Emma Stensland

Reese Pearsall

CSCI 476

October 26, 2025

**Seedlabs: Secret Key Encryption Lab**

**Task 1: Breaking A Substitution Cypher**

A substitution cypher was analyzed using the frequency.py script then the cypher was cracked.

```
[10/31/25]seed@VM:~/.../08_ske$ tr 'kbspdzevwtfgxnyujmalohc' 'EATHOFPISCVBMNRWGDKLYUX
' < lab_cipher.txt > out.txt
[10/31/25]seed@VM:~/.../08_ske$ cat out.txt
THE TURKEY IS A LARGE BIRD NATIVE TO NORTH AMERICA. THERE ARE TWO TURKEY SPECIES: THE
 WILD TURKEY  OF EASTERN AND CENTRAL NORTH AMERICA AND THE OCELLATED TURKEY OF MEXICO
. MALES OF BOTH TURKEY SPECIES HAVE A DISTINCTIVE FLESHY WATTLE, CALLED A SNOOD, THAT
 HANGS FROM THE TOP OF THE BEAK.
[10/31/25]seed@VM:~/.../08_ske$
```

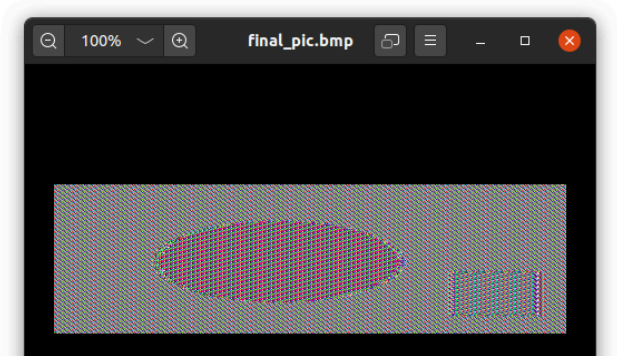
Observations: The frequency analysis directly correlated to the frequencies of the English language, and the cypher was successfully cracked.

**Task 2: Encryption Cyphers and Modes**

The command “openssl rand -hex 16 > iv.txt” and “openssl rand -hex 16 > key.txt” were used to generate hashes. These hashes were then inputted into each encryption algorithm and put into the output

```
[10/31/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-ofb -in out.txt -ou
t cipher1.bin -K $(cat key.txt) -iv $(cat iv.txt) -p
salt=2013000000000000
key=6A6AFA34CFB4C772BB5548222C6CC30E
iv =38EA51E0AB365D6CDA2E648C09D6CB9A
[10/31/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -in out.txt -ou
t cipher2.bin -K $(cat key.txt) -iv $(cat iv.txt) -p
salt=2013000000000000
key=6A6AFA34CFB4C772BB5548222C6CC30E
iv =38EA51E0AB365D6CDA2E648C09D6CB9A
[10/31/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -in out.txt -ou
t cipher3.bin -K $(cat key.txt) -iv $(cat iv.txt) -p
salt=2013000000000000
key=6A6AFA34CFB4C772BB5548222C6CC30E
iv =38EA51E0AB365D6CDA2E648C09D6CB9A
[10/31/25]seed@VM:~/.../08_ske$ cat cipher3.bin
?[illegible]
[illegible]
[illegible]
```

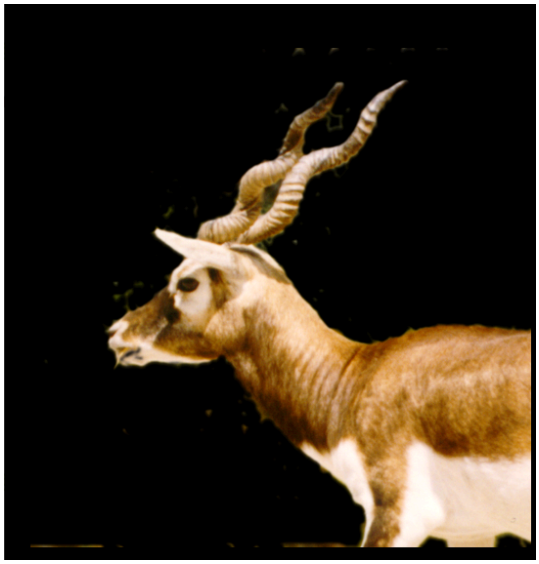
```
[10/31/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-ecb -e -in pic_orig
inal.bmp -out enc_pic.bmp -K $(cat key.txt)
[10/31/25]seed@VM:~/.../08_ske$ head -c 54 pic_original.bmp > header
[10/31/25]seed@VM:~/.../08_ske$ tail -c +55 enc_pic.bmp > body
[10/31/25]seed@VM:~/.../08_ske$ cat header body > final_pic.bmp
[10/31/25]seed@VM:~/.../08_ske$ eog final_pic.bmp
```

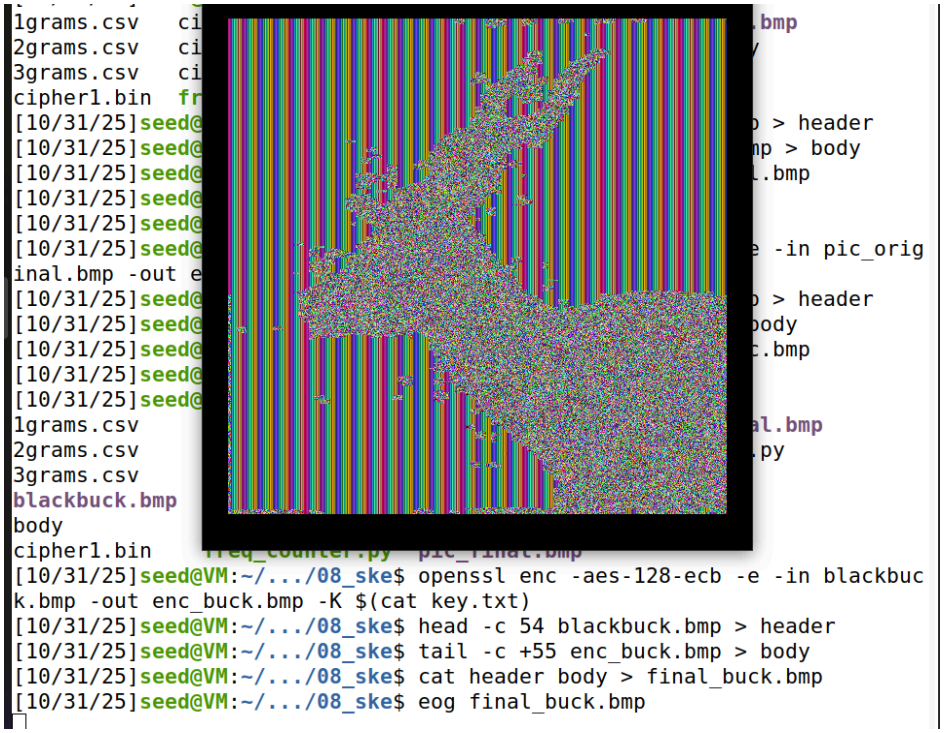



Observations: The image is encrypted, but the general shape is still seen as this is a block cipher and will output matching ciphertexts in the matching blocks.

**3.2**

The process from before was repeated with an image of a buck.






Observations: The image successfully was encrypted, and is more obscure than before but still has a distinct general shape since the block cypher preserves the solid background, while the more complex figure makes for better encryption.

**Task 4: Padding**

**4.1**

A 5, 10, and 16 byte message were created, then encrypted and unencrypted using

AES-128-CBC.

```
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in f1.txt -
out enc_f1.txt -K $(cat key.txt) -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in f2.txt -
out enc_f2.txt -K $(cat key.txt) -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in 3.txt -o
ut enc_f3.txt -K $(cat key.txt) -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ ls -ld enc_f1.txt enc_f2.txt enc_f3.txt
-rw-rw-r-- 1 seed seed 16 Nov  1 17:47 enc_f1.txt
-rw-rw-r-- 1 seed seed 16 Nov  1 17:47 enc_f2.txt
-rw-rw-r-- 1 seed seed 32 Nov  1 17:47 enc_f3.txt
```

```
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -d -nopad -in enc_f1.txt -out denc_1.txt -K $(cat key.txt
 -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -d -nopad -in enc_f2.txt -out denc_2.txt -K $(cat key.txt
 -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -d -nopad -in enc_f3.txt -out denc_3.txt -K $(cat key.txt
 -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ hexdump -C denc_1.txt
00000000  30 31 32 33 34 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |01234...........|
00000010
[11/01/25]seed@VM:~/.../08_ske$ hexdump -C denc_2.txt
00000000  30 31 32 33 34 35 36 37  38 39 06 06 06 06 06 06  |0123456789......|
00000010
[11/01/25]seed@VM:~/.../08_ske$ hexdump -C denc_3.txt
00000000  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35  |0123456789012345|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |................|
00000020
```

Observations: The padding is added as eleven bytes of 0x0b for the 5 byte message, six bytes of 0x06 for the 10 byte message, and 16 bytes of 0x10 for the 16 byte message.

**4.2**

This process was repeated with AES-128-ECB and AES-128-CFB.

```
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-ecb -d -nopad -in
enc_f1.txt -out denc_f1.txt -K $(cat key.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-ecb -d -nopad -in
enc_f2.txt -out denc_f2.txt -K $(cat key.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-ecb -d -nopad -in
enc_3.txt -out denc_3.txt -K $(cat key.txt)
[11/01/25]seed@VM:~/.../08_ske$ hexdump -C denc_f1.txt
00000000  30 31 32 33 34 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |01234.....
......|
00000010
[11/01/25]seed@VM:~/.../08_ske$ hexdump -C denc_f2.txt
00000000  30 31 32 33 34 35 36 37  38 39 06 06 06 06 06 06  |0123456789
......|
00000010
[11/01/25]seed@VM:~/.../08_ske$ hexdump -C denc_3.txt
00000000  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35  |0123456789
012345|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |..........
......|
00000020
```

```
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -d -nopad -in
enc_f1.txt -out denc_f1.txt -K $(cat key.txt) -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -d -nopad -in
enc_f2.txt -out denc_f2.txt -K $(cat key.txt) -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cfb -d -nopad -in
enc_3.txt -out denc_3.txt -K $(cat key.txt) -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ hexdump -C denc_f1.txt
00000000  30 31 32 33 34                                    |01234|
00000005
[11/01/25]seed@VM:~/.../08_ske$ hexdump -C denc_f2.txt
00000000  30 31 32 33 34 35 36 37  38 39                    |0123456789
|
0000000a
[11/01/25]seed@VM:~/.../08_ske$ hexdump -C denc_3.txt
00000000  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35  |0123456789
012345|
00000010
```

Observations: AES-128-CFB does not pad, while AES-128-ECB does pad like CBC.

**Task 5: Common Mistakes with IVs**

**5.1**

Plaintext was encrypted with different IVs and the same IV, and the two were compared.

```
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in plainte
xt.txt -out cipherout.txt -K $(cat key.txt) -iv $(cat iv2.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in plainte
xt.txt -out cipherout2.txt -K $(cat key.txt) -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ hexdump cipherout.txt
0000000 52b4 709b c807 ed2b 2d39 bc30 8e66 2d80
0000010 b9bd f0e0 e582 3206 5f59 ce00 ac4b 3320
0000020 9607 4fd9 5c1f 4ffb 2ee6 1fae cfea 2466
0000030
[11/01/25]seed@VM:~/.../08_ske$ hexdump cipherout2.txt
0000000 1575 3a8c 8ea9 80ae c7a1 6593 f334 59c4
0000010 320d 4170 0f27 1ba7 0440 34ac 7e64 f7c7
0000020 ffdd 0cb4 d1c8 be36 93c6 f8a1 329b 90ff
0000030
```

```
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in plain
xt.txt -out cipherout.txt -K $(cat key.txt) -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ openssl enc -aes-128-cbc -e -in plain
xt.txt -out cipherout2.txt -K $(cat key.txt) -iv $(cat iv.txt)
[11/01/25]seed@VM:~/.../08_ske$ hexdump cipherout.txt
0000000 1575 3a8c 8ea9 80ae c7a1 6593 f334 59c4
0000010 320d 4170 0f27 1ba7 0440 34ac 7e64 f7c7
0000020 ffdd 0cb4 d1c8 be36 93c6 f8a1 329b 90ff
0000030
[11/01/25]seed@VM:~/.../08_ske$ hexdump cipherout2.txt
0000000 1575 3a8c 8ea9 80ae c7a1 6593 f334 59c4
0000010 320d 4170 0f27 1ba7 0440 34ac 7e64 f7c7
0000020 ffdd 0cb4 d1c8 be36 93c6 f8a1 329b 90ff
0000030
```

Observations: With different IVs, they had different outputs, but with the same IV they generated the came ciphertext.

**5.2**

A python script was written to XOR the ciphertext to the known plaintext, then the recovered IV/Key pair was XORed with the mystery ciphertext.

```
[11/01/25]seed@VM:~/.../08_ske$ ./xor.py
f001d8b622a8b99907b6353e2d2356c1d67e2ce356c3a478
Order: Launch a missile!
[11/01/25]seed@VM:~/.../08_ske$ cat xor.py
#!/usr/bin/python3

#
# XOR strings (ascii strings and hex strings).
#

MSG   = "This is a known message!"
HEX_1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159"
HEX_2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159"

# XOR two bytearrays
def xor(first, second):
   return bytearray(x^y for x,y in zip(first, second))

D1 = bytes(MSG, 'utf-8')      # Convert ascii string to bytearray
D2 = bytearray.fromhex(HEX_1) # Convert hex string to bytearray
D3 = bytearray.fromhex(HEX_2) # Convert hex string to bytearray

r1 = xor(D1, D2)
r2 = xor(r1, D3)
print(r1.hex())
print(r2.decode())
```

Observations: The message was successfully decoded and said “Order: Launch a missile!”