Network Implementation & Security

Secret-Key Encryption Lab

Jin Jung, Joshua Barrs, John Park

Objective

2.1 Task 1: Frequency Analysis Against Monoalphabetic Substitution Cipher

Step 1:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ tr [:upper:] [:lower:] <article.txt> l
owercase.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat lowercase.txt
substitution ciphers are probably the most common form of cipher.
they work by replacing each letter of the plaintext and sometimes puntuation mar
ks and spaces with another letter
or possibly even a random symbol.

a monoalphabetic substitution cipher, also known as a simple substitution cipher
,
relies on a fixed replacement structure. that is, the substitution is fixed for
each letter of the alphabet.
thus, if "a" is encrypted to "r", then every time we see the letter "a" in the p
laintext,
we replace it with the letter "r" in the ciphertext

a simple example is where each letter is encrypted as the next letter in the alp
habet:
"a simple message" becomes "b tjnqmf nfttbhf". in general, when performing a sim
ple substitution manually,
it is easiest to generate the ciphertext alphabet first, and encrypt by comparin
g this to the plaintext alphabet.
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plaintext.txt
substitution ciphers are probably the most common form of cipher
they work by replacing each letter of the plaintext and sometimes puntuation marks and spaces with another let
ter
or possibly even a random symbol

a monoalphabetic substitution cipher also known as a simple substitution cipher
relies on a fixed replacement structure that is the substitution is fixed for each letter of the alphabet
thus if a is encrypted to r then every time we see the letter a in the plaintext
we replace it with the letter r in the ciphertext

a simple example is where each letter is encrypted as the next letter in the alphabet
a simple message becomes b tjnqmf nfttbhf in general when performing a simple substitution manually
it is easiest to generate the ciphertext alphabet first and encrypt by comparing this to the plaintext alphabe
t
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

Step 2: Permute the alphabet to generate the encryption key

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> s="abcdefghijklmnopqrstuvwxyz"
>>> list = random.sample(s, len(s))
>>> ''.join(list)
'xtyhkwnpqrocealusfzjibmdvg'
>>>
```

Step 3: encrypt the plaintext.txt

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ tr 'abcdefghijklmnopqrstuvwxyz' 'xtyhkwnpqrocealusfzjibmdvg' <plaint
ext.txt> ciphertext.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat ciphertext.txt
zitzjqjijqla yqupkfz xfk ufltxtcv jpk elzj yleela wlfe lw yqupkf
jpkv mlfo tv fkucxyqan kxyp ckjjkf lw jpk ucxqajkdj xah zlekjqekz uiajixjqla exfoz xah zuxykz mqjp xaljpkf ckj
jkf
lf ulzzqtcv kbka x fxahle zvetlc

x elalxcupxtkjqy zitzjqjijqla yqupkf xczl oalma xz x zqeuck zitzjqjijqla yqupkf
fkcqkz la x wqdkh fkucxykekaj zjfiyjifk jpxj qz jpk zitzjqjijqla qz wqdkh wlf kxyp ckjjkf lw jpk xcupxtkj
jpiz qw x qz kayfvujkh jl f jpka kbkfv jqek mk zkk jpk ckjjkf x qa jpk ucxqajkdj
mk fkucxyk qj mqjp jpk ckjjkf f qa jpk yqupkfjkdj

x zqeuck kdxeuck qz mpkfk kxyp ckjjkf qz kayfvujkh xz jpk akdj ckjjkf qa jpk xcupxtkj
x zqeuck ekzzxnk tkylekz t jrasew awjjtpw qa nkakfxc mpka ukfwlfeqan x zqeuck zitzjqjijqla exaixccv
qj qz kxzqkzj jl nkakfxjk jpk yqupkfjkdj xcupxtkj wqfzj xah kayfvuj tv yleuxfqan jpqz jl jpk ucxqajkdj xcupxtk
j
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

Cipher file from blackboard:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat ciphertext
ytn numrcv cvatmunq lnhn v qnhmnq xb ninayhxcnatvumavi hxyxh ametnh
cvatmunq pnfnixenp vup zqnp mu ytn nvhid yx cmpyt anuyzhd yx
ehxynay axccnhamvi pmeixcvyma vup cmimyvhd axcczumavymxu numrcv lvq
mufnuynp gd ytn rnhcvu nurmunnh vhytzh qatnhgmzq vy ytn nup xb
lxhip lvh m nvhid cxpniq lnhn zqnp axccnhamviid bhxc ytn nvhid q
vup vpxeynp gd cmimyvhd vup rxfnhucnuy qnhfmanq xb qnfnhvi
axzuyhmnq cxqy uxyvgid uvwm rnhcvud gnbxhn vup pzhmur lxhip lvh mm
qnfnhvi pmbbnhnuy numrcv cxpniq lnhn ehxpzanp gzy ytn rnhcvu
cmimyvhd cxpniq tvfmur v eizrgxvhp lnhn ytn cxqy axceink ovevunqn
vup myvimvu cxpniq lnhn viqx mu zqn ...

[03/04/20]seed@VM:~/crypto_encryptionlab$
```

The ciphertext from blackboard was decrypted iteratively by the following guesses:

Guess 1: tr 'vm' 'AI' <ciphertext> guess1.txt

Guess 2: tr 'up' 'ND' <guess1.txt> guess2.txt

Guess 3: tr 'ytn' 'THE' <guess2.txt> guess3.txt

Guess 4: tr 'fr' 'VG' <guess3.txt> guess4.txt

Guess 5: tr 'caixe' 'MCLOP' <guess4.txt> guess5.txt

Guess 6: tr 'qhdzbl' 'SRYUFW' <guess5.txt> guess6.txt

Guess 7: tr 'gwko' 'BZXJ' <guess6.txt> guess7.txt

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ tr 'gwko' 'BZXJ' <guess6.txt> guess7.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat guess7.txt
THE ENIGMA MACHINES WERE A SERIES OF ELECTROMECHANICAL ROTOR CIPHER
MACHINES DEVELOPED AND USED IN THE EARLY TO MIDTH CENTURY TO
PROTECT COMMERCIAL DIPLOMATIC AND MILITARY COMMUNICATION ENIGMA WAS
INVENTED BY THE GERMAN ENGINEER ARTHUR SCHERBIUS AT THE END OF
WORLD WAR I EARLY MODELS WERE USED COMMERCIALLY FROM THE EARLY S
AND ADOPTED BY MILITARY AND GOVERNMENT SERVICES OF SEVERAL
COUNTRIES MOST NOTABLY NAZI GERMANY BEFORE AND DURING WORLD WAR II
SEVERAL DIFFERENT ENIGMA MODELS WERE PRODUCED BUT THE GERMAN
MILITARY MODELS HAVING A PLUGBOARD WERE THE MOST COMPLEX JAPANESE
AND ITALIAN MODELS WERE ALSO IN USE ...

[03/04/20]seed@VM:~/crypto_encryptionlab$
```

2.2 Task 2: Encryption using Different Ciphers and Modes

Using different ciphers in the openssl enc cryptographic library

1st cipher: Advanced Encryption Algorithm 256 output feedback mode using password:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain.txt
This sentence should be kept confidential.x
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-256-ofb -in plain.txt -out cipher.txt -e
enter aes-256-ofb encryption password:
Verifying - enter aes-256-ofb encryption password:
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat cipher.txt
Salted__&F@jA(G4,^[illegible]k@@1t[illegible]w[illegible]@@~N[illegible]@@8@@^IY@u@@@2:g>@@&@M@[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc
-aes-256-ofb -in cipher.txt -out plain2.txt -d
enter aes-256-ofb decryption password:
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain2.txt
This sentence should be kept confidential.x
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

2nd cipher: AES-192 Cipher Feedback Mode using cipher key and initialization vector

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain.txt
This sentence should be kept confidential.x
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-192-cfb -in plain.txt -out cipher.txt -e -K 00112233445566778899aabbccddeef
f0011223344556677 -iv 00102030405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat cipher.txt
B+@@JW@
      @@+@@[illegible]7[illegible]{@~@@@[@jv@*@@@Ov@@@d@@b[openssl enc -aes-192-cfb -in cipher.txt -out plain2.txt -d -K 00112233445566778899aabbccddee
ff0011223344556677 -iv 00102030405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain2.txt
This sentence should be kept confidential.x
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

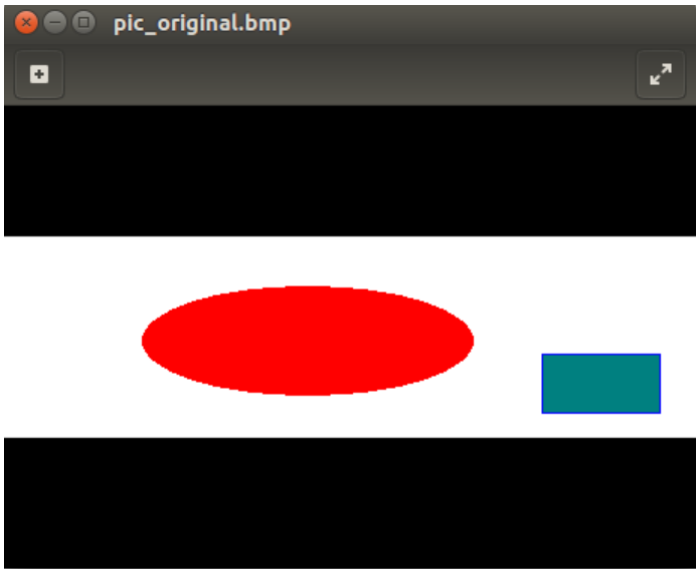
3rd cipher: AES-128 Electronic Codebook mode using cipher key

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain.txt
This sentence should be kept confidential.x
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-ecb -in plain.txt -out cipher.txt -e -K 00112233445566778899aabbccddeef
f
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat cipher.txt
@@@N[illegible],@@@
        s@#@@@y@@U@@n@@@@@i"]@@qE[illegible]@@%3y[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-ecb -in cipher.txt -out plain2.
fft -d -K 00112233445566778899aabbccddee
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain2.txt
This sentence should be kept confidential.x
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

4th cipher: Triple Data Encryption Standard

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain2.txt
This sentence should be kept confidential.x
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -des-ede3-cbc -in plain.txt -out cipher.txt -e
enter des-ede3-cbc encryption password:
Verifying - enter des-ede3-cbc encryption password:
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat cipher.txt
Salted__@@(qn@@[l@@\@> @@@@@@@r8@@@@@@[illegible]L@i=@@@[illegible]T@@@P@9@@@u~@@[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -des-ede3-cbc -in pl
ain.txt -out cipher.txt -e                openssl enc -des-ede3-cbc -in cipher.txt -out plain2.txt -d
enter des-ede3-cbc decryption password:
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain2.txt
This sentence should be kept confidential.x
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

2.3 Task 3: Encryption Mode - ECB vs. CBC

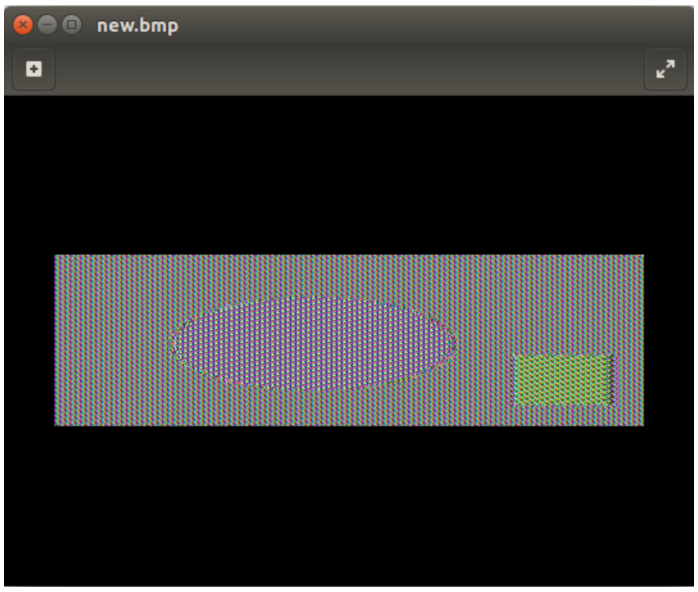


Step 1: encrypt file using AES-128-ECB and AES-128-CBC:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-ecb -in pic_original.bmp -out ecbcipher -e -K 00112233445566778899aabbc
cddeeff
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in pic_original.bmp -out cbccipher -K 00112233445566778899aabbccdd
eeff -iv 00102030405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

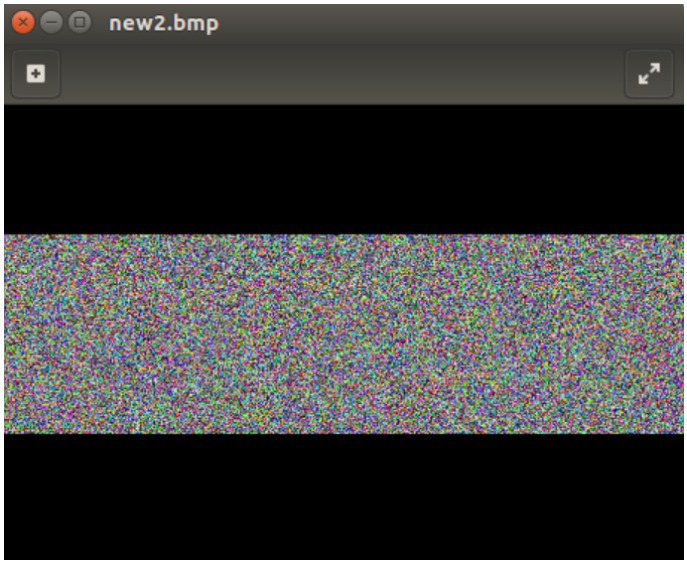
Step 2: take first 54 bits of original .bmp file and replace the first 54bits of encrypted files:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ head -c 54 pic_original.bmp > header
[03/04/20]seed@VM:~/crypto_encryptionlab$ tail -c +55 ecbcipher > body
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat header body > new.bmp
[03/04/20]seed@VM:~/crypto_encryptionlab$
```



With ECB, you can still make out the some of the original image

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ head -c 54 pic_original.bmp > header
[03/04/20]seed@VM:~/crypto_encryptionlab$ tail -c +55 cbccipher > body
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat header body > new2.bmp
[03/04/20]seed@VM:~/crypto_encryptionlab$
```



It seems like CBC is much more effective in obfuscating .bmp files.

2.4 Task 4: Padding

Step 1: Using different encryption modes to encrypt a file

For ECB:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain.txt
0123456789
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-ecb -in plain.txt -out cipher.txt -e -K 00112233445566778899aabbccddeef
f
[03/04/20]seed@VM:~/crypto_encryptionlab$ ls -ld plain.txt cipher.txt
-rw-rw-r-- 1 seed seed 16 Mar  4 16:22 cipher.txt
-rw-rw-r-- 1 seed seed 11 Mar  4 16:19 plain.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

The plain.txt has 11 bytes while cipher.txt has been padded to 16 bytes.

For CBC:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain.txt
0123456789
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in plain.txt -out cipher.txt -e -K 00112233445566778899aabbccddeef
f -iv 00102030405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ ls -ld plain.txt cipher.txt
-rw-rw-r-- 1 seed seed 16 Mar  4 16:26 cipher.txt
-rw-rw-r-- 1 seed seed 11 Mar  4 16:19 plain.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

Same results as ECB.

For CFB:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain.txt
0123456789
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cfb -in plain.txt -out cipher.txt -e -K 00112233445566778899aabbccddeef
f -iv 00102030405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ ls -ld plain.txt cipher.txt
-rw-rw-r-- 1 seed seed 11 Mar  4 16:27 cipher.txt
-rw-rw-r-- 1 seed seed 11 Mar  4 16:19 plain.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

Both the plain.txt and cipher.txt files remain at 11 bytes. No padding added. This is probably because there are no fixed block size requirements for stream cipher modes.

For OFB:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain.txt
0123456789
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-ofb -in plain.txt -out cipher.txt -e -K 00112233445566778899aabbccddeef
f -iv 00102030405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ ls -ld plain.txt cipher.txt
-rw-rw-r-- 1 seed seed 11 Mar  4 16:32 cipher.txt
-rw-rw-r-- 1 seed seed 11 Mar  4 16:19 plain.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

Same results as CFB. No padding added. Output Feedback Mode is also a stream cipher that does not need to encrypt data in fixed size chunks.

Step 2: comparing padding with different size plaintext files:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ ls -ld f1.txt f2.txt f3.txt
-rw-rw-r-- 1 seed seed  5 Mar  4 16:41 f1.txt
-rw-rw-r-- 1 seed seed 10 Mar  4 16:42 f2.txt
-rw-rw-r-- 1 seed seed 16 Mar  4 16:43 f3.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in f1.txt -out c1.txt -e -K 00112233445566778899aabbccddeeff -iv 0
0102030405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in f2.txt -out c2.txt -e -K 00112233445566778899aabbccddeeff -iv 0
0102030405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in f3.txt -out c3.txt -e -K 00112233445566778899aabbccddeeff -iv 0
0102030405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ ls -ld c1.txt c2.txt c3.txt
-rw-rw-r-- 1 seed seed 16 Mar  4 16:45 c1.txt
-rw-rw-r-- 1 seed seed 16 Mar  4 16:45 c2.txt
-rw-rw-r-- 1 seed seed 32 Mar  4 16:45 c3.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

The first 2 files are padded to a single 128 bit block while the last 16 byte file is given an additional 128 bits in padding.

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in c1.txt -out f1.txt -d -K 00112233445566778899aabbccddeeff -iv 0
0102030405060708090a0b0c0d0e0f0 -nopad
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in c2.txt -out f2.txt -d -K 00112233445566778899aabbccddeeff -iv 0
0102030405060708090a0b0c0d0e0f0 -nopad
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in c3.txt -out f3.txt -d -K 00112233445566778899aabbccddeeff -iv 0
0102030405060708090a0b0c0d0e0f0 -nopad
[03/04/20]seed@VM:~/crypto_encryptionlab$ ls -ld f1.txt f2.txt f3.txt
-rw-rw-r-- 1 seed seed 16 Mar  4 16:51 f1.txt
-rw-rw-r-- 1 seed seed 16 Mar  4 16:52 f2.txt
-rw-rw-r-- 1 seed seed 32 Mar  4 16:52 f3.txt
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ xxd f1.txt
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b  12345...........
[03/04/20]seed@VM:~/crypto_encryptionlab$ xxd f2.txt
00000000: 3132 3334 3536 3738 3961 0606 0606 0606  123456789a......
[03/04/20]seed@VM:~/crypto_encryptionlab$ xxd f3.txt
00000000: 3132 3334 3536 3738 3961 6263 6465 6630  123456789abcdef0
00000010: 1010 1010 1010 1010 1010 1010 1010 1010  ................
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

We can see that f1.txt (originally 5 bytes) was padded with 0x0b additional bytes to total 16 bytes, f2.txt (originally 10 bytes) was padded with 0x06 additional bytes to total 16 bytes, and f3.txt (originally 16 bytes) was padded with 0x10 additional bytes to total 32 bytes.

2.5 Task 5: Error Propagation - Corrupted Cipher Text

How much would corrupting a single bit of the 55th byte of the ciphertext affect different encryption modes:

Hypothesis:

ECB: Should be able to recover the most out of all the modes, since it has the simplest algorithm that uses the same ciphertext to decrypt each block of data.

CBC: Data should be a bit more corrupt than ECB since the output cipher of each previous data block is used as the IV of the next block of text.

CFB: Data should be less recoverable than ECB and CBC since the output cipher of each previous data block is used as the IV of the next block of text, and since it is a stream cipher, there may be additional complex implementations to address the lack of fixed block size requirement. Then, a single bit of corruption should affect a larger portion of data.

OFB: Data should be more recoverable than CFB or CBC since the one portion of ciphertext does not affect the input IV of following blocks of data.

Observation:

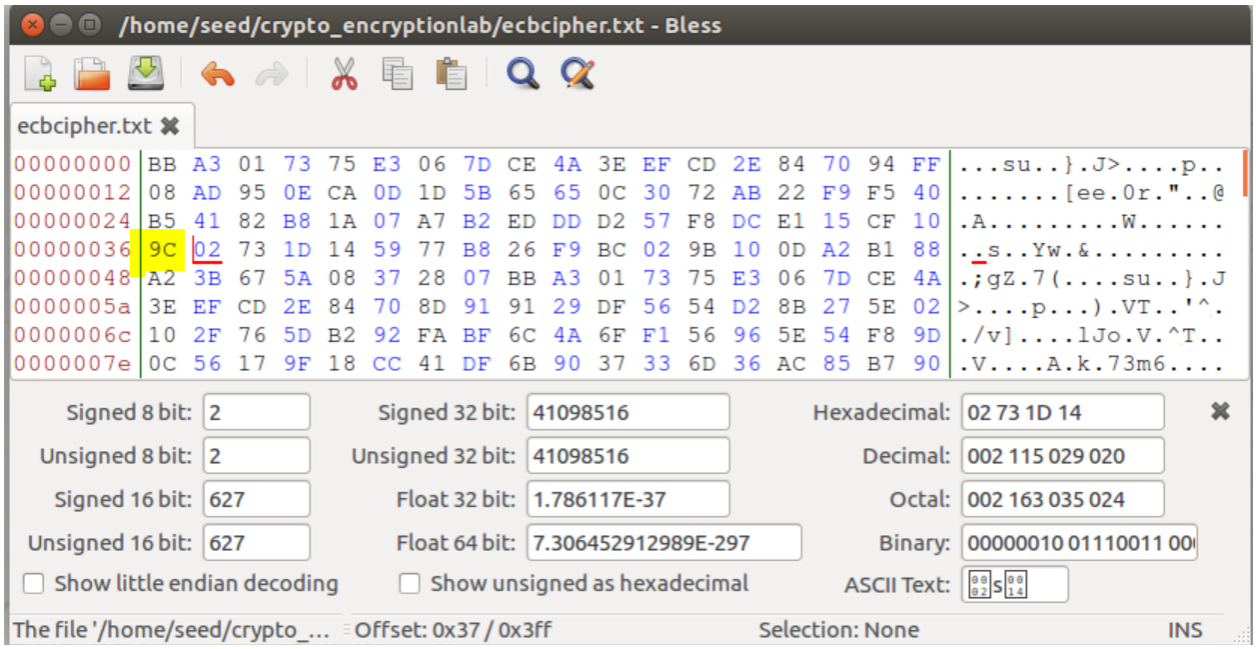
Created a plaintext file of size greater than 1000 bytes:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat plain.txt
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
[03/04/20]seed@VM:~/crypto_encryptionlab$ █
```

ECB:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-ecb -in plain.txt -out ecbcipher.txt -e -K 00112233445566778899aabbccddeeff
[03/04/20]seed@VM:~/crypto_encryptionlab$ xxd ecbcipher.txt
00000000: bba3 0173 75e3 067d ce4a 3eef cd2e 8470  ...su..}.J>....p
00000010: 94ff 08ad 950e ca0d 1d5b 6565 0c30 72ab  .........[ee.0r.
00000020: 22f9 f540 b541 82b8 1a07 a7b2 eddd d257  "..@.A.........W
00000030: f8dc e115 cf10 9d02 731d 1459 77b8 26f9  ........s..Yw.&.
00000040: bc02 9b10 0da2 b188 a23b 675a 0837 2807  .........;gZ.7(.
00000050: bba3 0173 75e3 067d ce4a 3eef cd2e 8470  ...su..}.J>....p
00000060: 8d91 9129 df56 54d2 8b27 5e02 102f 765d  ...).VT..'^../v]
00000070: b292 fabf 6c4a 6ff1 5696 5e54 f89d 0c56  ....lJo.V.^T...V
00000080: 179f 18cc 41df 6b90 3733 6d36 ac85 b790  ....A.k.73m6....
00000090: b59a 982d 61d9 b2b5 5e56 430c 3086 551b  ...-a...^VC.0.U.
000000a0: 7570 5151 2fd8 ad50 71b4 3f42 3cae 0d3b  upQQ/..Pq.?B<..;
000000b0: 513b 412d 9a12 9d5d 06ca 16b0 b576 d623  Q;A-...].....v.#
000000c0: 9c41 658e 9a87 9a73 d048 b22a 9813 0f00  .Ae....s.H.*....
000000d0: 94ff 08ad 950e ca0d 1d5b 6565 0c30 72ab  .........[ee.0r.
000000e0: 22f9 f540 b541 82b8 1a07 a7b2 eddd d257  "..@.A.........W
000000f0: f8dc e115 cf10 9d02 731d 1459 77b8 26f9  ........s..Yw.&.
00000100: bc02 9b10 0da2 b188 a23b 675a 0837 2807  .........;gZ.7(.
00000110: bba3 0173 75e3 067d ce4a 3eef cd2e 8470  ...su..}.J>....p
00000120: 65d5 d7e3 cf45 60e5 f887 fecd 7817 a840  e....E`.....x..@
00000130: b292 fabf 6c4a 6ff1 5696 5e54 f89d 0c56  ....lJo.V.^T...V
00000140: 179f 18cc 41df 6b90 3733 6d36 ac85 b790  ....A.k.73m6....
00000150: b59a 982d 61d9 b2b5 5e56 430c 3086 551b  ...-a...^VC.0.U.
00000160: 7570 5151 2fd8 ad50 71b4 3f42 3cae 0d3b  upQQ/..Pq.?B<..;
00000170: 513b 412d 9a12 9d5d 06ca 16b0 b576 d623  Q;A-...].....v.#
```

Change the 55th byte 0x9D by 1 bit -> 0x9C using bless hex editor



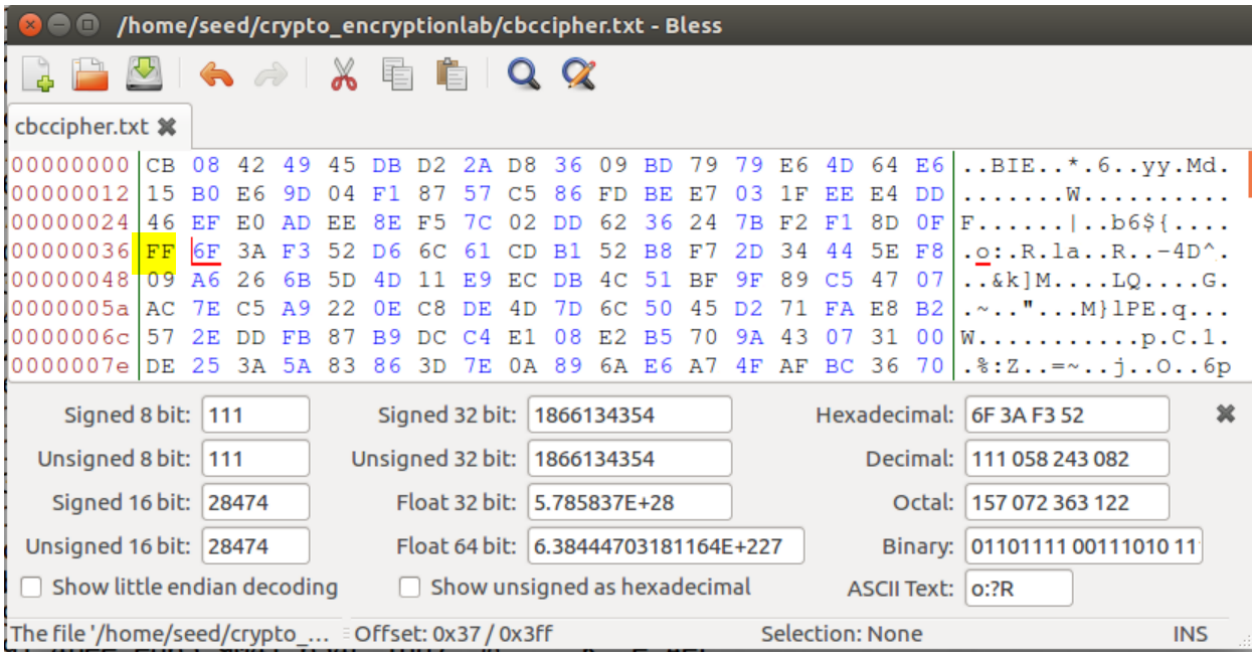

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-ecb -in ecbcipher.txt -out ecbplain.txt -d -K 00112233445566778899aabbccddeeff
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat ecbplain.txt
1234567890123456789012345678901234567890123456)@G@    \@@3@@@6@@&567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

Changing 1 bit corrupted about 16 bytes of data

CBC:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in plain.txt -out cbccipher.txt -e -K 00112233445566778899aabbccddeeff -iv 001020304
05060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ xxd cbccipher.txt
00000000: cb08 4249 45db d22a d836 09bd 7979 e64d  ..BIE..*.6..yy.M
00000010: 64e6 15b0 e69d 04f1 8757 c586 fdbe e703  d........W......
00000020: 1fee e4dd 46ef e0ad ee8e f57c 02dd 6236  ....F......|..b6
00000030: 247b f2f1 8d0f fe6f 3af3 52d6 6c61 cdb1  ${.....o:.R.la..
00000040: 52b8 f72d 3444 5ef8 09a6 266b 5d4d 11e9  R..-4D^...&k]M..
00000050: ecdb 4c51 bf9f 89c5 4707 ac7e c5a9 220e  ..LQ....G..~..".
00000060: c8de 4d7d 6c50 45d2 71fa e8b2 572e ddfb  ..M}lPE.q...W...
00000070: 87b9 dcc4 e108 e2b5 709a 4307 3100 de25  ........p.C.1..%
00000080: 3a5a 8386 3d7e 0a89 6ae6 a74f afbc 3670  :Z..=~..j..O..6p
00000090: 020a 0a98 b89c 81b4 fb97 cdd5 0b14 b1d4  ................
000000a0: 4ce0 0513 4064 8ac5 7c31 46d2 0a24 693e  L...@d..|1F..$i>
000000b0: 4577 bce7 c9f4 6830 8951 fb61 e21b 5515  Ew....h0.Q.a..U.
000000c0: c65b b78b 2d39 7399 18d3 72c5 6df8 c7d9  .[..-9s...r.m...
000000d0: 2808 14ee 54c3 1084 6127 c808 8507 822a  (...T...a'.....*
000000e0: 7d87 7b72 cd28 37f1 04aa a197 5910 14ff  }.{r.(7.....Y...
000000f0: e450 5222 ad05 db99 f1ef a8d1 5b9d 85a5  .PR".........[...
00000100: 7333 ed6d b050 bf79 7031 3e65 3a43 40c9  s3.m.P.yp1>e:C@.
```

Change the 55th byte of cbccipher.txt 0xFE by 1 bit -> 0xFF using the bless hex editor:



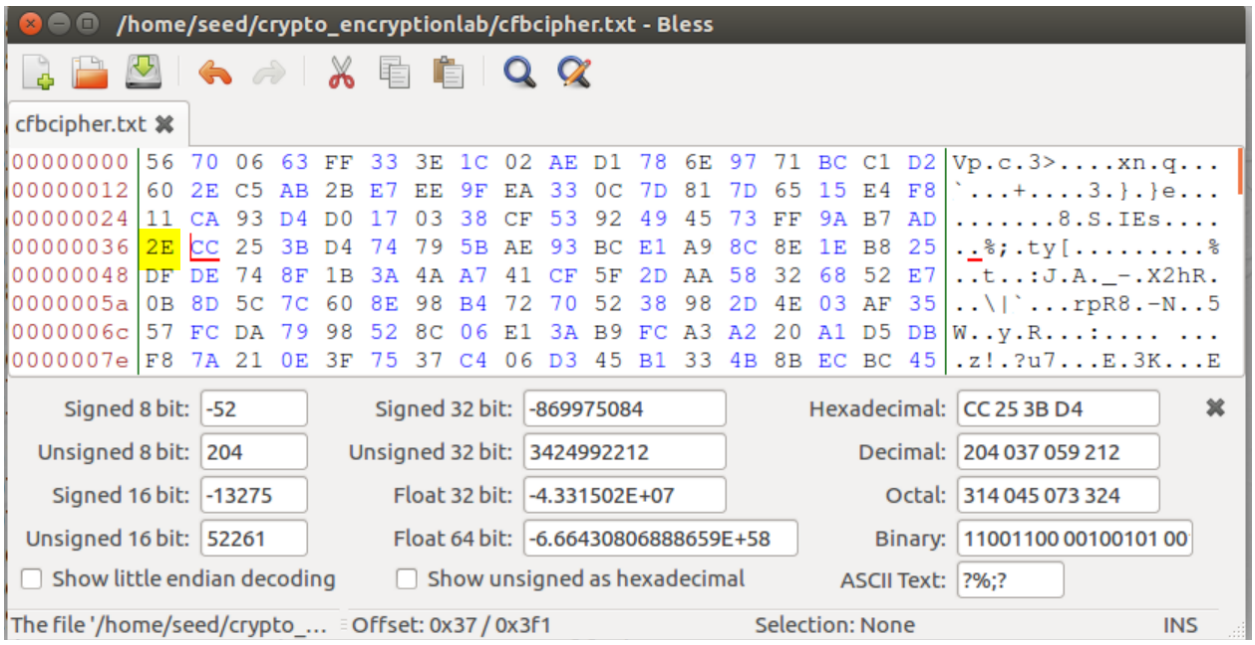

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cbc -in cbccipher.txt -out cbcplain.txt -d -K 00112233445566778899aabbccddeeff -iv 001020
30405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat cbcplain.txt
12345678901234567890123456789012345678901234567Y64j6I
6Y6567890023456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

Changing 1 bit corrupted a much larger portion of original data (more than 100 bytes)

CFB:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cfb -in plain.txt -out cfbcipher.txt -e -K 00112233445566778899aabbccddeeff -iv 001020304
05060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ xxd cfbcipher.txt
00000000: 5670 0663 ff33 3e1c 02ae d178 6e97 71bc  Vp.c.3>....xn.q.
00000010: c1d2 602e c5ab 2be7 ee9f ea33 0c7d 817d  ..`...+....3.}.}
00000020: 6515 e4f8 11ca 93d4 d017 0338 cf53 9249  e..........8.S.I
00000030: 4573 ff9a b7ad 2fcc 253b d474 795b ae93  Es..../.%;.ty[..
00000040: bce1 a98c 8e1e b825 dfde 748f 1b3a 4aa7  .......%..t..:J.
00000050: 41cf 5f2d aa58 3268 52e7 0b8d 5c7c 608e  A._-.X2hR...\|`.
00000060: 98b4 7270 5238 982d 4e03 af35 57fc da79  ..rpR8.-N..5W..y
00000070: 9852 8c06 e13a b9fc a3a2 20a1 d5db f87a  .R...:.... ....z
00000080: 210e 3f75 37c4 06d3 45b1 334b 8bec bc45  !.?u7...E.3K...E
00000090: 4e98 c489 1a64 e131 c655 7544 7b53 3ac1  N....d.1.UuD{S:.
000000a0: 8281 9a9f f63e d952 f241 a101 f491 3e9d  .....>.R.A....>.
000000b0: b040 a523 7b57 3012 5f73 e3bd 8127 fe86  .@.#{W0._s...'..
000000c0: a9af 1f15 fced 6475 ab40 dbf0 eb6f 2424  ......du.@...o$$
000000d0: 6fce 325e b165 cde9 1c02 fb45 c68d 73f7  o.2^.e.....E..s.
000000e0: a807 acd7 c27b 5362 bfde 87f1 101b 6cc4  .....{Sb......l.
```

Change the 55th byte of cfbcipher.txt 0x2F by 1 bit -> 0x2E using the bless hex editor:



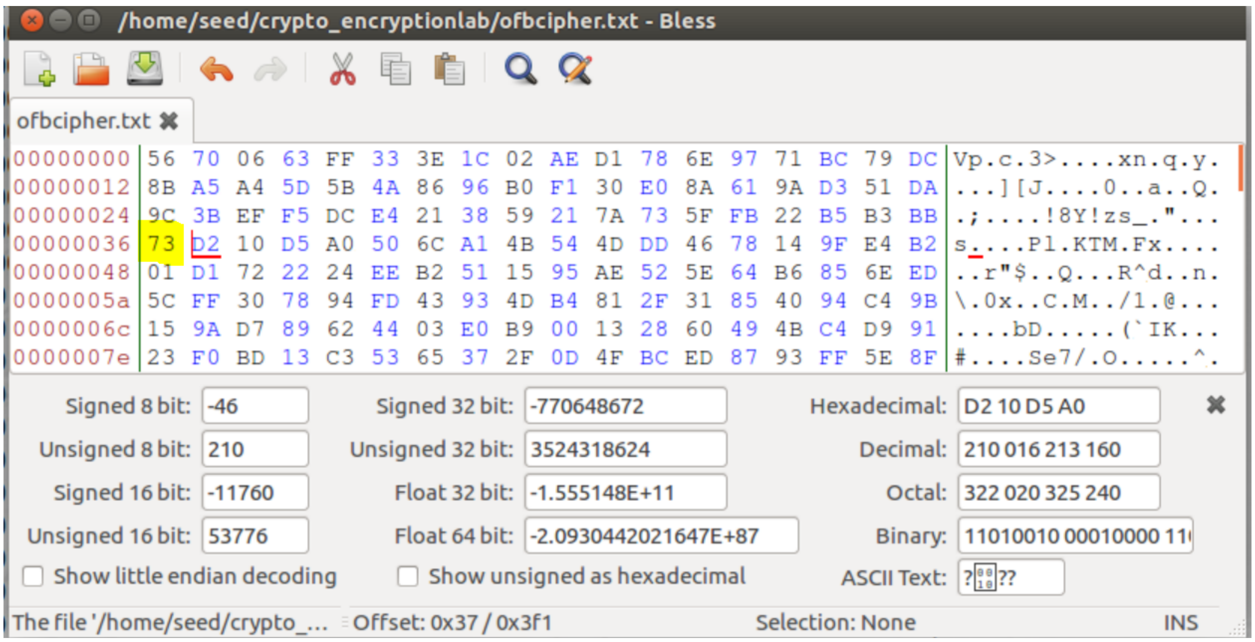

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-cfb -in cfbcipher.txt -out cfbplain.txt -d -K 00112233445566778899aabbccddeeff -iv 001020
30405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat cfbplain.txt
1234567890123456789012345678901234567890123446789012341?6?????U5K?[?? ?1234567890123456789
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

Changing 1 bit corrupted about 16 bytes of data

OFB:

```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-ofb -in plain.txt -out ofbcipher.txt -e -K 00112233445566778899aabbccddeeff -iv 001020304
05060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ xxd ofbcipher.txt
00000000: 5670 0663 ff33 3e1c 02ae d178 6e97 71bc  Vp.c.3>....xn.q.
00000010: 79dc 8ba5 a45d 5b4a 8696 b0f1 30e0 8a61  y....][J....0..a
00000020: 9ad3 51da 9c3b eff5 dce4 2138 5921 7a73  ..Q..;....!8Y!zs
00000030: 5ffb 22b5 b3bb 71d2 10d5 a050 6ca1 4b54  _."...q....Pl.KT
00000040: 4ddd 4678 149f e4b2 01d1 7222 24ee b251  M.Fx......r"$..Q
00000050: 1595 ae52 5e64 b685 6eed 5cff 3078 94fd  ...R^d..n.\.0x..
00000060: 4393 4db4 812f 3185 4094 c49b 159a d789  C.M../1.@.......
00000070: 6244 03e0 b900 1328 6049 4bc4 d991 23f0  bD.....(`IK...#.
00000080: bd13 c353 6537 2f0d 4fbc ed87 93ff 5e8f  ...Se7/.O.....^.
00000090: 1bbc b1ee cd53 bd17 118a 2d69 858a 9014  .....S....-i....
000000a0: f48f 6485 75fc 65e6 58f6 a87e a2fa 21ea  ..d.u.e.X..~..!.
000000b0: 95a4 281d e3cc e895 641d 9cb7 5586 a11c  ..(.....d...U...
000000c0: 2989 2f16 ef0e 58bf b69e 0781 6658 fc62  )./...X.....fX.b
000000d0: 8f3b a3a7 760c bfdb 9e6a 52b6 cfd6 1561  .;..v....jR....a
000000e0: 6626 1a4b b249 b785 28ad ebfb c552 5744  f&.K.I..(....RWD
```

Change the 55th byte of cfbcipher.txt 0x71 by 1 bit -> 0x73 using the bless hex editor:

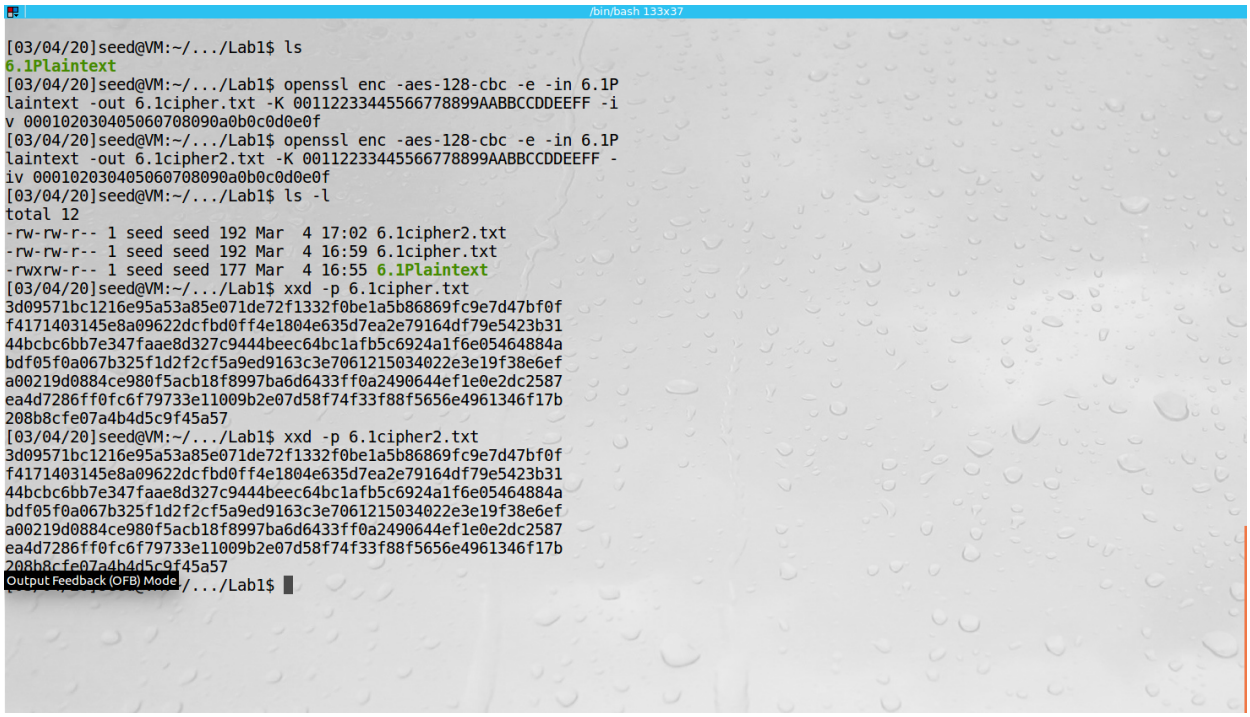



```
[03/04/20]seed@VM:~/crypto_encryptionlab$ openssl enc -aes-128-ofb -in ofbcipher.txt -out ofbplain.txt -d -K 00112233445566778899aabbccddeeff -iv 001020
30405060708090a0b0c0d0e0f0
[03/04/20]seed@VM:~/crypto_encryptionlab$ cat ofbplain.txt
1234567890123456789012345678901234567890123456789012347678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
[03/04/20]seed@VM:~/crypto_encryptionlab$
```

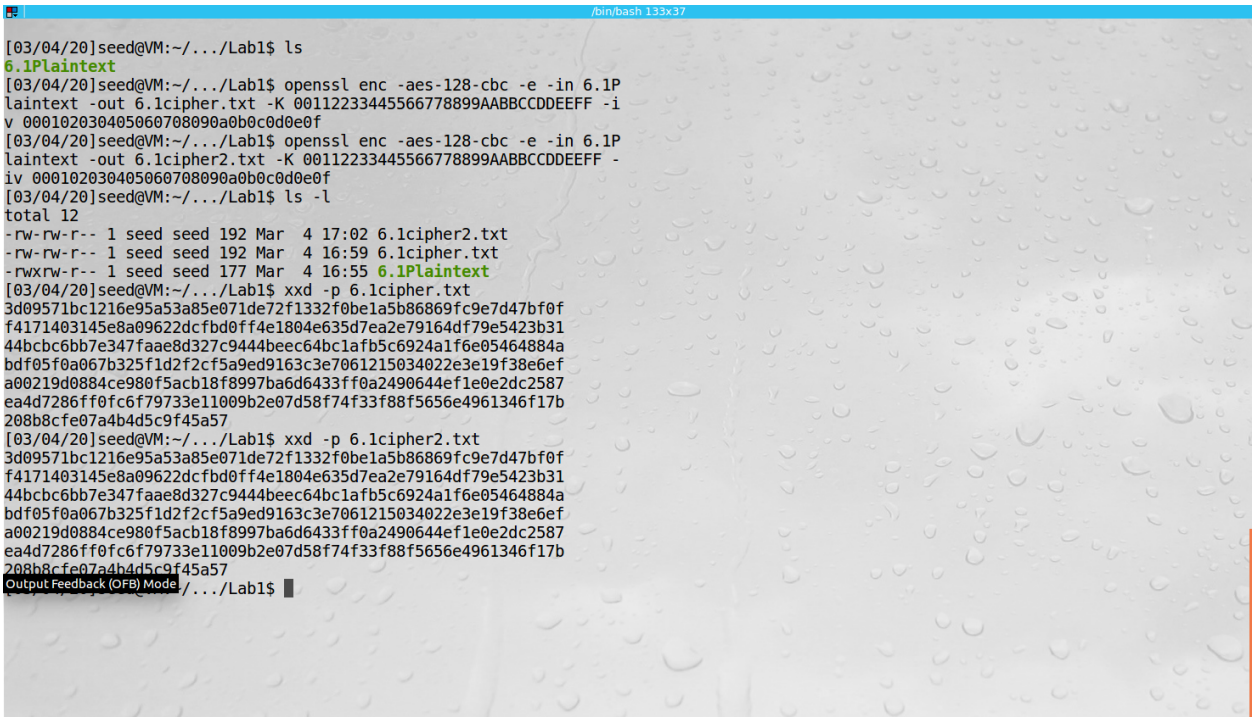
Changing 1 bit only affected a single byte of the plaintext.

2.6 Task 6: Initial Vector (IV)

Task 2.6.1: IV Uniqueness



- We can see that when we encrypted the 6.1Plaintext file twice using the same key and the same IV, the resulting encryption scheme was exactly the same, which is not what we would want. (If someone was able to break the encryption algorithm, then they would have access to both files.)



- **When we use different IVs to encrypt the same file, we can see that the encryption algorithm produces two different ciphertexts for the same file – which is what we want. Thus, when we use the same key, we must use different IVs.**

Task 2.6.2:

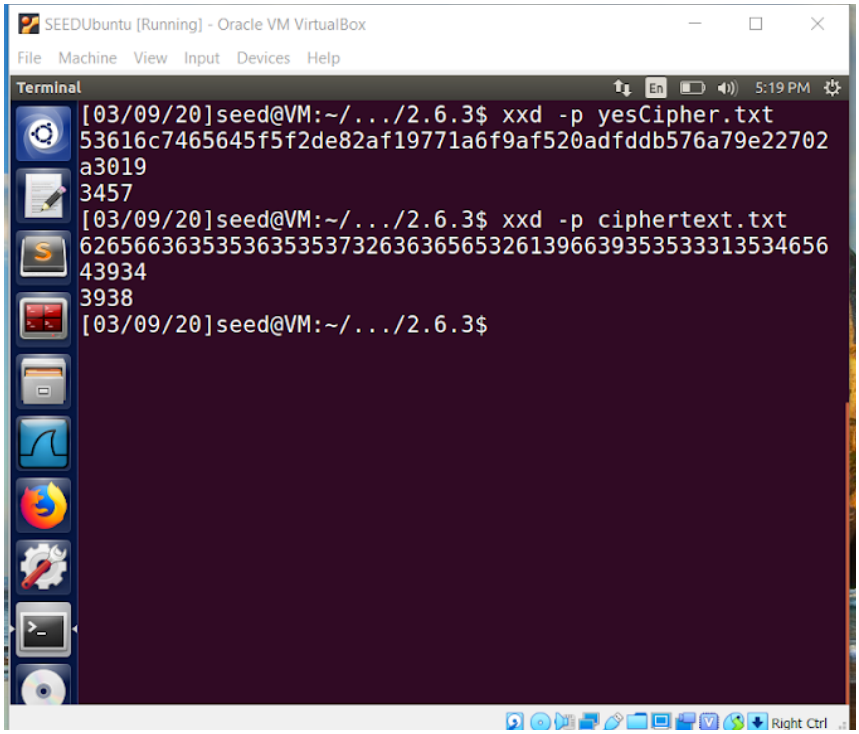
We know the encryption used is Output Feedback mode and we have a plaintext and its corresponding ciphertext. We also have a 2nd ciphertext that we need to decipher. Assuming the IV is always the same:

In Output Feedback Mode, we know that the Ciphertext xor Plaintext is the next iv for the following cipher text.

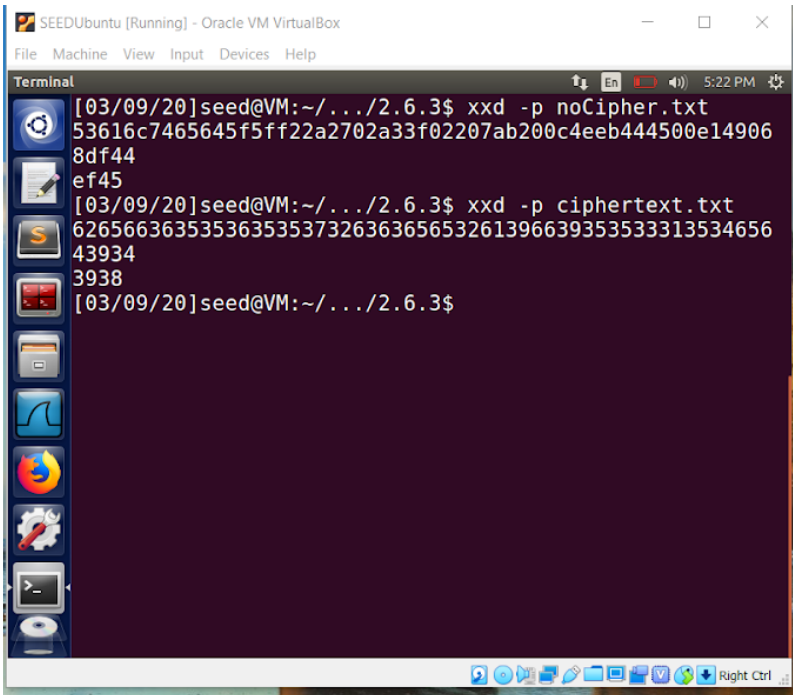
Task 2.6.3:

To guess the ciphertext, we must first XOR IV_1 and IV_2. With that value, we must XOR it with the hex value of the guessed plaintext.

Then, we must encrypt this file using -aes-128-cbc with the second IV. Finally, we can compare the hex value of the encrypted guess file with Bob's ciphertext.



The values are different, so yes is not the correct guess. Let's try to guess "No" now.



Apparently, No is not the correct either.

2.7 Task 7: Programming using the Crypto Library

Known: We are given a plaintext and a ciphertext, and our job is to find the key that is used for the encryption. We know that:

-The aes-128-cbc cipher is used for the encryption.

-The key used to encrypt this plaintext is an English word shorter than 16 characters; the word can be found from a typical English dictionary. Since the word has less than 16 characters, pound signs (0x23) are appended to the end of the word to form a key of 128 bits.

Approach: Take the list of words 1 line at a time, pad it, and use it to try and decrypt the known cipher text and compare generated plaintext to the original plaintext

```python
import binascii
from Crypto.Cipher import AES

#open filestreams with read-only access
fp = open("plain.txt", "r")
fc = open("cipher.txt", "r")
fwl = open("wordlist.txt", "r")

#declaration and initialization of known plaintext and ciphertext
plaintext = fp.read().rstrip()
ciphertext = fc.read().rstrip()

#declaration and initialization of initialization vector converted from hex to ascii
iv = binascii.unhexlify('aabbccddeeff00998877665544332211')

#iterate over each word in wordlist
for x in fwl:
    #skip empty words
    if len(x) == 0:
        continue
    #declaration and initialization of word as test for secret value converted to hex
    testkey = x.rstrip()
    testkey = binascii.hexlify(testkey)
    #append hex value 0x23 to testkey until 32 hexbits in length
    if len(testkey) > 32:
        continue
    while len(testkey) < 32:
        testkey += "23"
    #convert secret key back to ascii
    testkey = binascii.unhexlify(testkey)
    #declaration and instantiation of object from Crypto.Cipher library for AES-128-CBC encryption using known IV and our test secret key
    obj = AES.new(testkey, AES.MODE_CBC, iv)
    #decrypt ciphertext
    plain = obj.decrypt(ciphertext)
    #check for partial match
    if  'secret' in plain:
        print (plain)
        print ("ciphertext: " + testkey)
        exit()

#No plaintext match found
print("None of the decrypted plaintexts match the known plaintext")
```

In the end, no matches were found between the generated plaintext and the known plaintext.

```
[03/12/20]seed@VM:~/.../part7$ python findkey.py
None of the decrypted plaintexts match the known plaintext
[03/12/20]seed@VM:~/.../part7$
```