Ben Heinze Lab 8 : Encryption

Task 1: Break Substitution cypher:

By running our ciphertext through *freq_counter.py*, it gave us this output.

letter frequencies (all):

c : 76 (14.93%)

w : 46 (9.04%) A
u : 37 (7.27%)
i : 36 (7.07%)
l : 33 (6.48%)
s : 32 (6.29%)
t : 31 (6.09%)
z : 29 (5.70%)
q : 28 (5.50%)
r : 27 (5.30%)
p : 22 (4.32%)
y : 20 (3.93%)
v : 14 (2.75%)
k : 12 (2.36%)
j : 11 (2.16%)
n : 10 (1.96%)
f : 9 (1.77%)
b : 8 (1.57%)
, : 7 (1.38%)
m : 7 (1.38%)
d : 5 (0.98%)
. : 3 (0.59%)
o : 2 (0.39%)
(: 1 (0.20%)
) : 1 (0.20%)
e : 1 (0.20%)
h : 1 (0.20%)
writing csv data to: 1grams.csv
top 10 bigrams:
rc : 18 (3.54%)
pr : 15 (2.95%)
ci : 14 (2.76%)
cc : 10 (1.97%)
ic : 10 (1.97%)
sp : 10 (1.97%)
wt : 9 (1.77%)
lz : 9 (1.77%)
zc : 9 (1.77%)
iw : 8 (1.57%)
writing csv data to: 2grams.csv
top 10 trigrams:
prc : 9 (1.78%)
rcc : 8 (1.58%)
cci : 8 (1.58%)
cic : 8 (1.58%)
wtq : 7 (1.38%)
urc : 7 (1.38%)
iwt : 6 (1.18%)
cwq : 6 (1.18%)
spr : 5 (0.99%)
stb : 5 (0.99%)
writing csv data to: 3grams.csv

Through this analysis, we can predict specific what each letter could by based on how frequently they occur in the English Language.
https://www3.nd.edu/~busiforc/handouts/cryptography/Letter%20Frequencies.html
I used the link above to help me guess letters based on the most common n-grams. Once I found a successful tri-gram, Deciphering the text went much faster:

```
[04/14/23]seed@VM:~/.../08_ske$ cat out.txt
A GRILLED CHEESE (SOMETIMES KNOWN AS A TOASTED SANDWICH OR CHEESE TOASTIE) IS A HOT SANDWICH TYPICALLY PREPARED BY H
EATING ONE OR MORE SLICES Oj CHEESE BETWEEN SLICES Oj BREAD, WITH A COOKING jAT SUCH AS BUTTER OR MAYONNAISE ON A jR
YING PAN, GRIDDLE, OR SANDWICH TOASTER, UNTIL THE BREAD BROWNS AND THE CHEESE MELTS. A GRILLED CHEESE SANDWICH IS MA
DE BY PLACING A CHEESE jILLING, OjTEN CHEDDAR OR AMERICAN CHEESE, BETWEEN TWO SLICES Oj BREAD, WHICH IS THEN HEATED
UNTIL THE BREAD BROWNS AND THE CHEESE MELTS. A LAYER Oj BUTTER OR MAYONNAISE MAY BE ADDED TO THE OUTSIDE Oj THE BREA
D jOR ADDITIONAL jLAeOR AND TEhTURE.
[04/14/23]seed@VM:~/.../08_ske$ tr 'wisurcpytbnzqlokvmfdj' 'ASITHECLNGMRDOKWBUYPF' < ciphertext.txt > out.txt
[04/14/23]seed@VM:~/.../08_ske$ cat out.txt
A GRILLED CHEESE (SOMETIMES KNOWN AS A TOASTED SANDWICH OR CHEESE TOASTIE) IS A HOT SANDWICH TYPICALLY PREPARED BY H
EATING ONE OR MORE SLICES OF CHEESE BETWEEN SLICES OF BREAD, WITH A COOKING FAT SUCH AS BUTTER OR MAYONNAISE ON A FR
YING PAN, GRIDDLE, OR SANDWICH TOASTER, UNTIL THE BREAD BROWNS AND THE CHEESE MELTS. A GRILLED CHEESE SANDWICH IS MA
DE BY PLACING A CHEESE FILLING, OFTEN CHEDDAR OR AMERICAN CHEESE, BETWEEN TWO SLICES OF BREAD, WHICH IS THEN HEATED
UNTIL THE BREAD BROWNS AND THE CHEESE MELTS. A LAYER OF BUTTER OR MAYONNAISE MAY BE ADDED TO THE OUTSIDE OF THE BREA
D FOR ADDITIONAL FLAeOR AND TEhTURE.
[04/14/23]seed@VM:~/.../08_ske$ tr 'wisurcpytbnzqlokvmfdje' 'ASITHECLNGMRDOKWBUYPFV' < ciphertext.txt > out.txt
[04/14/23]seed@VM:~/.../08_ske$ cat out.txt
A GRILLED CHEESE (SOMETIMES KNOWN AS A TOASTED SANDWICH OR CHEESE TOASTIE) IS A HOT SANDWICH TYPICALLY PREPARED BY H
EATING ONE OR MORE SLICES OF CHEESE BETWEEN SLICES OF BREAD, WITH A COOKING FAT SUCH AS BUTTER OR MAYONNAISE ON A FR
YING PAN, GRIDDLE, OR SANDWICH TOASTER, UNTIL THE BREAD BROWNS AND THE CHEESE MELTS. A GRILLED CHEESE SANDWICH IS MA
DE BY PLACING A CHEESE FILLING, OFTEN CHEDDAR OR AMERICAN CHEESE, BETWEEN TWO SLICES OF BREAD, WHICH IS THEN HEATED
UNTIL THE BREAD BROWNS AND THE CHEESE MELTS. A LAYER OF BUTTER OR MAYONNAISE MAY BE ADDED TO THE OUTSIDE OF THE BREA
D FOR ADDITIONAL FLAVOR AND TEhTURE.
[04/14/23]seed@VM:~/.../08_ske$ tr 'wisurcpytbnzqlokvmfdjeh' 'ASITHECLNGMRDOKWBUYPFVX' < ciphertext.txt > out.txt
[04/14/23]seed@VM:~/.../08_ske$ cat out.txt
A GRILLED CHEESE (SOMETIMES KNOWN AS A TOASTED SANDWICH OR CHEESE TOASTIE) IS A HOT SANDWICH TYPICALLY PREPARED BY H
EATING ONE OR MORE SLICES OF CHEESE BETWEEN SLICES OF BREAD, WITH A COOKING FAT SUCH AS BUTTER OR MAYONNAISE ON A FR
YING PAN, GRIDDLE, OR SANDWICH TOASTER, UNTIL THE BREAD BROWNS AND THE CHEESE MELTS. A GRILLED CHEESE SANDWICH IS MA
DE BY PLACING A CHEESE FILLING, OFTEN CHEDDAR OR AMERICAN CHEESE, BETWEEN TWO SLICES OF BREAD, WHICH IS THEN HEATED
UNTIL THE BREAD BROWNS AND THE CHEESE MELTS. A LAYER OF BUTTER OR MAYONNAISE MAY BE ADDED TO THE OUTSIDE OF THE BREA
D FOR ADDITIONAL FLAVOR AND TEXTURE.
[04/14/23]seed@VM:~/.../08_ske$
```

Task 2: Encryption Ciphers and Modes

I created a plain.txt file and encrypted it with the 3 different ciphers

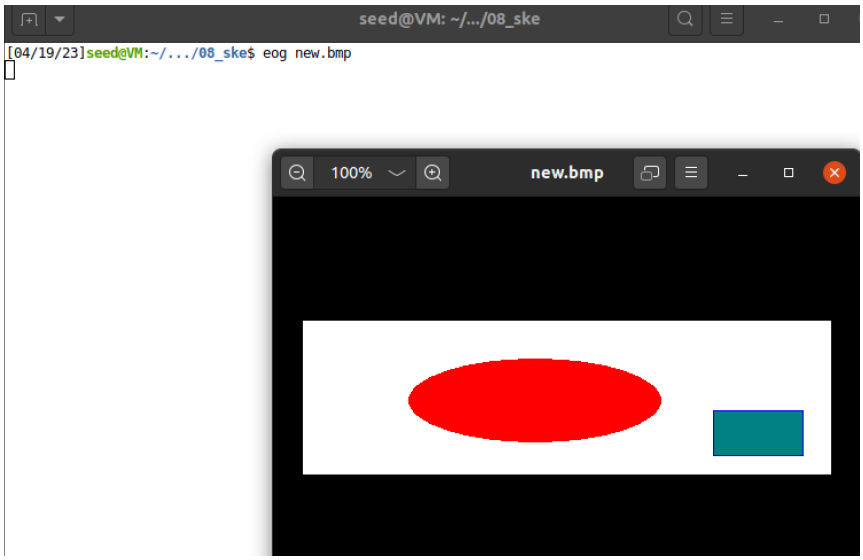
```
[04/19/23]seed@VM:~/.../08_ske$ vim plain.txt
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher.bin -K 0001 -i
v abcd -p
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
salt=2013000000000000
key=00010000000000000000000000000000
iv =ABCD0000000000000000000000000000
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -e -in plain.txt -out cipher.bin -K 0001 -i
v abcd -p
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
salt=2013000000000000
key=00010000000000000000000000000000
iv =ABCD0000000000000000000000000000
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-ofb -e -in plain.txt -out cipher.bin -K 0001 -i
v abcd -p
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
salt=2013000000000000
key=00010000000000000000000000000000
iv =ABCD0000000000000000000000000000
[04/19/23]seed@VM:~/.../08_ske$
```

Task 3.1:

We added padd

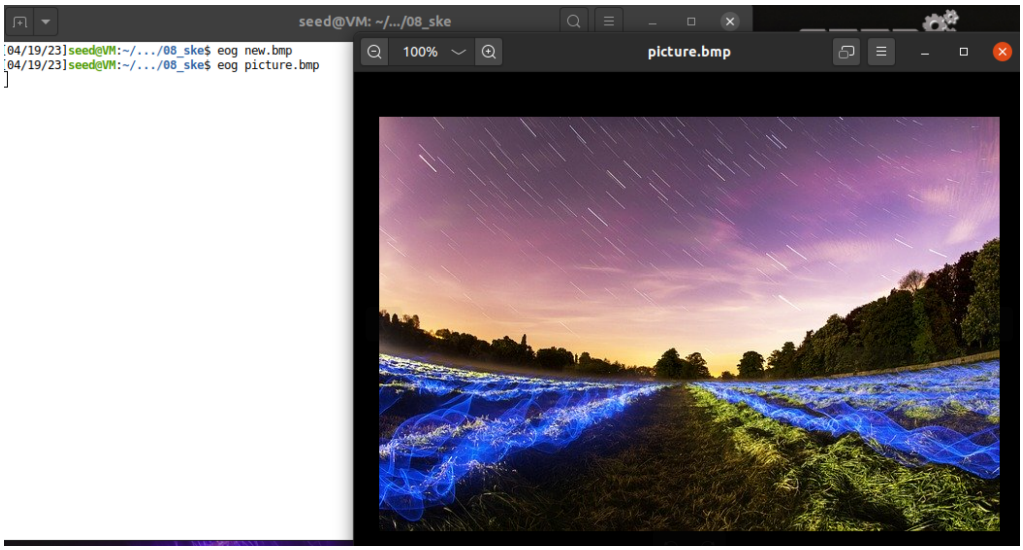
```
[04/19/23]seed@VM:~/.../08_ske$ ls
1grams.csv  3grams.csv  ciphertext.txt   out.txt          plain.txt      words.txt
2grams.csv  cipher.bin  freq_counter.py  pic_original.bmp  sample_xor.py
[04/19/23]seed@VM:~/.../08_ske$ head -c 54 pic_original.bmp > header
[04/19/23]seed@VM:~/.../08_ske$ tail -c +55 pic_original.bmp > body
[04/19/23]seed@VM:~/.../08_ske$ cat header body > new.bmp
[04/19/23]seed@VM:~/.../08_ske$ cat new.bmp
BM66(00X
```

Catting out the .bmp file was a mistake, but I was still able to view the picture with eog for some reason. Looking back, I didn't even type the encryption command which is why I was still able to view it.

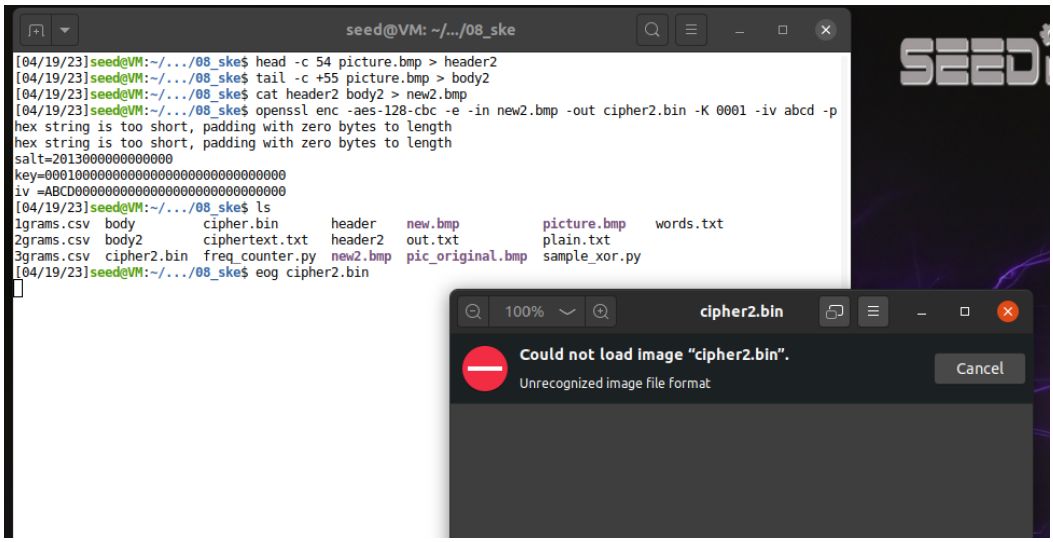


Task 3.2

Before encryption



Now we encrypted it with -aes-128-cbc calling it cipher2.bin, we tried to open it and it couldn't open the file.



Task 4:

I created 3 different files with 5, 10, and 16 bytes (f1.txt, f2.txt, f3.txt respectively) and encrypted them.

```
[04/19/23]seed@VM:~/.../08_ske$ eog cipher2.bin
[04/19/23]seed@VM:~/.../08_ske$ echo -n "12345" > f1.txt
[04/19/23]seed@VM:~/.../08_ske$ echo -n "1234567890" > f2.txt
[04/19/23]seed@VM:~/.../08_ske$ echo -n "1234567890123456" > f3.txt
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in f1.txt -out out1.txt -K 0003 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in f2.txt -out out2.txt -K 0003 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in f3.txt -out out3.txt -K 0003 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out1.txt
00000000  1d 1d c0 1f 58 bc 44 70  79 27 11 ef c2 9c 44 1b  |....X.Dpy'....D.|
00000010
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out2.txt
00000000  d8 8b d9 37 49 d1 7b f8  34 d3 09 67 2f 5b 15 13  |...7I.{.4..g/[..|
00000010
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out3.txt
00000000  c7 3c ca a5 46 b3 28 9d  df d4 5c f8 8d 31 cf 40  |.<..F.(...\..1.@|
00000010  e7 a3 a7 3c 3f 49 1f d4  47 f3 f0 01 60 83 40 25  |...<?I..G...`.@%|
00000020
[04/19/23]seed@VM:~/.../08_ske$
```

With ECB:

```
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in f1.txt -out out1.txt -K ECB -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in f2.txt -out out2.txt -K ECB -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in f3.txt -out out3.txt -K ECB -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ hexdump out1.txt
0000000 ca7a 4bb9 6d23 8da0 0445 3ede 5535 aa2c
0000010
[04/19/23]seed@VM:~/.../08_ske$ cat f1.txt
12345[04/19/23]seed@VM:~/.../08_hexdump out2.txt
0000000 857d d803 ae8a 36b8 3533 a90c e97e abea
0000010
[04/19/23]seed@VM:~/.../08_ske$ hexdump out3.txt
0000000 b8ca 0f70 6512 7942 9ebc fc00 158a 77fc
0000010 08f3 8041 588e b2a9 4812 585b f7f4 1361
0000020
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out1.txt
00000000  7a ca b9 4b 23 6d a0 8d  45 04 de 3e 35 55 2c aa  |z..K#m..E..>5U,.|
00000010
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out2.txt
00000000  7d 85 03 d8 8a ae b8 36  33 35 0c a9 7e e9 ea ab  |}......635..~...|
00000010
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out3.txt
00000000  ca b8 70 0f 12 65 42 79  bc 9e 00 fc 8a 15 fc 77  |..p..eBy.......w|
00000010  f3 08 41 80 8e 58 a9 b2  12 48 5b 58 f4 f7 61 13  |..A..X...H[X..a.|
00000020
[04/19/23]seed@VM:~/.../08_ske$
```

With CFB:

```
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -e -in f1.txt -out out1.txt -K 0001 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -e -in f2.txt -out out2.txt -K 0001 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -e -in f3.txt -out out3.txt -K 0001 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out1.txt
00000000  22 2f 4f 3f ac                                    |"/O?.|
00000005
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out2.txt
00000000  22 2f 4f 3f ac b0 69 3e  5f a9                    |"/O?..i>_.|
0000000a
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out3.txt
00000000  22 2f 4f 3f ac b0 69 3e  5f a9 e4 8c 8f 4d f1 8d  |"/O?..i>_....M..|
00000010
[04/19/23]seed@VM:~/.../08_ske$
```

With OFB:

```
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-ofb -e -in f1.txt -out out1.txt -K 0001 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-ofb -e -in f2.txt -out out2.txt -K 0001 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-ofb -e -in f3.txt -out out3.txt -K 0001 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out1.txt
00000000  22 2f 4f 3f ac                                    |"/O?.|
00000005
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out2.txt
00000000  22 2f 4f 3f ac b0 69 3e  5f a9                    |"/O?..i>_.|
0000000a
[04/19/23]seed@VM:~/.../08_ske$ hexdump -C out3.txt
00000000  22 2f 4f 3f ac b0 69 3e  5f a9 e4 8c 8f 4d f1 8d  |"/O?..i>_....M..|
00000010
[04/19/23]seed@VM:~/.../08_ske$
```

CFB and OFB did not seem to add any additional padding when encrypting.

Task 5.1:

With ECB it depends on where it's corrupted because if one ciphertext block is corrupted then the plaintext block will be corrupted as well.
CBC is not vulnerable to plaintext attacks
Because CFB uses exclusive-ors a single bit corruption can break a ciphertext block. The adjacent blocks still wont be affected though
OFB adds pseudorandom numbers to encrypt plaintext so I'd imagine if the corruption occurred on the pseudorandom numbers, the file itself should likely be fine and you could pull a lot of information from it.

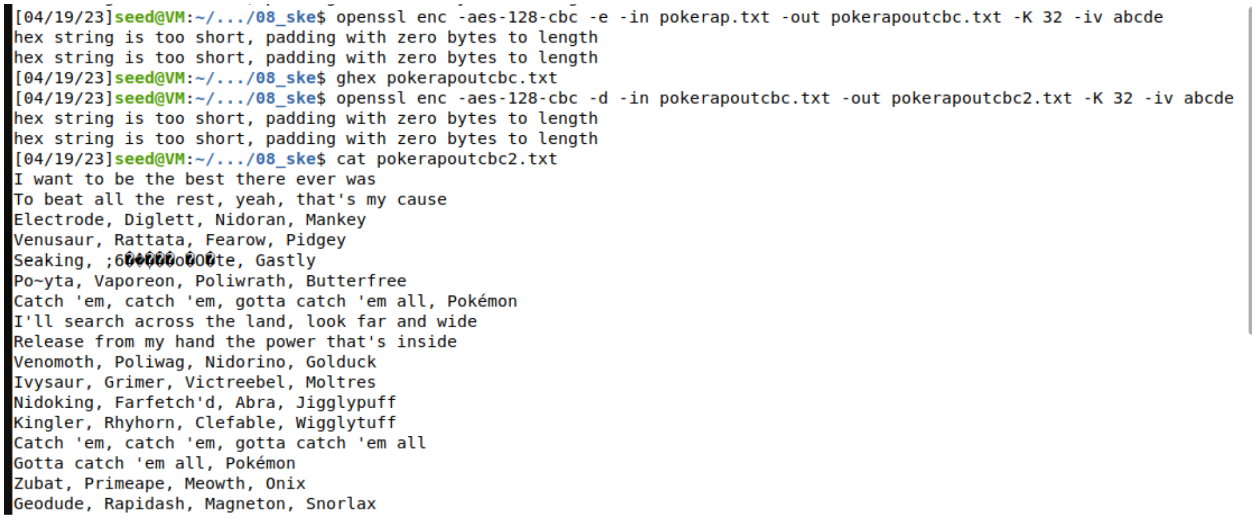
Task 5.2:

We put the pokerap inside a file, encrypted it, then purposely corrupted a bit on the file.

The entire file was corrupted and I was not able to read the original file? This counters my original answer of only a block would be corrupted, I'm not sure if this is an error on my part.

Task 5.3:

Here I used CBC to encrypt the pokerap file, then decrypted it. As you can see, only one string was corrupted.



Task 5.4:

Using the same methods in 5.3, the file was slightly corrupted. This time it appears to have changed the order of some of the text in that first line.

```
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -e -in pokerap.txt -out pokerapoutcfb.txt -K 32 -iv abcde
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ ghex pokerapoutcfb.txt
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -d -in pokerapoutcfb.txt -out pokerapoutcfb2.txt -K 32 -iv abcde
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ cat pokerapoutcfb2.txt
I want to be the bfst there ever��E##-`��{x��bd the rest, yeah, that's my cause
Electrode, Diglett, Nidoran, Mankey
Venusaur, Rattata, Fearow, Pidgey
Seaking, Jolteon, Dragonite, Gastly
Ponyta, Vaporeon, Poliwrath, Butterfree
Catch 'em, catch 'em, gotta catch 'em all, Pokémon
```

Task 6.1:

Here is an encryption with different IV's, I used *diff* to identify the differences in the files:

```
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in plain.txt -out out1.txt -K 32 -iv abcdef
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in plain.txt -out out2.txt -K 32 -iv abcd
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ diff out1.txt out2.txt
1c1
< F"�AIm_��b�H�zb�<��,��J8�F[�
\ No newline at end of file
---
> f�������e4C��sE⋆��g�s�H¬�`
\ No newline at end of file
[04/19/23]seed@VM:~/.../08_ske$
```

Now I encrypted plain.txt with the same IV as out2.txt and used the diff command, there is no difference in files anymore.

```
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in plain.txt -out out2.txt -K 32 -iv abcd
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ diff out1.txt out2.txt
1c1
< F"�AIm_��b�H�zb�<��,��J8�F[�
\ No newline at end of file
---
> f�������e4C��sE⋆��g�s�H¬�`
\ No newline at end of file
[04/19/23]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in plain.txt -out out1.txt -K 32 -iv abcd
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[04/19/23]seed@VM:~/.../08_ske$ diff out1.txt out2.txt
[04/19/23]seed@VM:~/.../08_ske$
```

Task 6.2:
I used the XOR calculator to combine the two ciphertexts then threw it into this hex-to-text converter, but the information was still encrypted.