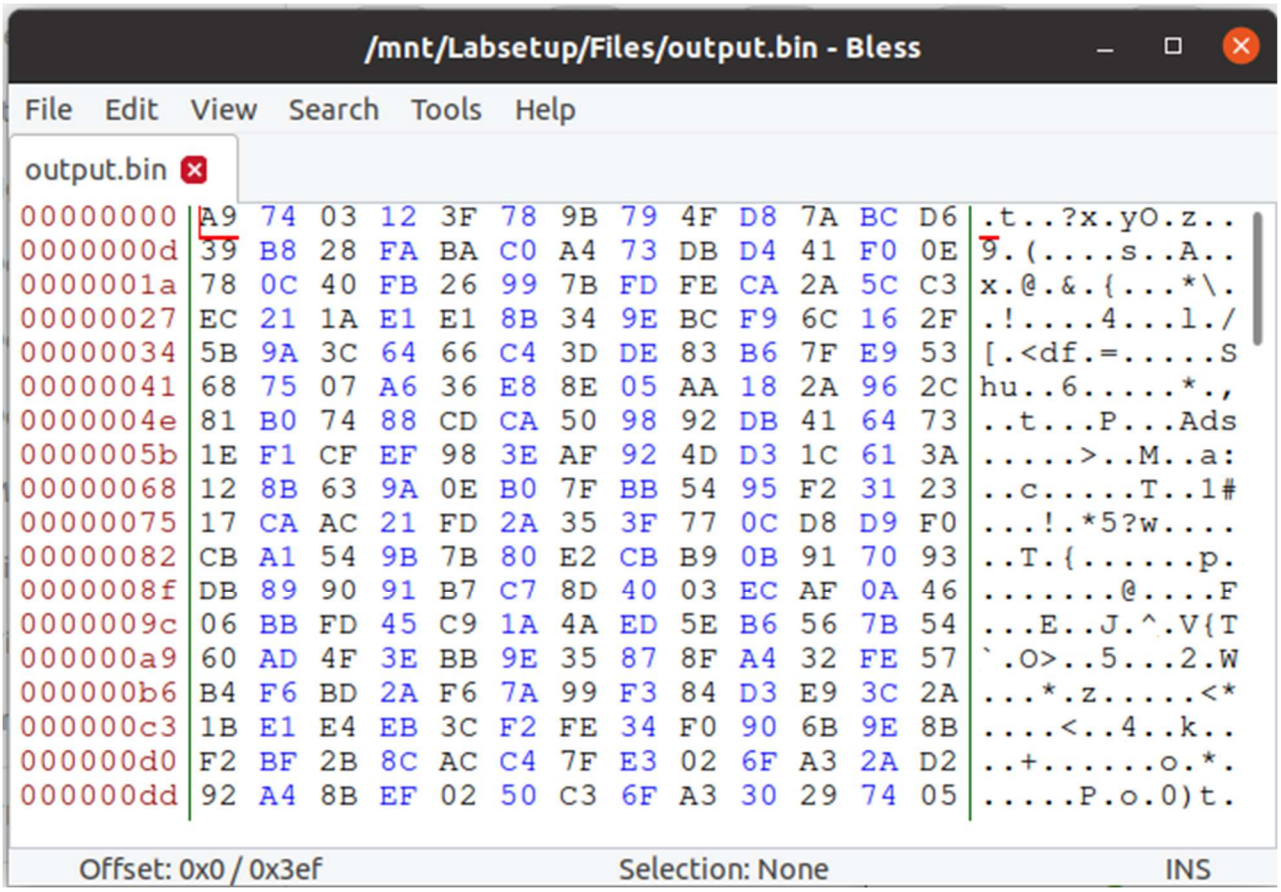
The we viewed the text:

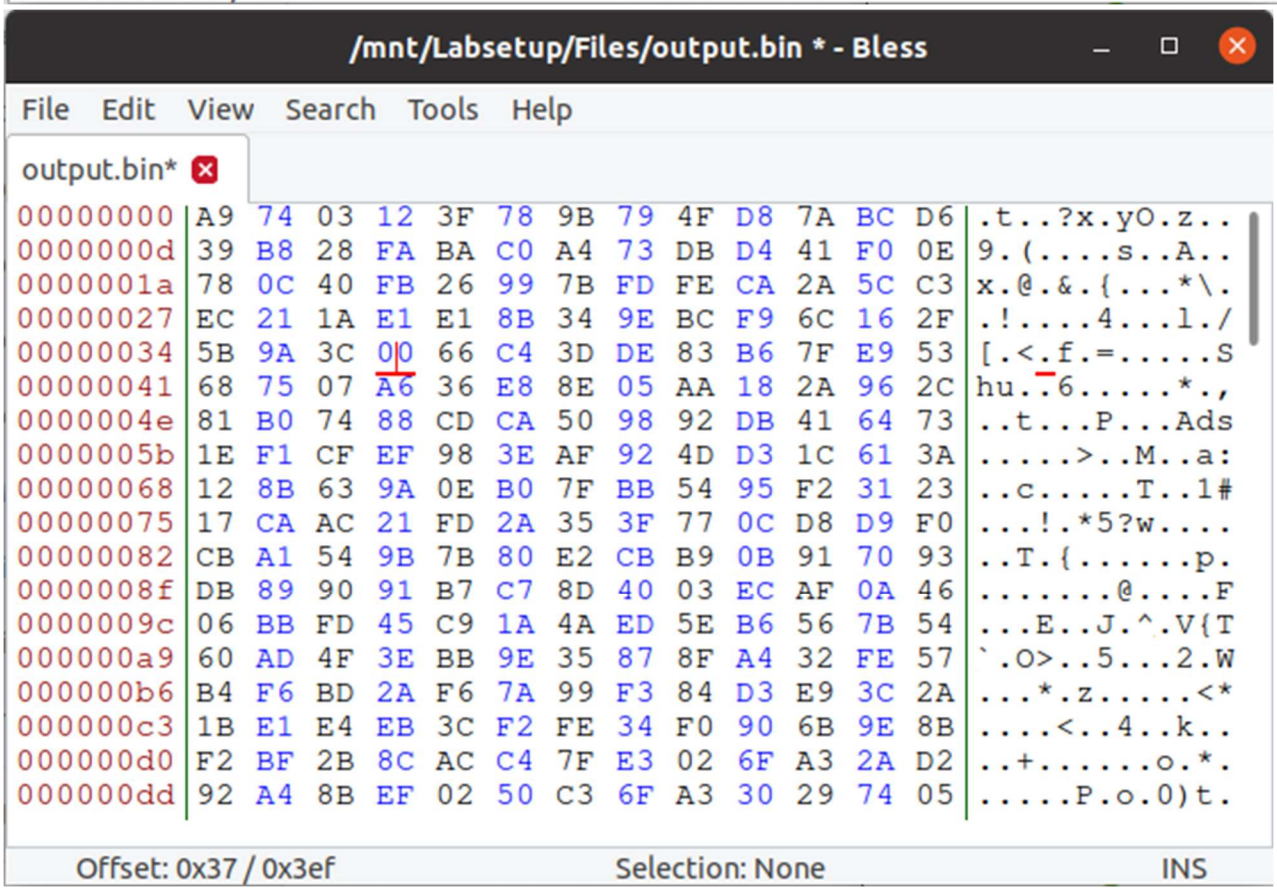


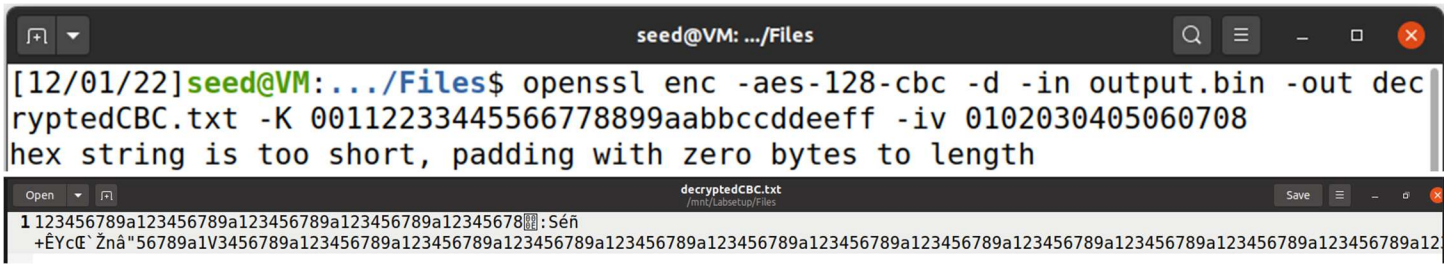
We did this in all four decryption styles.

CBC:

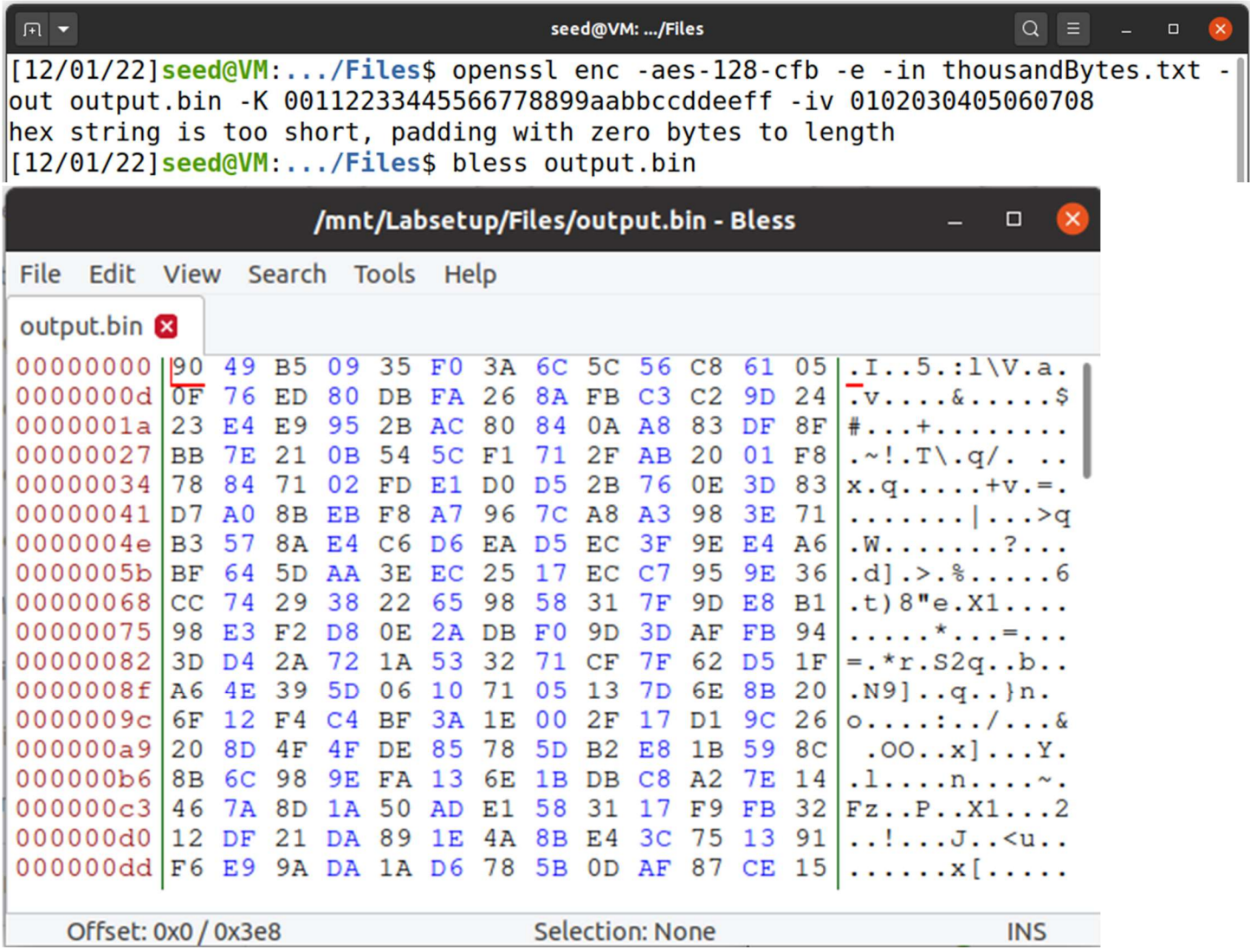


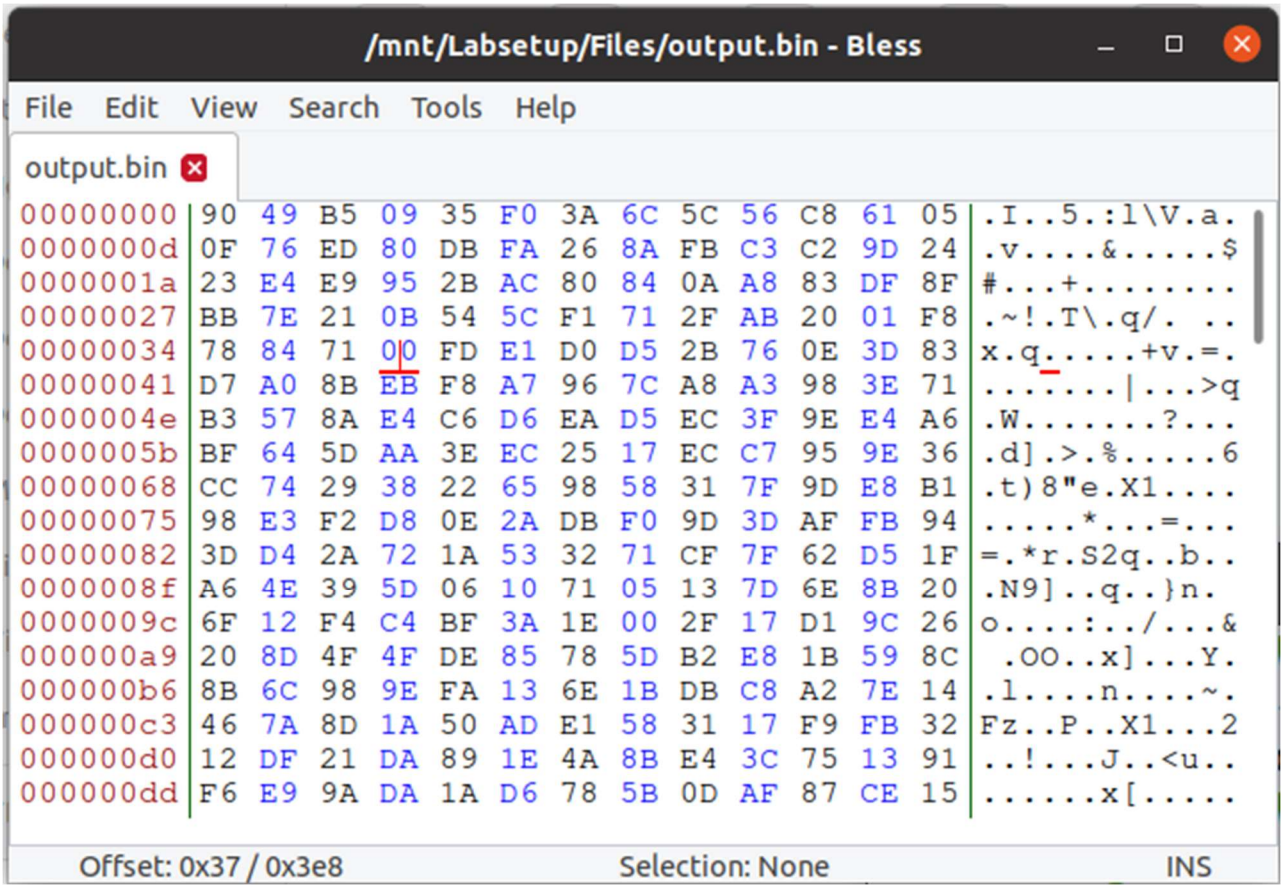

/mnt/Labsetup/Files/output.bin - Bless
File Edit View Search Tools Help
output.bin
00000000 A9 74 03 12 3F 78 9B 79 4F D8 7A BC D6 .t..?x.yO.z..
0000000d 39 B8 28 FA BA C0 A4 73 DB D4 41 F0 0E 9.(....s..A..
0000001a 78 0C 40 FB 26 99 7B FD FE CA 2A 5C C3 x.@.&.{...*\.
00000027 EC 21 1A E1 E1 8B 34 9E BC F9 6C 16 2F .!....4...l./
00000034 5B 9A 3C 64 66 C4 3D DE 83 B6 7F E9 53 [.<df.=.....S
00000041 68 75 07 A6 36 E8 8E 05 AA 18 2A 96 2C hu..6.....*.,
0000004e 81 B0 74 88 CD CA 50 98 92 DB 41 64 73 ..t...P...Ads
0000005b 1E F1 CF EF 98 3E AF 92 4D D3 1C 61 3A .....>..M..a:
00000068 12 8B 63 9A 0E B0 7F BB 54 95 F2 31 23 ..c.....T..1#
00000075 17 CA AC 21 FD 2A 35 3F 77 0C D8 D9 F0 ...!.*5?w....
00000082 CB A1 54 9B 7B 80 E2 CB B9 0B 91 70 93 ..T.{......p.
0000008f DB 89 90 91 B7 C7 8D 40 03 EC AF 0A 46 ....@....F
0000009c 06 BB FD 45 C9 1A 4A ED 5E B6 56 7B 54 ...E..J.^.V{T
000000a9 60 AD 4F 3E BB 9E 35 87 8F A4 32 FE 57 `.O>..5...2.W
000000b6 B4 F6 BD 2A F6 7A 99 F3 84 D3 E9 3C 2A ...*.z.....<*
000000c3 1B E1 E4 EB 3C F2 FE 34 F0 90 6B 9E 8B ....<..4..k..
000000d0 F2 BF 2B 8C AC C4 7F E3 02 6F A3 2A D2 ..+......o.*.
000000dd 92 A4 8B EF 02 50 C3 6F A3 30 29 74 05 .....P.o.0)t.
Offset: 0x0 / 0x3ef
Selection: None
INS


/mnt/Labsetup/Files/output.bin * - Bless
File Edit View Search Tools Help
output.bin*
00000000 A9 74 03 12 3F 78 9B 79 4F D8 7A BC D6 .t..?x.yO.z..
0000000d 39 B8 28 FA BA C0 A4 73 DB D4 41 F0 0E 9.(....s..A..
0000001a 78 0C 40 FB 26 99 7B FD FE CA 2A 5C C3 x.@.&.{...*\.
00000027 EC 21 1A E1 E1 8B 34 9E BC F9 6C 16 2F .!....4...l./
00000034 5B 9A 3C 00 66 C4 3D DE 83 B6 7F E9 53 [.<.f.=.....S
00000041 68 75 07 A6 36 E8 8E 05 AA 18 2A 96 2C hu..6.....*.,
0000004e 81 B0 74 88 CD CA 50 98 92 DB 41 64 73 ..t...P...Ads
0000005b 1E F1 CF EF 98 3E AF 92 4D D3 1C 61 3A .....>..M..a:
00000068 12 8B 63 9A 0E B0 7F BB 54 95 F2 31 23 ..c.....T..1#
00000075 17 CA AC 21 FD 2A 35 3F 77 0C D8 D9 F0 ...!.*5?w....
00000082 CB A1 54 9B 7B 80 E2 CB B9 0B 91 70 93 ..T.{......p.
0000008f DB 89 90 91 B7 C7 8D 40 03 EC AF 0A 46 ....@....F
0000009c 06 BB FD 45 C9 1A 4A ED 5E B6 56 7B 54 ...E..J.^.V{T
000000a9 60 AD 4F 3E BB 9E 35 87 8F A4 32 FE 57 `.O>..5...2.W
000000b6 B4 F6 BD 2A F6 7A 99 F3 84 D3 E9 3C 2A ...*.z.....<*
000000c3 1B E1 E4 EB 3C F2 FE 34 F0 90 6B 9E 8B ....<..4..k..
000000d0 F2 BF 2B 8C AC C4 7F E3 02 6F A3 2A D2 ..+......o.*.
000000dd 92 A4 8B EF 02 50 C3 6F A3 30 29 74 05 .....P.o.0)t.
Offset: 0x37 / 0x3ef
Selection: None
INS


seed@VM: .../Files
[12/01/22]seed@VM:.../Files$ openssl enc -aes-128-cbc -d -in output.bin -out dec
ryptedCBC.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
Open
decryptedCBC.txt
/mnt/Labsetup/Files
Save
1 123456789a123456789a123456789a123456789a12345678:Séñ
+ÊYc`Žnå"56789a1V3456789a123456789a123456789a123456789a123456789a123456789a123456789a123456789a123456789a123456789a12

CFB:

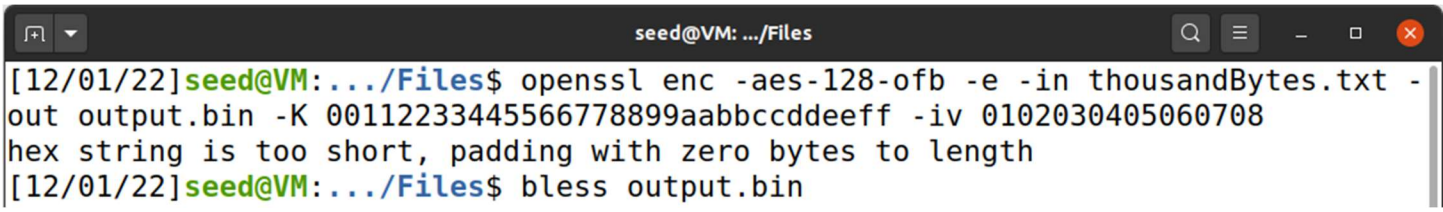

seed@VM: .../Files
[12/01/22]seed@VM:.../Files$ openssl enc -aes-128-cfb -e -in thousandBytes.txt -
out output.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[12/01/22]seed@VM:.../Files$ bless output.bin
/mnt/Labsetup/Files/output.bin - Bless
File Edit View Search Tools Help
output.bin
00000000 90 49 B5 09 35 F0 3A 6C 5C 56 C8 61 05 .I..5.:l\V.a.
0000000d 0F 76 ED 80 DB FA 26 8A FB C3 C2 9D 24 .v....&.....$
0000001a 23 E4 E9 95 2B AC 80 84 0A A8 83 DF 8F #...+.........
00000027 BB 7E 21 0B 54 5C F1 71 2F AB 20 01 F8 .~!.T\.q/. ..
00000034 78 84 71 02 FD E1 D0 D5 2B 76 0E 3D 83 x.q.....+v.=.
00000041 D7 A0 8B EB F8 A7 96 7C A8 A3 98 3E 71 ....|...>q
0000004e B3 57 8A E4 C6 D6 EA D5 EC 3F 9E E4 A6 .W.......?...
0000005b BF 64 5D AA 3E EC 25 17 EC C7 95 9E 36 .d].>.%.....6
00000068 CC 74 29 38 22 65 98 58 31 7F 9D E8 B1 .t)8"e.X1.....
00000075 98 E3 F2 D8 0E 2A DB F0 9D 3D AF FB 94 .....*...=...
00000082 3D D4 2A 72 1A 53 32 71 CF 7F 62 D5 1F =.*r.S2q..b..
0000008f A6 4E 39 5D 06 10 71 05 13 7D 6E 8B 20 .N9]..q..}n.
0000009c 6F 12 F4 C4 BF 3A 1E 00 2F 17 D1 9C 26 o....:../...&
000000a9 20 8D 4F 4F DE 85 78 5D B2 E8 1B 59 8C .OO..x]...Y.
000000b6 8B 6C 98 9E FA 13 6E 1B DB C8 A2 7E 14 .l....n....~.
000000c3 46 7A 8D 1A 50 AD E1 58 31 17 F9 FB 32 Fz..P..X1...2
000000d0 12 DF 21 DA 89 1E 4A 8B E4 3C 75 13 91 ..!...J..<u..
000000dd F6 E9 9A DA 1A D6 78 5B 0D AF 87 CE 15 ......x[.....
Offset: 0x0 / 0x3e8
Selection: None
INS

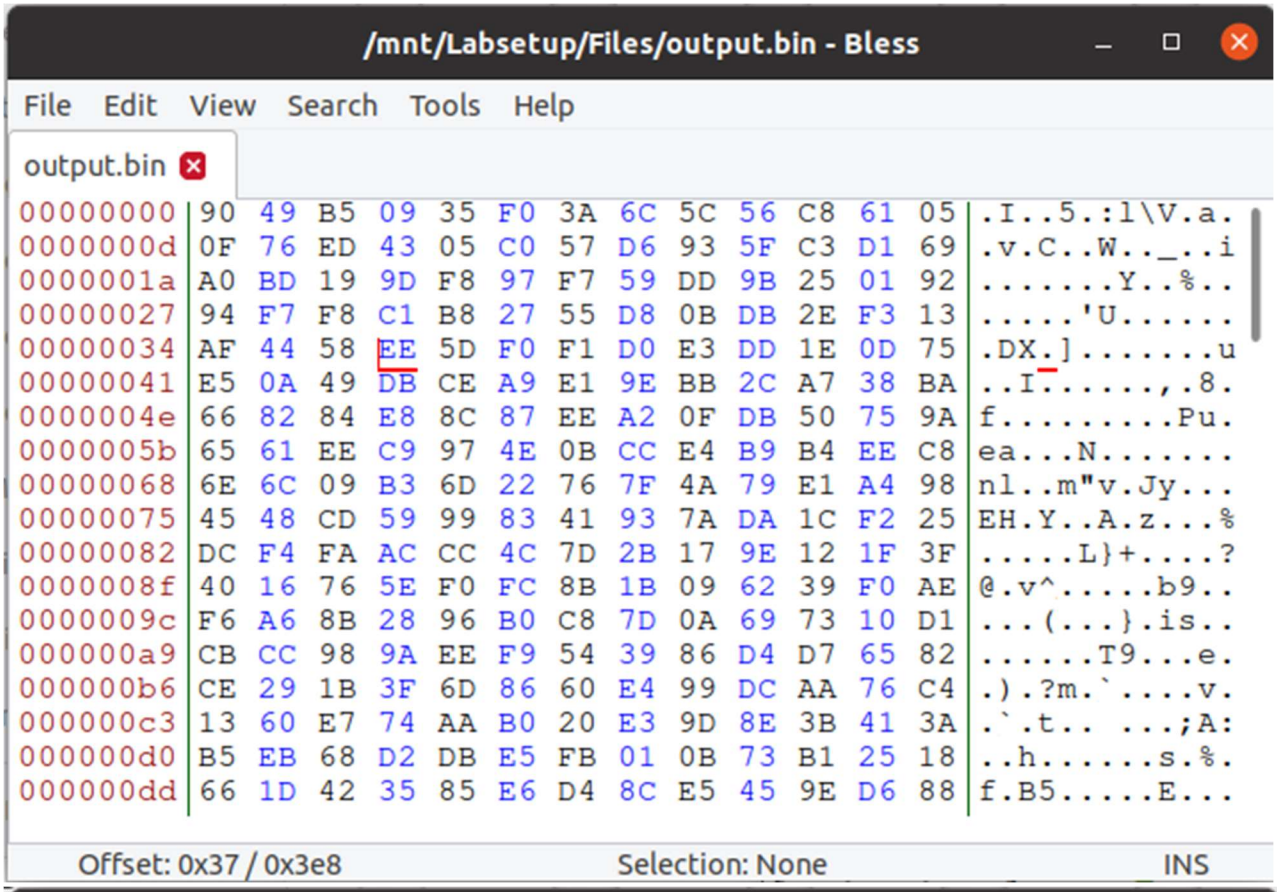

/mnt/Labsetup/Files/output.bin - Bless
File Edit View Search Tools Help
output.bin
00000000 90 49 B5 09 35 F0 3A 6C 5C 56 C8 61 05 .I..5.:l\V.a.
0000000d 0F 76 ED 80 DB FA 26 8A FB C3 C2 9D 24 .v....&.....$
0000001a 23 E4 E9 95 2B AC 80 84 0A A8 83 DF 8F #...+.......
00000027 BB 7E 21 0B 54 5C F1 71 2F AB 20 01 F8 .~!.T\.q/. ..
00000034 78 84 71 00 FD E1 D0 D5 2B 76 0E 3D 83 x.q.....+v.=.
00000041 D7 A0 8B EB F8 A7 96 7C A8 A3 98 3E 71 ....|...>q
0000004e B3 57 8A E4 C6 D6 EA D5 EC 3F 9E E4 A6 .W.......?...
0000005b BF 64 5D AA 3E EC 25 17 EC C7 95 9E 36 .d].>.%.....6
00000068 CC 74 29 38 22 65 98 58 31 7F 9D E8 B1 .t)8"e.X1....
00000075 98 E3 F2 D8 0E 2A DB F0 9D 3D AF FB 94 .....*...=...
00000082 3D D4 2A 72 1A 53 32 71 CF 7F 62 D5 1F =.*r.S2q..b..
0000008f A6 4E 39 5D 06 10 71 05 13 7D 6E 8B 20 .N9]..q..}n.
0000009c 6F 12 F4 C4 BF 3A 1E 00 2F 17 D1 9C 26 o....:../...&
000000a9 20 8D 4F 4F DE 85 78 5D B2 E8 1B 59 8C .OO..x]...Y.
000000b6 8B 6C 98 9E FA 13 6E 1B DB C8 A2 7E 14 .l....n....~.
000000c3 46 7A 8D 1A 50 AD E1 58 31 17 F9 FB 32 Fz..P..X1...2
000000d0 12 DF 21 DA 89 1E 4A 8B E4 3C 75 13 91 ..!...J..<u..
000000dd F6 E9 9A DA 1A D6 78 5B 0D AF 87 CE 15 ......x[.....
Offset: 0x37 / 0x3e8
Selection: None
INS

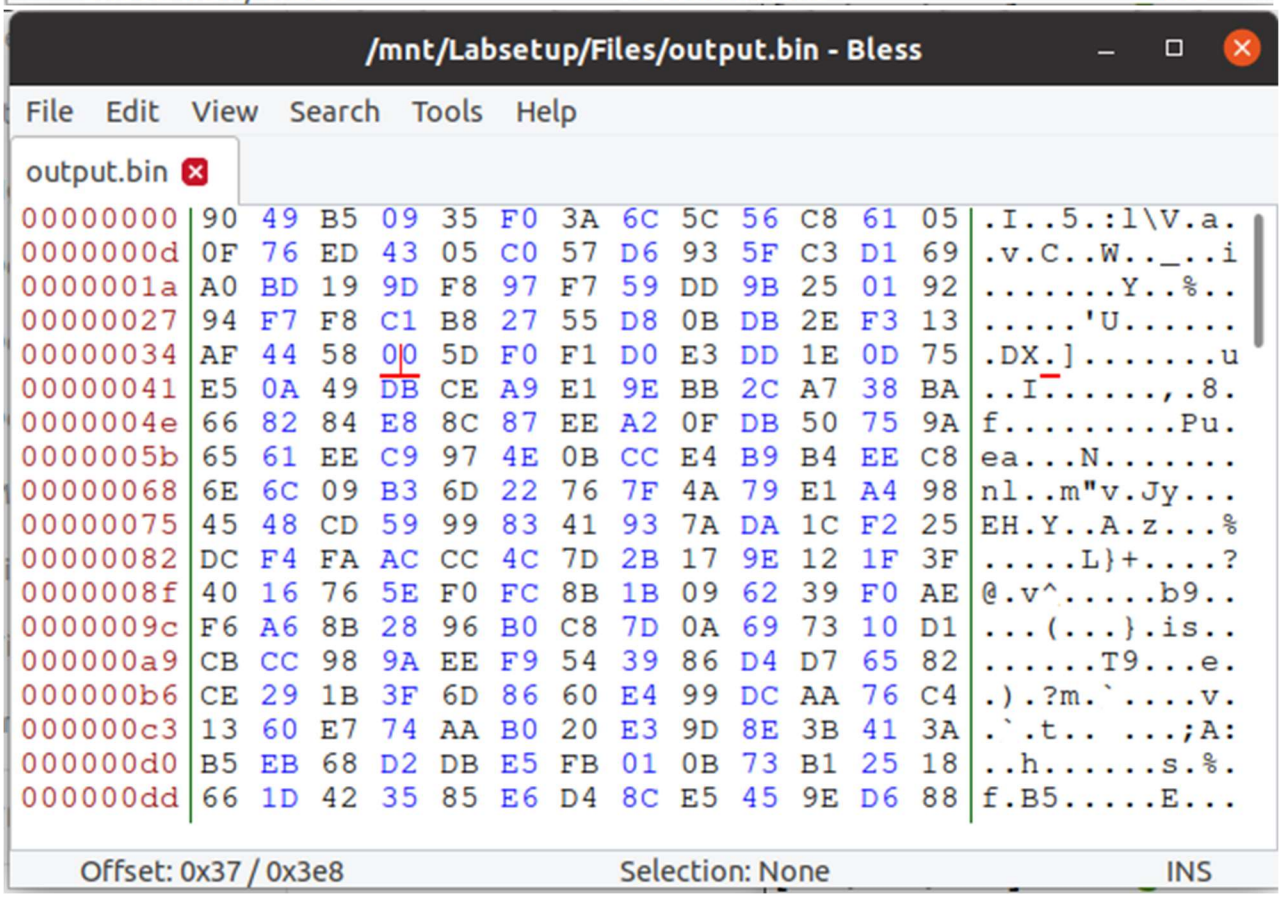

seed@VM: .../Files
[12/01/22]seed@VM:.../Files$ openssl enc -aes-128-cfb -d -in output.bin -out dec
ryptedCFB.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length

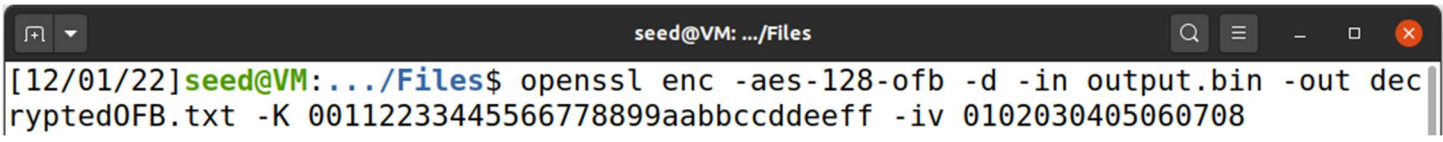
Open
decryptedCFB.txt
/mnt/Labsetup/Files
Save
1 123456789a123456789a123456789a123456789a123456789a123454789a12345O¿Ï▒oÏr,gÞmž:®#123456789a123456789a123456789a123456789a12345678

OFB:


seed@VM: .../Files
[12/01/22]seed@VM:.../Files$ openssl enc -aes-128-ofb -e -in thousandBytes.txt -
out output.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[12/01/22]seed@VM:.../Files$ bless output.bin


/mnt/Labsetup/Files/output.bin - Bless
File Edit View Search Tools Help
output.bin
00000000 90 49 B5 09 35 F0 3A 6C 5C 56 C8 61 05 .I..5.:l\V.a.
0000000d 0F 76 ED 43 05 C0 57 D6 93 5F C3 D1 69 .v.C..W.._..i
0000001a A0 BD 19 9D F8 97 F7 59 DD 9B 25 01 92 ....Y..%..
00000027 94 F7 F8 C1 B8 27 55 D8 0B DB 2E F3 13 .....'U.....
00000034 AF 44 58 EE 5D F0 F1 D0 E3 DD 1E 0D 75 .DX.].......u
00000041 E5 0A 49 DB CE A9 E1 9E BB 2C A7 38 BA ..I......,.8.
0000004e 66 82 84 E8 8C 87 EE A2 0F DB 50 75 9A f.........Pu.
0000005b 65 61 EE C9 97 4E 0B CC E4 B9 B4 EE C8 ea...N.......
00000068 6E 6C 09 B3 6D 22 76 7F 4A 79 E1 A4 98 nl..m"v.Jy...
00000075 45 48 CD 59 99 83 41 93 7A DA 1C F2 25 EH.Y..A.z...%
00000082 DC F4 FA AC CC 4C 7D 2B 17 9E 12 1F 3F .....L}+....?
0000008f 40 16 76 5E F0 FC 8B 1B 09 62 39 F0 AE @.v^.....b9..
0000009c F6 A6 8B 28 96 B0 C8 7D 0A 69 73 10 D1 ...(...}.is..
000000a9 CB CC 98 9A EE F9 54 39 86 D4 D7 65 82 ......T9...e.
000000b6 CE 29 1B 3F 6D 86 60 E4 99 DC AA 76 C4 .).?m.`....v.
000000c3 13 60 E7 74 AA B0 20 E3 9D 8E 3B 41 3A .`.t.. ...;A:
000000d0 B5 EB 68 D2 DB E5 FB 01 0B 73 B1 25 18 ..h......s.%.
000000dd 66 1D 42 35 85 E6 D4 8C E5 45 9E D6 88 f.B5.....E...
Offset: 0x37 / 0x3e8
Selection: None
INS


/mnt/Labsetup/Files/output.bin - Bless
File Edit View Search Tools Help
output.bin
00000000 90 49 B5 09 35 F0 3A 6C 5C 56 C8 61 05 .I..5.:l\V.a.
0000000d 0F 76 ED 43 05 C0 57 D6 93 5F C3 D1 69 .v.C..W.._..i
0000001a A0 BD 19 9D F8 97 F7 59 DD 9B 25 01 92 ....Y..%..
00000027 94 F7 F8 C1 B8 27 55 D8 0B DB 2E F3 13 .....'U.....
00000034 AF 44 58 00 5D F0 F1 D0 E3 DD 1E 0D 75 .DX.].......u
00000041 E5 0A 49 DB CE A9 E1 9E BB 2C A7 38 BA ..I......,.8.
0000004e 66 82 84 E8 8C 87 EE A2 0F DB 50 75 9A f.........Pu.
0000005b 65 61 EE C9 97 4E 0B CC E4 B9 B4 EE C8 ea...N.......
00000068 6E 6C 09 B3 6D 22 76 7F 4A 79 E1 A4 98 nl..m"v.Jy...
00000075 45 48 CD 59 99 83 41 93 7A DA 1C F2 25 EH.Y..A.z...%
00000082 DC F4 FA AC CC 4C 7D 2B 17 9E 12 1F 3F .....L}+....?
0000008f 40 16 76 5E F0 FC 8B 1B 09 62 39 F0 AE @.v^.....b9..
0000009c F6 A6 8B 28 96 B0 C8 7D 0A 69 73 10 D1 ...(...}.is..
000000a9 CB CC 98 9A EE F9 54 39 86 D4 D7 65 82 ......T9...e.
000000b6 CE 29 1B 3F 6D 86 60 E4 99 DC AA 76 C4 .).?m.`....v.
000000c3 13 60 E7 74 AA B0 20 E3 9D 8E 3B 41 3A .`.t.. ...;A:
000000d0 B5 EB 68 D2 DB E5 FB 01 0B 73 B1 25 18 ..h......s.%.
000000dd 66 1D 42 35 85 E6 D4 8C E5 45 9E D6 88 f.B5.....E...
Offset: 0x37 / 0x3e8
Selection: None
INS


seed@VM: .../Files
[12/01/22]seed@VM:.../Files$ openssl enc -aes-128-ofb -d -in output.bin -out dec
ryptedOFB.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708

Results:

ECB: We were correct.

CBC: We were correct.

CFB: Our prediction was somewhat correct, but the scope was inaccurate. Not all data was corrupted.

OFB: Our prediction was entirely incorrect. Only the character(s) represented by the data corrupted would be lost.

Task 6. Initial Vector (IV) and Common Mistakes

Task 6.1.

When the plaintexts are the same, ciphertext would also be the same if the IV remain unchanged. Therefore, it's important to keep the IV unique and update it for every encryption.

Task 6.2.

In OFB encryption mode, if the key and the IV are unchanged, there's a great chance of 'Known-Plaintext' attack. The output stream is obtained as the result of block-by-block XOR or plaintext and ciphertext. Therefore, to get the plaintext, plaintext and ciphertext can be XOR-ed. In this mode, the output streams of the same key and the IV are identical compared to other encryption modes.

If the plaintext (`pk`) and its OFB ciphertext (`ck`), and another OFB ciphertext (`co`) with same key and IV are known, to find the unknown plaintext (`po`) of co we need to get the output stream of the encryption of the `pk` plaintext:

```
output = pk XOR ck
po = output XOR co
po = pk XOR ck XOR co
```

Given following code:

```
Plaintext  (P1): This is a known message!
Ciphertext (C1): a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159

Plaintext  (P2): (unknown to you)
Ciphertext (C2): bf73bcd3509299d566c35b5d450337e1bb175f903fafc159
```

We used the following python code to de/encrypt the ciphertext:

```
from sys import afgv
_, first, second, third = argv
pk = bytearray(first,encoding='utf-8')
ck = bytearray.fromhext(second)
co = bytearray.fromhex(third)
po = bytearray(x^y^z for x, y, z, in zip(pk, ck, co))
print(po.decode('utf-8'))
```

In the CFB mode, the procedure would stay the same for initial block, but the key would remain a secret and the proceeding parts of the ciphertext would not be revealed.

Task 6.3.

Our initial assumption was pk (Bob's initial message) was "Yes."

`po = "Yes" XOR IV XOR IV_NEXT`, where the IV used was the same to generate the ck and IV_NEXT, making IV predictable to be used to encrypt the plaintext input.

```
Encryption mode used: 128-bit AES (CBC mode).
Hex key: 00112233445566778899aabbccddeeff
Ciphertext (ck): bef6556557ccee2a9f9553154ed9498
```

```
ASCII IV of pk: 1234567890123456
Hex IV of pk: 31323333435363738393031323334353 6
New ASCII IV: 1234567890123457
New Hex IV: 31323334353637383930313233343537
```

Due to the length of the payload, PKCS#7 padding and code adoption are required:

```
from sys import afgv
_, first, second, third = argv
pk = bytearray(first,encoding='utf-8')
padding = 16 – len(p1) % 16
pk.extend([padding] * padding)
IV = bytearray.fromhext(second)
IV_Next = bytearray.fromhex(third)
po = bytearray(x^y^z for x, y, z, in zip(pk, IV, IV_Next))
print(po.decode('utf-8'), end='')
```

Plaintext is denoted as a multiple of 16 bytes; therefore, PKCS#7 padding is required for encryption. Lastly, we compared ck with co to get the result of the first 16 bites of co the same as ck. Ultimately, our assumption was correct, and Bob's initial message was "Yes."

Task 7. Programming using the Crypto Library

```
// Author: Nathan Mahnke (with massive credit to stackoverflow for a lot of the algorithms
used)
#include <openssl/conf.h>
#include <openssl/evp.h>
#include <openssl/err.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
// This is a helper function for hex_to_ascii. I found this stackoverflow post regarding the
concept: https://stackoverflow.com/questions/5403103/hex-to-ascii-string-conversion. This
// code receives a character and then returns it's integer value assuming the input was in
hex.
int hex_to_int(char c) {
    int first = c / 16 - 3;
    int second = c % 16;
    int result = first * 10 + second;
    if(result > 9) {
        result--;
        return result;
    }
}
// This function receives two hex characters and returns the int value assuming the int
returned corresponds to the ASCII value of the incoming hex characters.
int hex_to_ascii(char c, char d) {
    int high = hex_to_int(c) * 16;
    int low = hex_to_int(d);
    return high + low;
}
int main(int arc, char *argv[]) {
    // cipher is a buffer array that is used to store encrypted strings
    unsigned char cipher[1024];
    // outbuf is a buffer array that is used to store output strings
    unsigned char outbuf[1024];
    // temp is a storage variable
```

```
    // key is the where the key string used for encryption will be stored when/if it is found
    unsigned char temp, key[16];
    // These are all various variables used for counting and storage
    int outLength, tempLength, l, i, length, count, found = 0, k = 0;
    // An input file which will be used to read in the list of English words
    FILE *in;
    // A char array that will story the initialization vector
    unsigned char iv[17];
    // Populating the IV
    for(i = 0; i < 17; i++) {
        iv[i] = 0;
        iv[16] = '\0';
    }
    // The text to be encrypted
    char intext[] = "This is a top secret.";
    // SHA-256 key
    char st[] = "8d20e5056a8d24d0462ce74e4904c1b513e10d1df4a2ef2ad4540fae1ca0aaf9";
    // Conver st[] to all capital letters
    i = 0;
    while(i < 64) {
        if(st[i] >= 'a' && st[i] <= 'z')
        st[i] = st[i] - 32;
        i++;
    }
    // Populating cipher[]
    length = strlen(st);
    char buf = 0;
    for(i = 0; i < length; i++) {
        if(i % 2 != 0) {
            cipher[k] = hex_to_ascii(buf, st[i]);
            k++;
        } else {
            buf = st[i];
        }
    }
    cipher[k] = '\0';
    // Opening the file of English words for reference
    in = fopen("/mnt/Labsetup/Files/words.txt", "r");
    if(in == NULL) {
        printf("\n cannot open file");
        exit(1);
    }
    // Declaring the EVP cipher
    EVP_CIPHER_CTX *ctx;
    ctx = EVP_CIPHER_CTX_new();
    // Set values of key and attempt decryption. If successfull, ouput plain text and key,
otherwise output failure.
    while(fgets(key, sizeof(key), in) != NULL) {
        l = 0;
        if(strlen(key) < 16) {
            l = strlen(key) - 1;
            while(l < 16) {
                key[l] = ' ';
                l++;
            }
            key[l] = '\0';
        } else {
            key[16] = '\0';
```

```
        }
        // Compare resulting values to that of the input, if they do not match, return 0,
meaning move to the next word
        EVP_EncryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv);
        if(!EVP_EncryptUpdate(ctx, outbuf, &outLength, intext, strlen(intext))) {
            return 0;
        }
        if(!EVP_EncryptFinal_ex(ctx, outbuf + outLength, &tempLength)){
            return 0;
        }
        // If this portion of the code executes, the key has been proven valid, thus we
cleanup the EVP cipher addresses,
        // write outbuf into cipher, and output the plain text along with the key
        outLength += tempLength;
        EVP_CIPHER_CTX_cleanup(ctx);
        count = 0;
        for(i = 0; i < 32; i++) {
            if(cipher[i] == outbuf[i])
            count++;
        }
        if(count == 32) {
            printf("Plain text:\n%s\n",intext);
            printf("Key:\n%s\n",key);
            found = 1;
            break;
        }
    }
    // If this potion of the code executes, a key could not be found within the words.txt
file that results in a matching
    // encryption, thus the search failed. The input is closed and the program is terminated
    if(found == 0) {
        printf("\nCould not find key\n");
    }
    fclose(in);
    return 0;
}
```