F21CN: COMPUTER NETWORK SECURITY

Student Declaration of Authorship

Course code and name:	F21CN: Computer Network Security
Type of assessment:	**Individual**
Coursework Title:	Symmetric Encryption
Student Name:	Sasha Fahima Suhel
Student ID Number:	H00410394

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature *(type your name):* *Sasha Fathima Suhel*

Date: *24/10/2022*

Copy this page and insert it into your coursework file in front of your title page.
For group assessment each group member must sign a separate form and all forms must be included with the group submission.

Your work will not be marked if a signed copy of this form is not included with your submission.

Assessed Coursework 1 — Symmetric Encryption

Sasha Fathima Suhel, MSc. Data Science

Heriot Watt University Dubai

H00410394

Introduction

This coursework is for the subject F21CN: Computer Network Security. The main topic for this given coursework is on Symmetric encryption. Each task given in the course work is attempted to the best of my comprehension on the given assignment. Ever task is divided into objective, methodology and then snippets/output of the respective tasks.

All tasks are clearly described below whilst referencing the sources that helped me achieve my desired output. I hope that within this assignment given I can comprehend various methods used in Symmetric Encryption and how they are implanted. Through this I hope to broaden my knowledge in cryptography especially with -AES-128 cipher and the different modes of operation whilst using appropriate key and iv.

I look forward to comprehending how frequency analysis is used in Task 1 and how it displays Monoalphabetic Substitution Ciphers as being incredibly inadequate.
I look forward to understanding in Task 2 the affect of padding during encryption and decryption in different file sizes.
I look forward to comprehending how different modes of encryption get affected when the file is corrupted in Task 3.
I hope to match two cipher files with the help of a password for Task 4

Specifications

Windows OS Version 10.0.22000, Openssl version 3.0.1, CentOS Stream release 9, VirtualBox 6.1.40

INDEX

Contents

Task 1: Frequency Analysis: Monoalphabetic Substitution Cipher

Objective

The objective of the task is to decipher the encrypted text with the help of frequency analysis. Frequency analysis is the rate at which each individual letter or a group of letters gets repeated the most in common English language. Long paragraphs or texts are required to get an accurate rate of frequency of letters which is then matched to the frequency of letters of the encrypted message to possibly decipher it. Through this process I will be attempting to decrypt the file cipher-task1-197.

Links

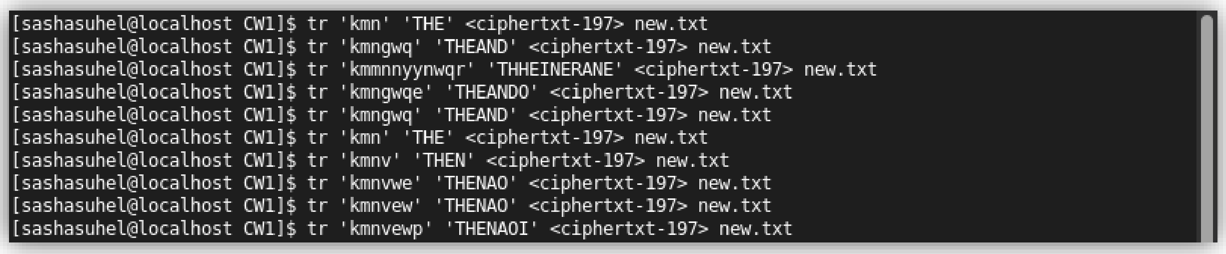
Using the first resource provided (https://onlinetoolz.net/letter-frequency) helped me obtain the frequency analysis of my cipher text. The following resources were used to compare the rate of general English frequency to that of mine cipher text in attempt to decrypt the file. (https://en.wikipedia.org/wiki/Frequency_analysis), (https://en.wikipedia.org/wiki/Bigram) (https://en.wikipedia.org/wiki/Trigram)

Methodology

Through the websites and links used above, I compared the frequency of letters and replaced them with Bigram and Trigram frequency as they seemed most efficient than single letter frequency. Initially starting with Trigram, I was able to replace the highest frequent combination of letters ‘kmn’ with ‘THE’ as it’s the most common word. This process as whilst it works can seem inefficient due to the time taken and the rate of error that could occur. The first 7 lines of code contained errors as the second highest 3 letter word ‘AND’ didn’t compute to make logical sense. I then shifted to Bigram and attempted different combinations until I replaced ‘ kmnvewp’ with ‘THENAOI’.

I then followed to interchange the alphabets mapped in Bigram as the next indexed alphabet then has the second highest chance of being the correct alphabet.

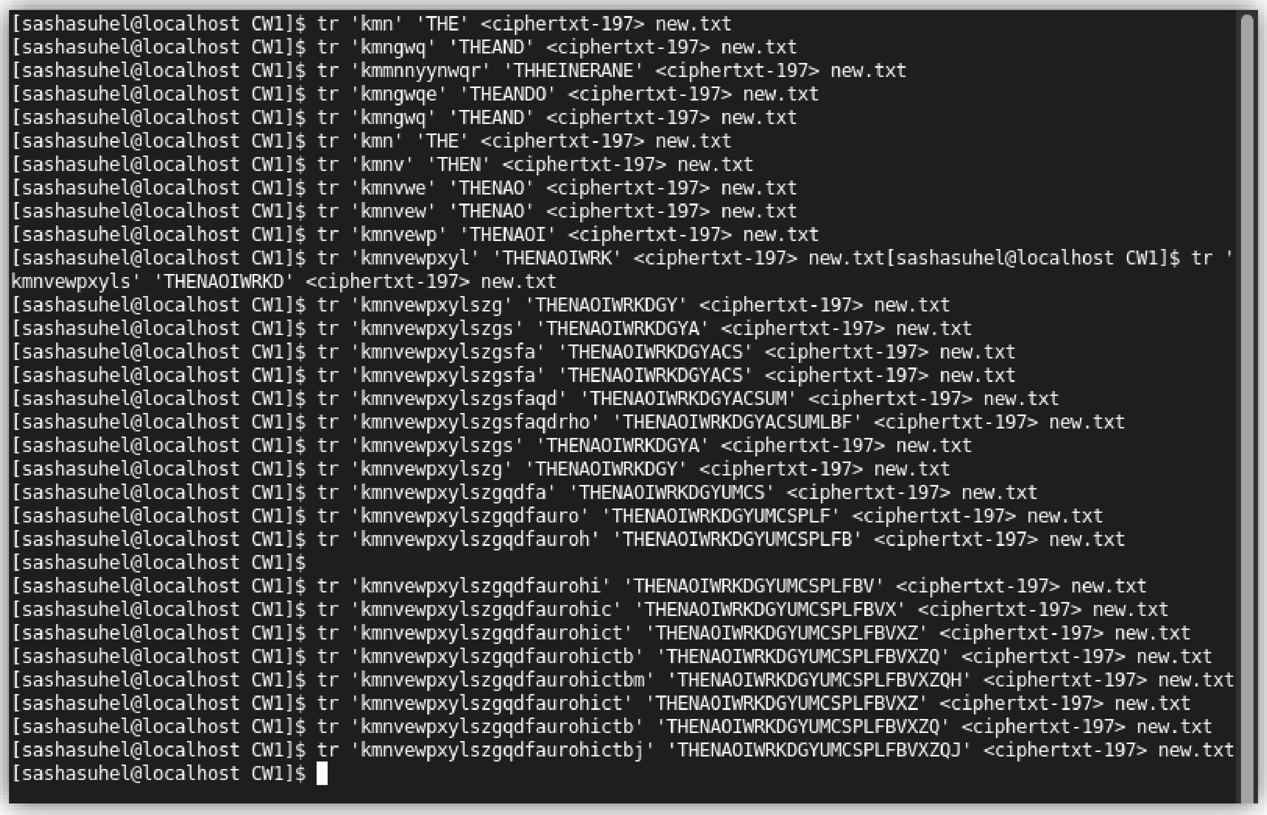
Lastly, I found the analysis works best initially with replacing up to 4 – 5 letters, as after that the frequency analysis was not as accurate since words such as Names, Places could wary country to country. Trigram and Bigram are most efficient as it covers a larger portion of words .



```
[sashasuhel@localhost CW1]$ tr 'kmn' 'THE' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmngwq' 'THEAND' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmmnnyyynwqr' 'THHEINERANE' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmngwqe' 'THEANDO' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmngwq' 'THEAND' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmn' 'THE' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnv' 'THEN' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvwe' 'THENAO' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvew' 'THENAO' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewp' 'THENAOI' <ciphertxt-197> new.txt
```

Figure 1.1

Snippet



```
[sashasuhel@localhost CW1]$ tr 'kmn' 'THE' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmngwq' 'THEAND' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmmnnyynwqr' 'THHEINERANE' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmngwqe' 'THEANDO' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmngwq' 'THEAND' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmn' 'THE' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnv' 'THEN' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvwe' 'THENAO' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvew' 'THENAO' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewp' 'THENAOI' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxyl' 'THENAOIWRK' <ciphertxt-197> new.txt[sashasuhel@localhost CW1]$ tr '
kmnvewpxyls' 'THENAOIWRKD' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszg' 'THENAOIWRKDGY' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgs' 'THENAOIWRKDGYA' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgsfa' 'THENAOIWRKDGYACS' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgsfa' 'THENAOIWRKDGYACS' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgsfaqd' 'THENAOIWRKDGYACSUM' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgsfaqdrho' 'THENAOIWRKDGYACSUMLBF' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgs' 'THENAOIWRKDGYA' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszg' 'THENAOIWRKDGY' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfa' 'THENAOIWRKDGYUMCS' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfauro' 'THENAOIWRKDGYUMCSPLF' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfauroh' 'THENAOIWRKDGYUMCSPLFB' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfaurohi' 'THENAOIWRKDGYUMCSPLFBV' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfaurohic' 'THENAOIWRKDGYUMCSPLFBVX' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfaurohict' 'THENAOIWRKDGYUMCSPLFBVXZ' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfaurohictb' 'THENAOIWRKDGYUMCSPLFBVXZQ' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfaurohictbm' 'THENAOIWRKDGYUMCSPLFBVXZQH' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfaurohict' 'THENAOIWRKDGYUMCSPLFBVXZ' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfaurohictb' 'THENAOIWRKDGYUMCSPLFBVXZQ' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ tr 'kmnvewpxylszgqdfaurohictbj' 'THENAOIWRKDGYUMCSPLFBVXZQJ' <ciphertxt-197> new.txt
[sashasuhel@localhost CW1]$ 
```

Figure 1.2

Alphabet Mapping
= SQXMACYBVJTKHEFIULDZPNOWRG (Cipher key)

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
S	Q	X	M	A	C	Y	B	V	J	T	K	H	E	F	I	U	L	D	Z	P	N	O	W	R	G

The above key was used to encrypt <ciphertext-197> which mapped every letter of the English alphabet to the key and replaced them to create a cipher text. When the key is remapped to the ciphertext, the output is the decrypted/original message (Plain.txt).

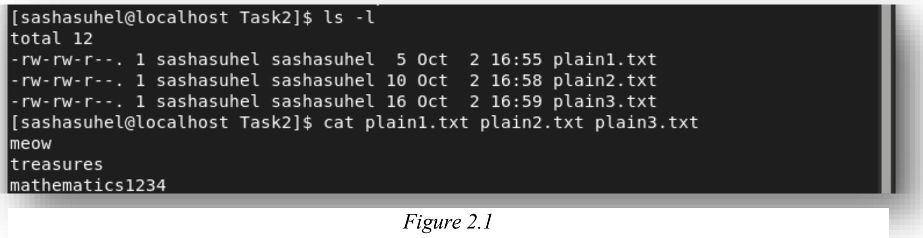
Task 2: Symmetric encryption: Padding

Objective

To explore padding during the process of encryption and be able to view the process of padding while decrypting to see the change in the byte size. Padding involves adding extra bytes to a plaintext message before encryption when the size of a plaintext message is not a multiple of the block size during block ciphers.

Methodology

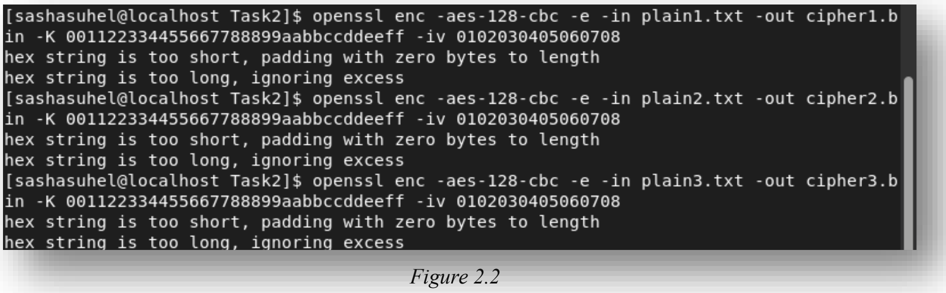
Step 1 is to create 3 files of exact 5,10 and 16 byte respectively named as shown



```
[sashasuhel@localhost Task2]$ ls -l
total 12
-rw-rw-r--. 1 sashasuhel sashasuhel  5 Oct  2 16:55 plain1.txt
-rw-rw-r--. 1 sashasuhel sashasuhel 10 Oct  2 16:58 plain2.txt
-rw-rw-r--. 1 sashasuhel sashasuhel 16 Oct  2 16:59 plain3.txt
[sashasuhel@localhost Task2]$ cat plain1.txt plain2.txt plain3.txt
meow
treasures
mathematics1234
```

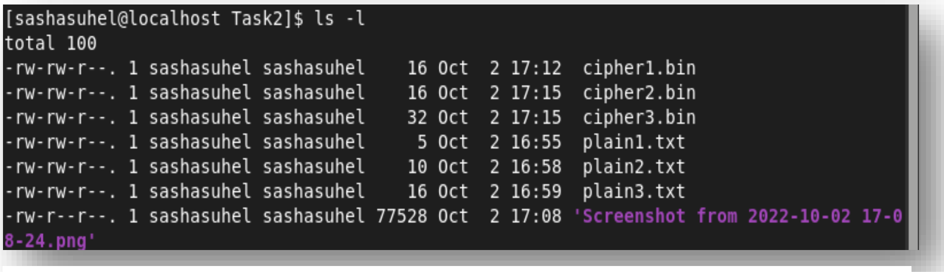
Figure 2.1

Step 2 is to encrypt each of these files to another file. The algorithm which is used to encrypt is -aes-128-cbc which is 128-bit size. While encrypting, it pads the byte size of each file to the next multiples of 128 to secure the data. For ex. 5 to 16, 10 to 16, 16 to 32.



```
[sashasuhel@localhost Task2]$ openssl enc -aes-128-cbc -e -in plain1.txt -out cipher1.b
in -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too long, ignoring excess
[sashasuhel@localhost Task2]$ openssl enc -aes-128-cbc -e -in plain2.txt -out cipher2.b
in -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too long, ignoring excess
[sashasuhel@localhost Task2]$ openssl enc -aes-128-cbc -e -in plain3.txt -out cipher3.b
in -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too long, ignoring excess
```

Figure 2.2

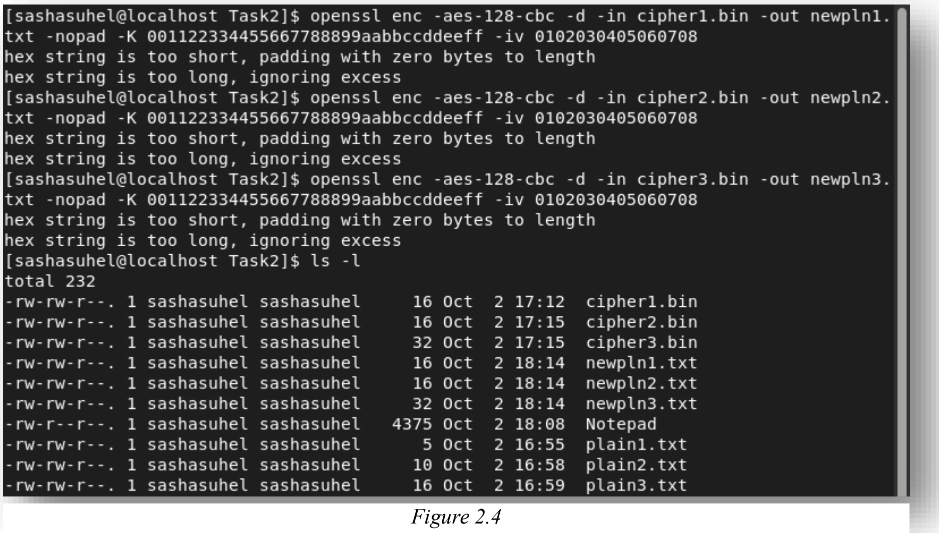


```
[sashasuhel@localhost Task2]$ ls -l
total 100
-rw-rw-r--. 1 sashasuhel sashasuhel    16 Oct  2 17:12  cipher1.bin
-rw-rw-r--. 1 sashasuhel sashasuhel    16 Oct  2 17:15  cipher2.bin
-rw-rw-r--. 1 sashasuhel sashasuhel    32 Oct  2 17:15  cipher3.bin
-rw-rw-r--. 1 sashasuhel sashasuhel     5 Oct  2 16:55  plain1.txt
-rw-rw-r--. 1 sashasuhel sashasuhel    10 Oct  2 16:58  plain2.txt
-rw-rw-r--. 1 sashasuhel sashasuhel    16 Oct  2 16:59  plain3.txt
-rw-r--r--. 1 sashasuhel sashasuhel 77528 Oct  2 17:08 'Screenshot from 2022-10-02 17-0
8-24.png'
```

Figure 2.3

As seen above when mentioned to list the details of the files, cipher1.bin and cipher2.bin went to the next byte allocation which is 16, while cipher3.bin went to 32 bytes.

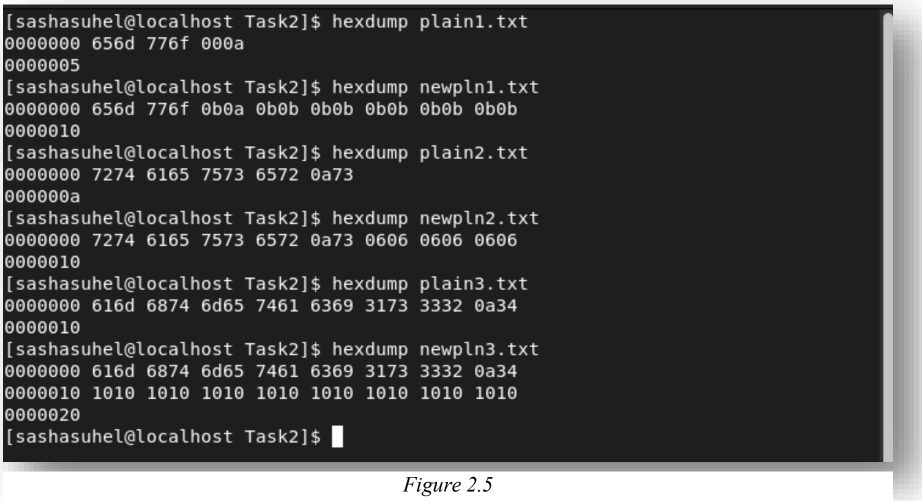
Step 3 is to convert these encrypted cipher files to a new decrypted file while keeping its padding by using the command '-nopad'



```
[sashasuhel@localhost Task2]$ openssl enc -aes-128-cbc -d -in cipher1.bin -out newpln1.
txt -nopad -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too long, ignoring excess
[sashasuhel@localhost Task2]$ openssl enc -aes-128-cbc -d -in cipher2.bin -out newpln2.
txt -nopad -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too long, ignoring excess
[sashasuhel@localhost Task2]$ openssl enc -aes-128-cbc -d -in cipher3.bin -out newpln3.
txt -nopad -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too long, ignoring excess
[sashasuhel@localhost Task2]$ ls -l
total 232
-rw-rw-r--. 1 sashasuhel sashasuhel     16 Oct  2 17:12  cipher1.bin
-rw-rw-r--. 1 sashasuhel sashasuhel     16 Oct  2 17:15  cipher2.bin
-rw-rw-r--. 1 sashasuhel sashasuhel     32 Oct  2 17:15  cipher3.bin
-rw-rw-r--. 1 sashasuhel sashasuhel     16 Oct  2 18:14  newpln1.txt
-rw-rw-r--. 1 sashasuhel sashasuhel     16 Oct  2 18:14  newpln2.txt
-rw-rw-r--. 1 sashasuhel sashasuhel     32 Oct  2 18:14  newpln3.txt
-rw-r--r--. 1 sashasuhel sashasuhel   4375 Oct  2 18:08  Notepad
-rw-rw-r--. 1 sashasuhel sashasuhel      5 Oct  2 16:55  plain1.txt
-rw-rw-r--. 1 sashasuhel sashasuhel     10 Oct  2 16:58  plain2.txt
-rw-rw-r--. 1 sashasuhel sashasuhel     16 Oct  2 16:59  plain3.txt
```

Figure 2.4

Outlook



```
[sashasuhel@localhost Task2]$ hexdump plain1.txt
0000000 656d 776f 000a
0000005
[sashasuhel@localhost Task2]$ hexdump newpln1.txt
0000000 656d 776f 0b0a 0b0b 0b0b 0b0b 0b0b 0b0b
0000010
[sashasuhel@localhost Task2]$ hexdump plain2.txt
0000000 7274 6165 7573 6572 0a73
000000a
[sashasuhel@localhost Task2]$ hexdump newpln2.txt
0000000 7274 6165 7573 6572 0a73 0606 0606 0606
0000010
[sashasuhel@localhost Task2]$ hexdump plain3.txt
0000000 616d 6874 6d65 7461 6369 3173 3332 0a34
0000010
[sashasuhel@localhost Task2]$ hexdump newpln3.txt
0000000 616d 6874 6d65 7461 6369 3173 3332 0a34
0000010 1010 1010 1010 1010 1010 1010 1010 1010
0000020
[sashasuhel@localhost Task2]$ █
```

Figure 2.5

Using the ‘hexdump’ function it is clearly visible on the comparison between the plaintext file and the decrypted newpln that there are additional bytes added.

Task 3: Encryption Mode — Corrupted Cipher Text

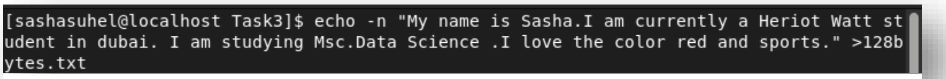
Objective

To understand how a file would change if the encrypted file got corrupted while through different modes of encryption ECB, CBC, CFB, and OFB, while using a general key and iv for the following. A standard AES-192 cipher is used throughout.

Methodology

The resource (https://www.rapidtables.com/convert/number/decimal-to-hex.html) helped me ensure that for the first part of corruption that I changed only a single bit of the 46th position of the hexadecimal value of the encrypted file. All files that were corrupted decrypted, but with varying results as every mode of operation uses a different mode of matrix to encrypt so as a result during decryption, we can further understand how every mode of encryption works. Some modes encrypt independently, while others rely on previous block of bytes.

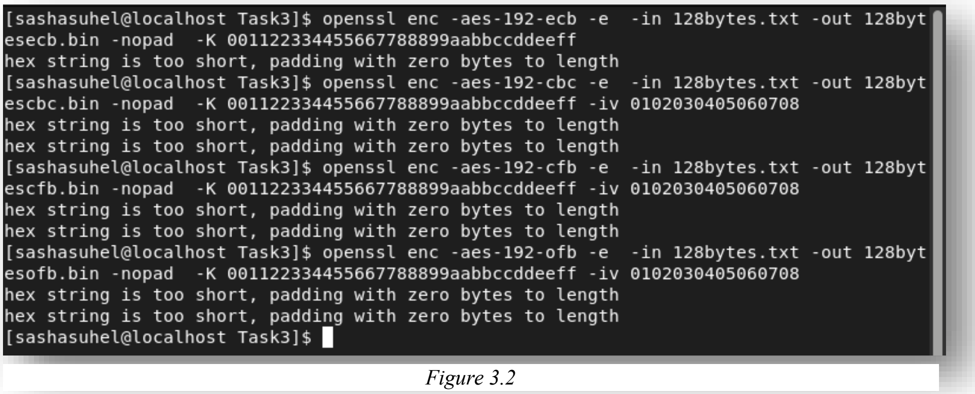
Step 1: To create a file 128bytes with exactly 128bytes which will be used throughout.



```
[sashasuhel@localhost Task3]$ echo -n "My name is Sasha.I am currently a Heriot Watt st
udent in dubai. I am studying Msc.Data Science .I love the color red and sports." >128b
ytes.txt
```

Figure 3.1

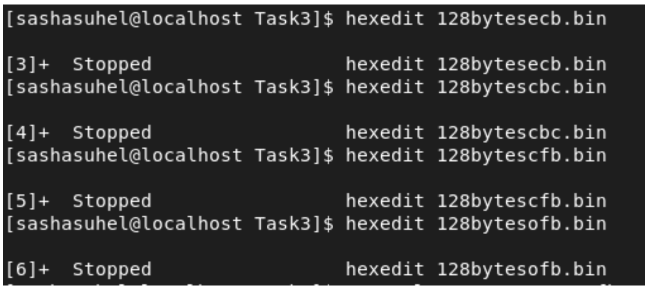
Step 2: Is to encrypt the file ‘128bytes’ with different modes of encryption . The command -nopad is used here so no additional padding is done while encrypting the files.



```
[sashasuhel@localhost Task3]$ openssl enc -aes-192-ecb -e  -in 128bytes.txt -out 128byt
esecb.bin -nopad  -K 00112233445566778899aabbccddeeff
hex string is too short, padding with zero bytes to length
[sashasuhel@localhost Task3]$ openssl enc -aes-192-cbc -e  -in 128bytes.txt -out 128byt
escbc.bin -nopad  -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[sashasuhel@localhost Task3]$ openssl enc -aes-192-cfb -e  -in 128bytes.txt -out 128byt
escfb.bin -nopad  -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[sashasuhel@localhost Task3]$ openssl enc -aes-192-ofb -e  -in 128bytes.txt -out 128byt
esofb.bin -nopad  -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[sashasuhel@localhost Task3]$ █
```

Figure 3.2

Step 3: Once encryption is done, next I have installed the library **hexedit** on my terminal using the command , *"yum install hexedit"* to help me corrupt the files and modify the hexadecimal values for each of the ciphertext files. Now let's view the process for the different modes on what happens when I modify a single bit in the 46th byte and remove the block containing the 86th byte from each of the ciphertext files.

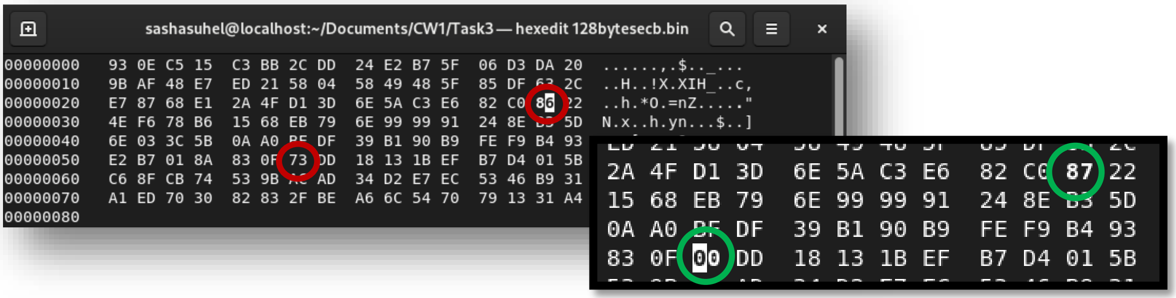


```
[sashasuhel@localhost Task3]$ hexedit 128bytesecb.bin

[3]+  Stopped                 hexedit 128bytesecb.bin
[sashasuhel@localhost Task3]$ hexedit 128bytescbc.bin

[4]+  Stopped                 hexedit 128bytescbc.bin
[sashasuhel@localhost Task3]$ hexedit 128bytescfb.bin

[5]+  Stopped                 hexedit 128bytescfb.bin
[sashasuhel@localhost Task3]$ hexedit 128bytesofb.bin

[6]+  Stopped                 hexedit 128bytesofb.bin
```

Figure 3.3

ECB (Electronic Codebook)

The binary value for 86(decimal) is 1010110, so after changing one bit the binary value will be 1010111 (87). To remove the block containing the 86^{th} byte I have modified it to 00



sashasuhel@localhost:~/Documents/CW1/Task3 — hexedit 128bytesecb.bin

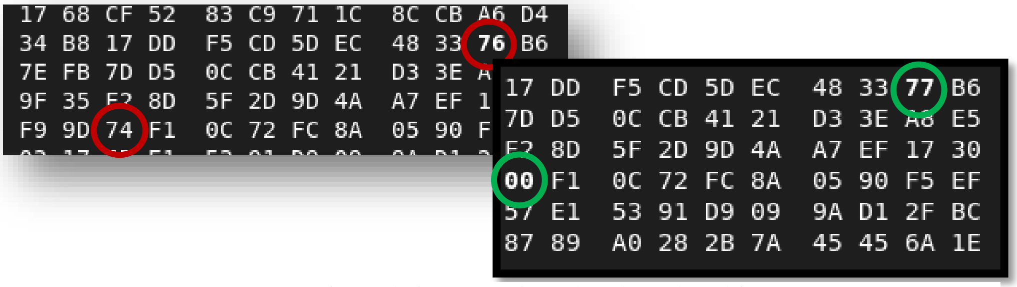
```
00000000   93 0E C5 15  C3 BB 2C DD  24 E2 B7 5F  06 D3 DA 20  ......,.$.._...
00000010   9B AF 48 E7  ED 21 58 04  58 49 48 5F  85 DF 63 2C  ..H..!X.XIH_..c,
00000020   E7 87 68 E1  2A 4F D1 3D  6E 5A C3 E6  82 C0 86 22  ..h.*O.=nZ....."
00000030   4E F6 78 B6  15 68 EB 79  6E 99 99 91  24 8E B3 5D  N.x..h.yn...$..]
00000040   6E 03 3C 5B  0A A0 BE DF  39 B1 90 B9  FE F9 B4 93
00000050   E2 B7 01 8A  83 0F 73 DD  18 13 1B EF  B7 D4 01 5B
00000060   C6 8F CB 74  53 9B A6 AD  34 D2 E7 EC  53 46 B9 31
00000070   A1 ED 70 30  82 83 2F BE  A6 6C 54 70  79 13 31 A4
00000080
```

```
2A 4F D1 3D  6E 5A C3 E6  82 C0 87 22
15 68 EB 79  6E 99 99 91  24 8E B3 5D
0A A0 BE DF  39 B1 90 B9  FE F9 B4 93
83 0F 00 DD  18 13 1B EF  B7 D4 01 5B
```

Figure 3.4 Left Image: Before Corruption, Right Image: After corruption

CBC (Cipher Block Chaining)

The binary value for 76(decimal) is 1001100, so after changing one bit the binary value will be 1001101 (77). To remove the block containing the 86^{th} byte I have modified it to 00



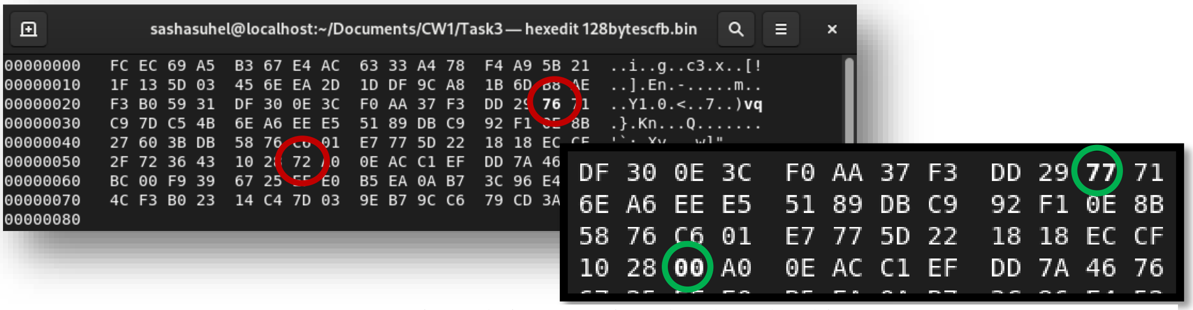
```
17 68 CF 52  83 C9 71 1C  8C CB A6 D4
34 B8 17 DD  F5 CD 5D EC  48 33 76 B6
7E FB 7D D5  0C CB 41 21  D3 3E A
9F 35 F2 8D  5F 2D 9D 4A  A7 EF 1
F9 9D 74 F1  0C 72 FC 8A  05 90 F
```

```
17 DD  F5 CD 5D EC  48 33 77 B6
7D D5  0C CB 41 21  D3 3E A8 E5
F2 8D  5F 2D 9D 4A  A7 EF 17 30
00 F1  0C 72 FC 8A  05 90 F5 EF
57 E1  53 91 D9 09  9A D1 2F BC
87 89  A0 28 2B 7A  45 45 6A 1E
```

Figure 3.5 Before and after image of CBC hexadecimal modification

CFB (Cipher Feedback)

The binary value for 76(decimal) is 1001100, so after changing one bit the binary value will be 1001101 (77). To remove the block containing the 86^{th} byte I have modified it to 00



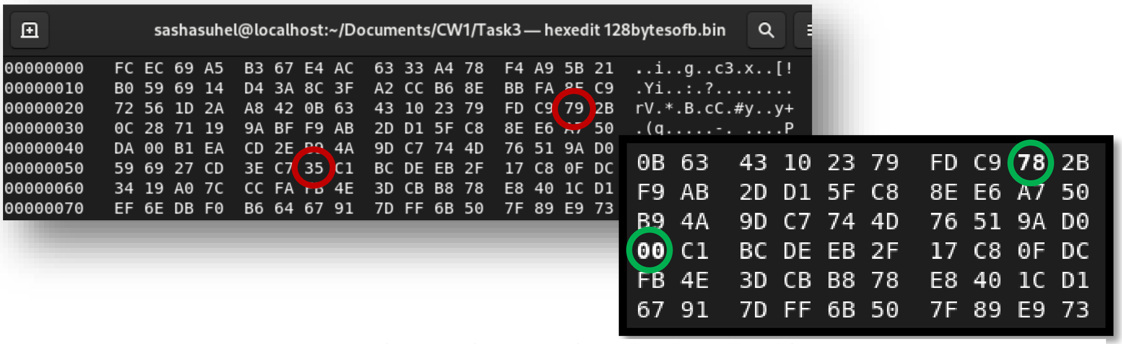
sashasuhel@localhost:~/Documents/CW1/Task3 — hexedit 128bytescfb.bin

```
00000000   FC EC 69 A5  B3 67 E4 AC  63 33 A4 78  F4 A9 5B 21  ..i..g..c3.x..[!
00000010   1F 13 5D 03  45 6E EA 2D  1D DF 9C A8  1B 6D B8 AE  ..].En.-.....m..
00000020   F3 B0 59 31  DF 30 0E 3C  F0 AA 37 F3  DD 29 76 71  ..Y1.0.<..7..)vq
00000030   C9 7D C5 4B  6E A6 EE E5  51 89 DB C9  92 F1 0E 8B  .}.Kn...Q.......
00000040   27 60 3B DB  58 76 C6 01  E7 77 5D 22  18 18 EC CF
00000050   2F 72 36 43  10 28 72 A0  0E AC C1 EF  DD 7A 46
00000060   BC 00 F9 39  67 25 FE E0  B5 EA 0A B7  3C 96 E4
00000070   4C F3 B0 23  14 C4 7D 03  9E B7 9C C6  79 CD 3A
00000080
```

```
DF 30 0E 3C  F0 AA 37 F3  DD 29 77 71
6E A6 EE E5  51 89 DB C9  92 F1 0E 8B
58 76 C6 01  E7 77 5D 22  18 18 EC CF
10 28 00 A0  0E AC C1 EF  DD 7A 46 76
```

Figure 3.6 Before and after image of CFB hexadecimal modification

OFB (Output Feedback)
The binary value for 79(decimal) is 1001111, so after changing one bit the binary value will be 1001100 (78). To remove the block containing the 86th byte I have modified it to 00



sashasuhel@localhost:~/Documents/CW1/Task3 — hexedit 128bytesofb.bin

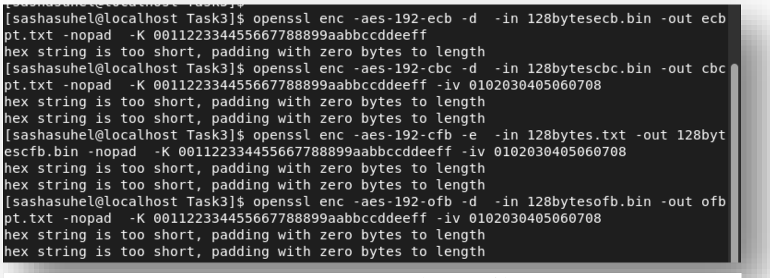
```
00000000   FC EC 69 A5  B3 67 E4 AC  63 33 A4 78  F4 A9 5B 21  ..i..g..c3.x..[!
00000010   B0 59 69 14  D4 3A 8C 3F  A2 CC B6 8E  BB FA 8E C9  .Yi..:.?........
00000020   72 56 1D 2A  A8 42 0B 63  43 10 23 79  FD C9 79 2B  rV.*.B.cC.#y..y+
00000030   0C 28 71 19  9A BF F9 AB  2D D1 5F C8  8E E6 A7 50  .(q.....-.  ....P
00000040   DA 00 B1 EA  CD 2E B9 4A  9D C7 74 4D  76 51 9A D0
00000050   59 69 27 CD  3E C7 35 C1  BC DE EB 2F  17 C8 0F DC
00000060   34 19 A0 7C  CC FA FB 4E  3D CB B8 78  E8 40 1C D1
00000070   EF 6E DB F0  B6 64 67 91  7D FF 6B 50  7F 89 E9 73
```

```
0B 63  43 10 23 79  FD C9 78 2B
F9 AB  2D D1 5F C8  8E E6 A7 50
B9 4A  9D C7 74 4D  76 51 9A D0
00 C1  BC DE EB 2F  17 C8 0F DC
FB 4E  3D CB B8 78  E8 40 1C D1
67 91  7D FF 6B 50  7F 89 E9 73
```

Figure 3.7 Before and after image of OFB hexadecimal modification

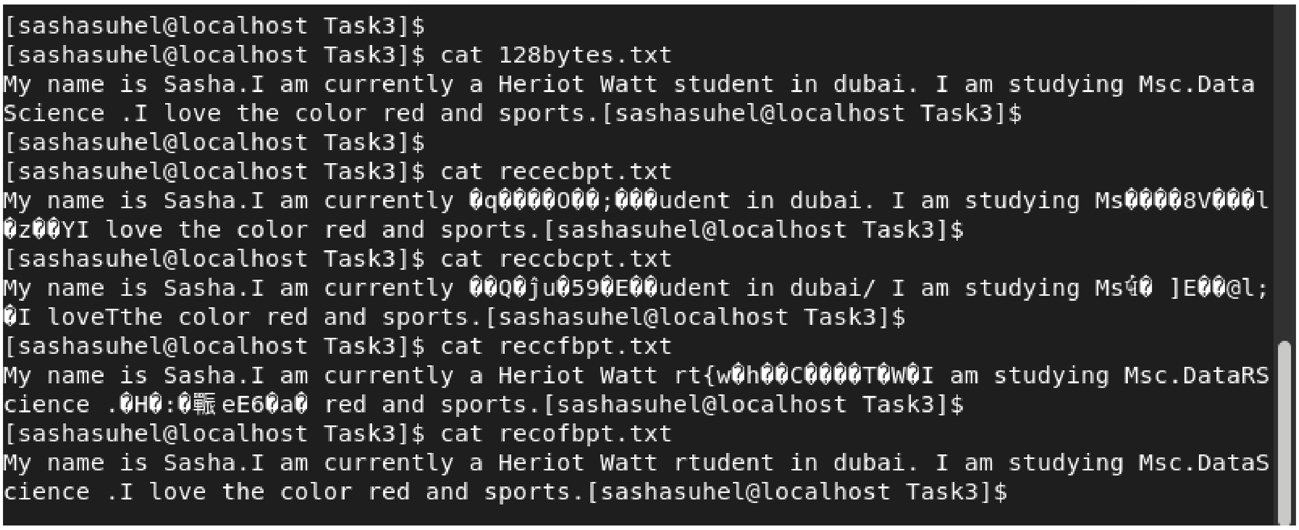
Hypothesis

Once the modifications are done, the respective files are decrypted with the appropriate key and iv along with -nopad.



```
[sashasuhel@localhost Task3]$ openssl enc -aes-192-ecb -d  -in 128bytesecb.bin -out ecb
pt.txt -nopad  -K 00112233445566778899aabbccddeeff
hex string is too short, padding with zero bytes to length
[sashasuhel@localhost Task3]$ openssl enc -aes-192-cbc -d  -in 128bytescbc.bin -out cbc
pt.txt -nopad  -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[sashasuhel@localhost Task3]$ openssl enc -aes-192-cfb -e  -in 128bytes.txt -out 128byt
escfb.bin -nopad  -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[sashasuhel@localhost Task3]$ openssl enc -aes-192-ofb -d  -in 128bytesofb.bin -out ofb
pt.txt -nopad  -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
```

Figure 3.8 Decrypt cipher text files



```
[sashasuhel@localhost Task3]$
[sashasuhel@localhost Task3]$ cat 128bytes.txt
My name is Sasha.I am currently a Heriot Watt student in dubai. I am studying Msc.Data
Science .I love the color red and sports.[sashasuhel@localhost Task3]$
[sashasuhel@localhost Task3]$
[sashasuhel@localhost Task3]$ cat rececbpt.txt
My name is Sasha.I am currently �q����O��;���udent in dubai. I am studying Ms����8V���l
�z��YI love the color red and sports.[sashasuhel@localhost Task3]$
[sashasuhel@localhost Task3]$ cat reccbcpt.txt
My name is Sasha.I am currently ��Q�ju�59�E��udent in dubai/ I am studying Msά� ]E��@l;
�I loveTthe color red and sports.[sashasuhel@localhost Task3]$
[sashasuhel@localhost Task3]$ cat reccfbpt.txt
My name is Sasha.I am currently a Heriot Watt rt{w�h��C����T�W�I am studying Msc.DataRS
cience .�H�:�甗eE6�a� red and sports.[sashasuhel@localhost Task3]$
[sashasuhel@localhost Task3]$ cat recofbpt.txt
My name is Sasha.I am currently a Heriot Watt rtudent in dubai. I am studying Msc.DataS
cience .I love the color red and sports.[sashasuhel@localhost Task3]$
```

Figure 3.9 Corrupted ciphertext result

1. ECB => $c_j = E_k(x_j)$

From the output I can understand that all the blocks in which the corrupted bit is held is recoverable. So, every block of plaintext is independently encrypted as a result there is no direct connection between the blocks of encrypted plaintext. Only the corrupted block is irrecoverable.

2. CBC => $c_0 : c_j = E_k(x_j \text{ XOR } c_{j-1})$

From the output I have understood that all the blocks that are corrupted and the remaining blocks successive to it is irrecoverable as well.

In CBC each cipher text block uses the previous cipher text blocks during encryption and for decryption. That's why in the output you can see the following texts after the corrupted block have been corrupted as well.

3. CFB => $C_j = E_k(c_{j-1})$ XOR x_j

For CFB encryption I have understood that the problem CFB face is similar to CBC. Due to the similar formula, we can understand here that all the blocks after the corrupted block will be irrecoverable.

4. OFB => $c_j = O_j$ XOR x_j

$O_j = E_j(O_{j-1})$

I have understood in OFB that only the single bit that is corrupted is irrecoverable. The encryption is not directly applied to the plain text but rather applied to the vector, which means that only one bit will be affected during decrypting as seen above, leaving rest of the blocks recoverable.

Task 4: Ciphertext-only cryptanalysis, with and without padding

Objective

The objective of this task is to utilize a dictionary file and one word from the dictionary file is encrypted using -aes-128 using CBC encryption. No such key or iv is implemented but rather a password(word from the dictionary with a number [0-9] appended) is used to match the words from two files

Methodology

First step is to encrypt the dictionary file with aes-128

- openssl enc -aes-128-cbc -d -in plain.txt -out cipher.txt -pass pass: sea5

Once encryption is done then we need to create a bash file that runs all the dictionary(encrypted) words with the cipher.txt file.

My logic to tackle this task is to create a bash file that duplicates all the words from the dictionary to another file that contains all the words from the dictionary appended with a number

For example :small.txt file contains (sea, treasure,etc). And the new file ‘duplicate.txt’

That would look like (sea1,sea2,sea3…………..treasure1,treasure2,…..etc)

Duplicate file is created to contain all the possible passwords to check with. Once this is done then the bash file should be able to match back the password with the duplicate file it should match up the word.

References

- https://command-not-found.com/hexedit

Appendices

Appendix A. Monoalphabetic Substitution Cipher (Plain.txt)

COMPUTER THAT CONTROLS
THE MFE NET
YOU MEAN THE HACKER IS ENTERING OUR LAB THROUGH THE MFE NETWORK
YEAH HES COMING FROM LAWRENCE LIVERMORE LABORATORY THE MAGNETIC FUSION
ENERGY NETWORK
I CALLED DOWN THE HALLWAY HEY DAVE GUESS WHOS VISITING LIVERMORE
DAVE AMBLED OVER TO WAYNES OFFICE HOWD HE GET THERE THERES NO CONNECTION
FROM THERE INTO OUR UNIX SYSTEM
I DONT KNOW HOW HE GOT INTO LIVERMORE BUT HES IN OUR ETHERNET COMING FROM
LIVERMORE
DAVE RAISED HIS EYEBROWS I DIDNT KNOW YOU COULD DO THAT YOUR HACKER FOUND A
PATH TO THE UNIX SYSTEM THAT I DIDNT KNOW ABOUT
WAYNE LAUNCHED INTO DAVE WITH HIS USUAL TIRADE AGAINST UNIX I LEFT THE TWO
BOSOM ENEMIES AND CALLED LIVERMORE
IT TOOK THREE CALLS TO FIND THE SYSTEM MANAGER OF THE MFE NETWORK HI YOU
DONT KNOW ME BUT YOUVE GOT A HACKER IN YOUR SYSTEM
A WOMAN ANSWERED HUH WHO ARE YOU
I WORK AT LBL SOMEONES MESSING AROUND IN MY COMPUTER AND HES COMING IN FROM
THE MFE NETWORK IT LOOKS LIKE HES LOGGED IN FROM LIVERMORE
OH HELL ILL SCAN OUR USERS   THERES ONLY ONE JOB THATS CONNECTED

FROM LIVERMORE TO BERKELEY ACCOUNT   IT BELONGS TO SOMEONE NAMED CROMWELL
THATS HIM I SAID THE HACKER FOUND THE PASSWORD A COUPLE HOURS AGO GOT
THE PASSWORD FROM A COMMAND FILE HERE IN BERKELEY
ILL KILL THAT ACCOUNT CROMWELL CAN USE OUR SYSTEM WHEN HE LEARNS TO KEEP
HIS PASSWORDS SECRET SHE SAW THE PROBLEM AS IGNORANT USERS NOT UNFRIENDLY
SYSTEMS THAT FORCED PEOPLE TO USE BIZARRE PASSWORDS LIKE AGNITFOM
CAN YOU TRACE THE CONNECTION I WANTED LIVERMORE TO KEEP THE HACKER ON LINE
AT LEAST LONG ENOUGH TO TRACE THE LINE
NO WERE NOT AUTHORIZED TO MAKE ANY TRACES YOULL HAVE TO TALK TO OUR
MANAGEMENT FIRST
BUT BY THE TIME ANYONE DECIDES THE HACKER WILL BE GONE
WE RUN A SECURE INSTALLATION SHE SAID IF ANYONE FINDS OUT THERES A HACKER
AT LIVERMORE HEADS WILL ROLL
UNLESS YOU TRACE WHERE THE HACKERS COMING FROM YOULL NEVER KNOW IF HES OUT
OF YOUR SYSTEM

MY JOB IS TO RUN A COMPUTER NOT TO CATCH CRIMINALS LEAVE ME OUT OF YOUR WILDGOOSE CHASE

SHE DECIDED TO CHOP OFF ALL ACCESS AND DISABLE THE STOLEN ACCOUNT THE HACKER DISAPPEARED FROM LIVERMORES COMPUTER AND FROM OURS

MAYBE IT WAS JUST AS WELL EVEN IF SHE HAD STARTED A TRACE I COULDNT MONITOR WHAT THE HACKER WAS DOING I COULD DETECT THAT HE WAS IN MY COMPUTER ALL RIGHT BUT THE MFE NETWORK CONNECTED DIRECTLY INTO MY COMPUTER WITHOUT GOING THROUGH THE SWITCHYARD MY PRINTERS WOULDNT CAPTURE WHAT THE HACKER TYPED

DEPRESSED I SHUFFLED TO LUNCH AT THE LBL CAFETERIA LUIS ALVAREZ SAT DOWN ACROSS FROM ME INVENTOR PHYSICIST AND NOBEL LAUREATE LUIE WAS THE TWENTIETH CENTURY RENAISSANCE MAN HE DIDNT WASTE TIME ON BUREAUCRACY HE DEMANDED RESULTS

HOWS ASTRONOMY EVEN FROM THE STRATOSPHERE ALVAREZ STILL FOUND TIME TO TALK TO PIPSQUEAKS LIKE ME STILL BUILDING THAT TELESCOPE

NAW IM WORKING AT THE COMPUTER CENTER NOW I OUGHT TO BE WRITING PROGRAMS BUT IVE BEEN SPENDING ALL MY TIME CHASING A HACKER

ANY LUCK

ITS PLAYING HIDEANDSEEK OVER THE WIRES FIRST I THINK HES COMING FROM BERKELEY THEN OAKLAND THEN ALABAMA THEN VIRGINIA LATELY IVE TRACED HIM TO LIVERMORE

CALLED THE FBI

SIX TIMES THEYVE GOT BETTER THINGS TO DO THE FRUSTRATING PART IS THE COMPLETE LACK OF SUPPORT I TOLD HIM ABOUT THE MORNINGS ACTIVITY AT LIVERMORE

YES THEYVE GOT THEIR JOBS TO WORRY ABOUT

BUT IM TRYING TO HELP THEM DARN IT THEY DONT CARE THAT THEIR NEIGHBORS BEING BURGLARIZED

STOP ACTING LIKE A CRUSADER CLIFF WHY DONT YOU LOOK AT THIS AS RESEARCH NOBODY ELSE IS INTERESTED NOT LIVERMORE NOT THE FBI HELL IN A WEEK OR TWO PROBABLY NOT EVEN OUR LABS ADMINISTRATION

THEY GAVE ME THREE WEEKS ITS ALREADY UP

THATS WHAT I MEAN WHEN YOURE DOING REAL RESEARCH YOU NEVER KNOW WHAT ITLL COST HOW MUCH TIME ITLL TAKE OR WHAT YOULL FIND YOU JUST KNOW THERES UNEXPLORED TERRITORY AND A CHANCE TO DISCOVER WHATS OUT THERE

THATS EASY FOR YOU TO SAY BUT IVE GOT TO KEEP THREE MANAGERS OFF MY BACK

THERE ARE PROGRAMS TO WRITE AND SYSTEMS TO MANAGE

SO WHAT YOURE FOLLOWING A FASCINATING SCENT YOURE AN EXPLORER THINK OF

WHO MIGHT BE BEHIND IT SOME INTERNATIONAL SPY PERHAPS
MORE LIKELY SOME BORED HIGH SCHOOL KID
WELL THEN FORGET WHOS CAUSING THE PROBLEMS LUIE SAID DONT TRY TO BE A
COP BE A SCIENTIST RESEARCH THE CONNECTIONS THE TECHNIQUES THE HOLES APPLY
PHYSICAL PRINCIPLES FIND NEW METHODS TO SOLVE PROBLEMS COMPILE STATISTICS
PUBLISH YOUR RESULTS AND ONLY TRUST WHAT YOU CAN PROVE BUT DONT EXCLUDE
IMPROBABLE SOLUTIONS KEEP YOUR MIND OPEN
BUT WHAT DO I DO WHEN I HIT A BRICK WALL
LIKE LIVERMORES SYSTEM MANAGER ASKED LUIE
OR THE TELEPHONE COMPANY WITHHOLDING A PHONE TRACE OR THE FBI REFUSING A
COURT ORDER OR OUR LABORATORY SHUTTING ME DOWN IN A COUPLE DAYS
DEAD ENDS ARE ILLUSORY WHEN DID YOU EVER LET A DO NOT ENTER SIGN KEEP YOU
AWAY FROM ANYTHING GO AROUND THE BRICK WALLS WHEN YOU CANT GO AROUND CLIMB OVER
OR DIG UNDER JUST DONT GIVE UP
BUT WHOS GOING TO PAY MY SALARY
PERMISSION BAH FUNDING FORGET IT NOBODY WILL PAY FOR RESEARCH THEYRE
ONLY INTERESTED IN RESULTS LUIE SAID SURE YOU COULD WRITE A DETAILED PROPOSAL
TO CHASE THIS HACKER IN FIFTY PAGES YOULL DESCRIBE WHAT YOU KNEW WHAT YOU
EXPECTED HOW MUCH MONEY IT WOULD TAKE INCLUDE THE NAMES OF THREE QUALIFIED
REFEREES COST BENEFIT RATIOS AND WHAT PAPERS YOUVE WRITTEN BEFORE OH AND DONT
FORGET THE THEO