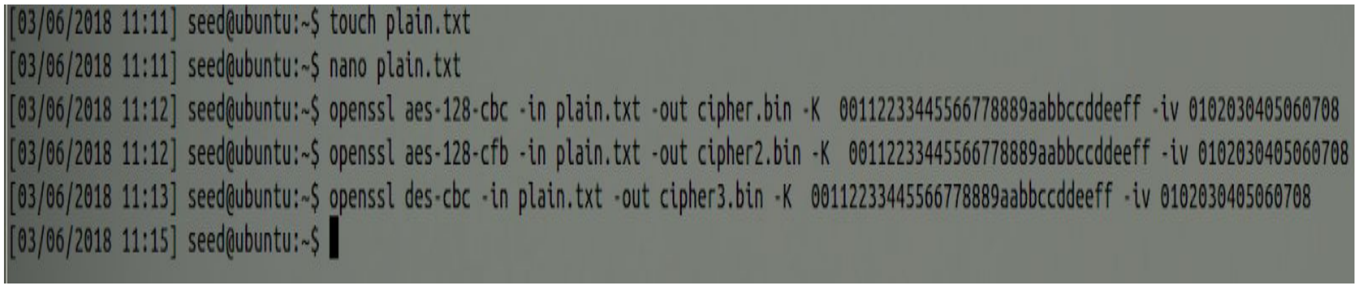
Joshua Braden
Yesenia Valles
DFSC 4317 Lab 5

Task 1: Encrypt a file with 3 different ciphers

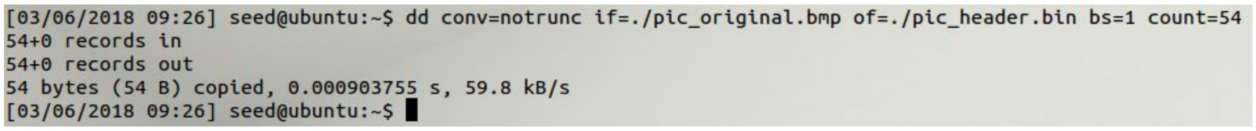
Here, we used aes-128-cbc, aes-128-cfb, des-cbc on the plain.txt file



```
[03/06/2018 11:11] seed@ubuntu:~$ touch plain.txt
[03/06/2018 11:11] seed@ubuntu:~$ nano plain.txt
[03/06/2018 11:12] seed@ubuntu:~$ openssl aes-128-cbc -in plain.txt -out cipher.bin -K  00112233445566778889aabbccddeeff -iv 0102030405060708
[03/06/2018 11:12] seed@ubuntu:~$ openssl aes-128-cfb -in plain.txt -out cipher2.bin -K  00112233445566778889aabbccddeeff -iv 0102030405060708
[03/06/2018 11:13] seed@ubuntu:~$ openssl des-cbc -in plain.txt -out cipher3.bin -K  00112233445566778889aabbccddeeff -iv 0102030405060708
[03/06/2018 11:15] seed@ubuntu:~$ 
```

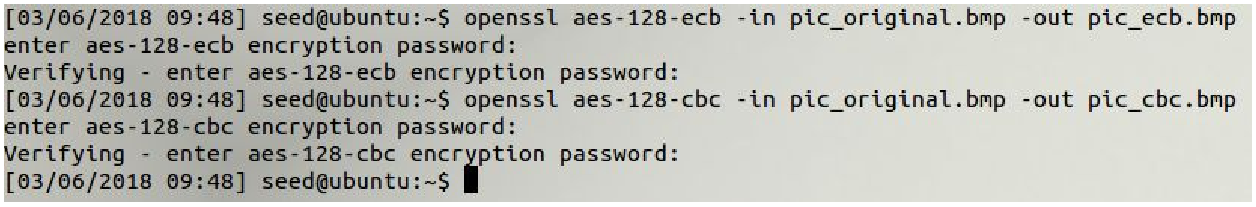
Task 2: Encryption Mode - ECB vs. CBC

Extracted the file header (first 54 bytes) from the .bmp using dd:



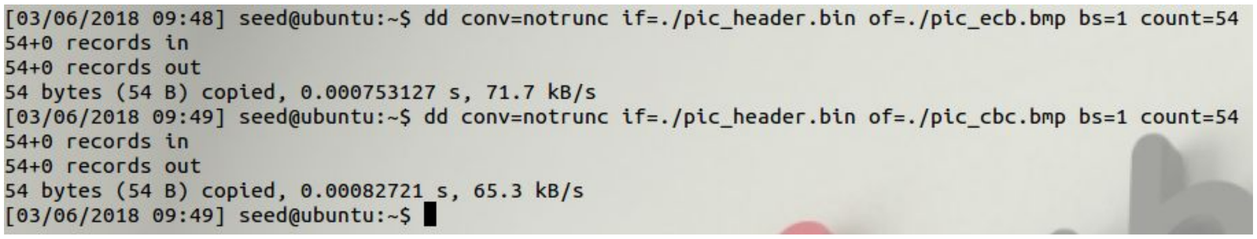
```
[03/06/2018 09:26] seed@ubuntu:~$ dd conv=notrunc if=./pic_original.bmp of=./pic_header.bin bs=1 count=54
54+0 records in
54+0 records out
54 bytes (54 B) copied, 0.000903755 s, 59.8 kB/s
[03/06/2018 09:26] seed@ubuntu:~$ 
```

Encrypted the picture file with both cbc and ecb modes:



```
[03/06/2018 09:48] seed@ubuntu:~$ openssl aes-128-ecb -in pic_original.bmp -out pic_ecb.bmp
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
[03/06/2018 09:48] seed@ubuntu:~$ openssl aes-128-cbc -in pic_original.bmp -out pic_cbc.bmp
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[03/06/2018 09:48] seed@ubuntu:~$ 
```

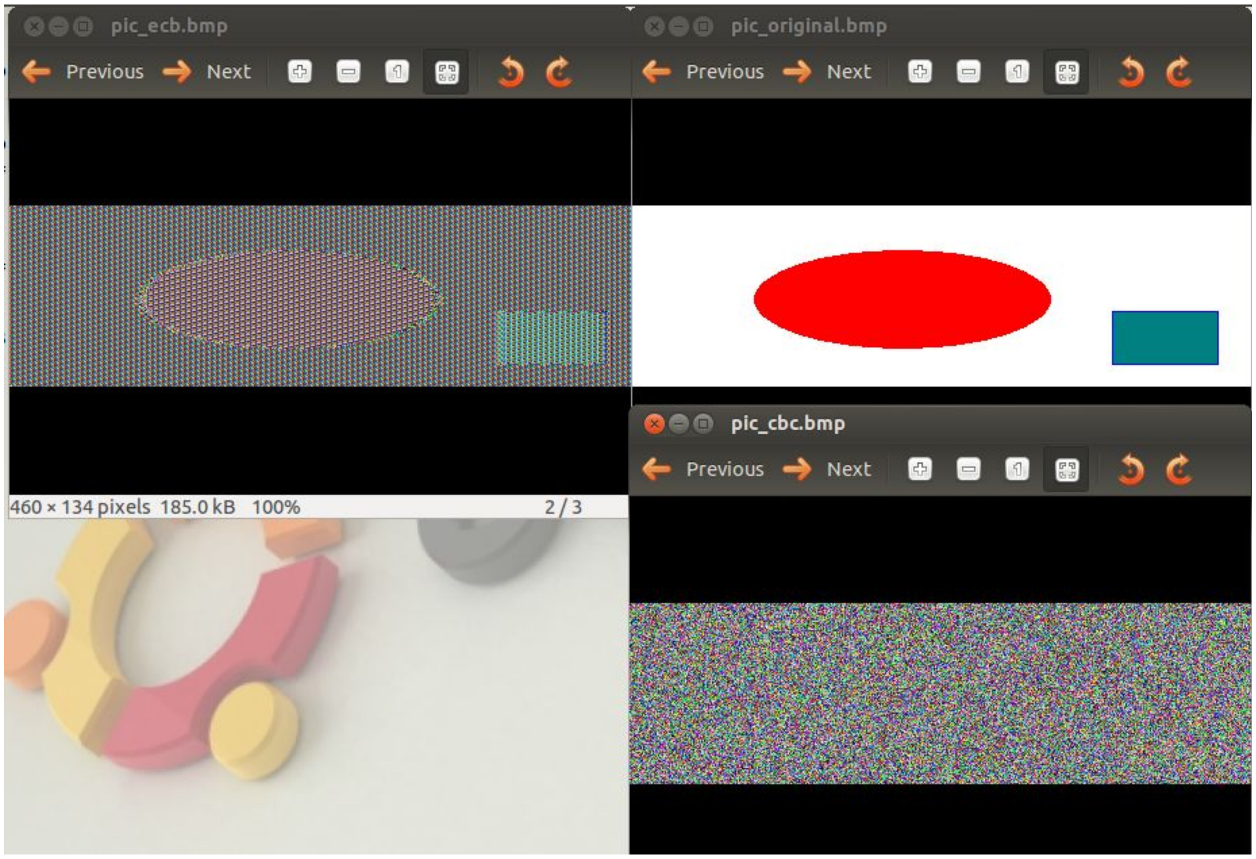
Copied the header back to the encrypted files:



```
[03/06/2018 09:48] seed@ubuntu:~$ dd conv=notrunc if=./pic_header.bin of=./pic_ecb.bmp bs=1 count=54
54+0 records in
54+0 records out
54 bytes (54 B) copied, 0.000753127 s, 71.7 kB/s
[03/06/2018 09:49] seed@ubuntu:~$ dd conv=notrunc if=./pic_header.bin of=./pic_cbc.bmp bs=1 count=54
54+0 records in
54+0 records out
54 bytes (54 B) copied, 0.00082721 s, 65.3 kB/s
[03/06/2018 09:49] seed@ubuntu:~$ 
```

Joshua Braden
Yesenia Valles
DFSC 4317 Lab 5

Displayed the two encrypted images, and the original:

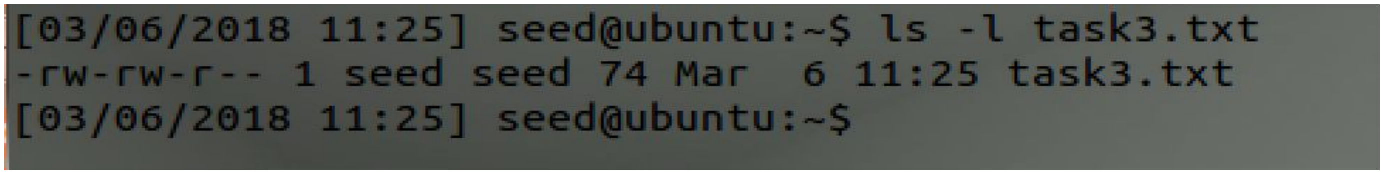


The ECB encrypted file is similar to the original, and an observer can approximate the original file’s layout. The CBC encrypted file, however, appears to be totally random noise.

Task 3: Encryption Mode

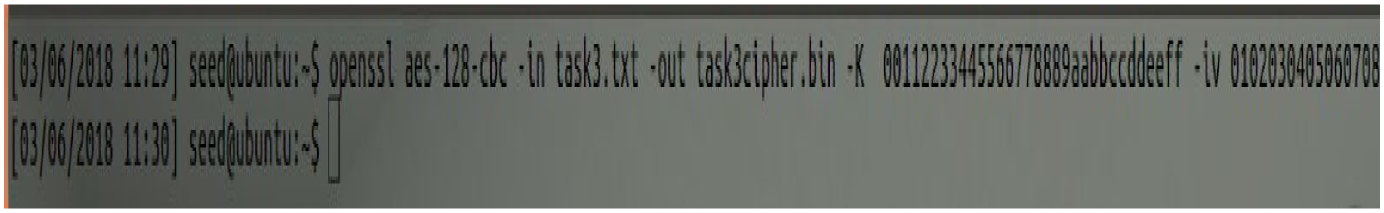
1. Create a file that is at least 64 bytes long.

Here, we created task3.txt that is 74 bytes long, meeting the criteria.



```
[03/06/2018 11:25] seed@ubuntu:~$ ls -l task3.txt
-rw-rw-r-- 1 seed seed 74 Mar  6 11:25 task3.txt
[03/06/2018 11:25] seed@ubuntu:~$
```

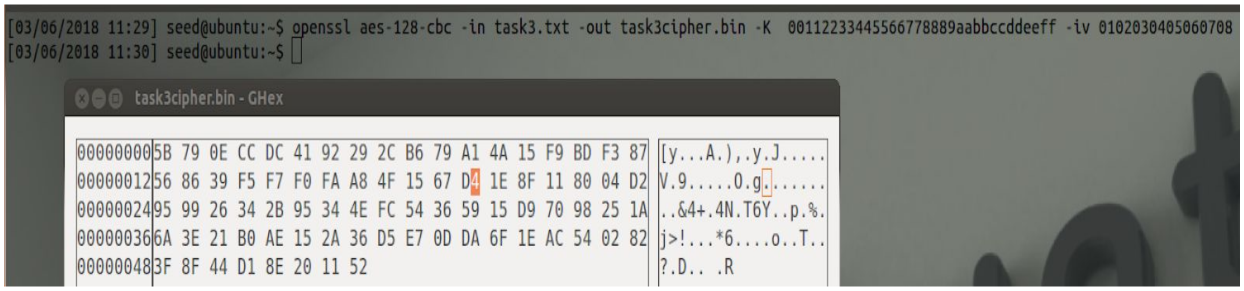
2. Encrypt the file using the AES-128 cipher.



```
[03/06/2018 11:29] seed@ubuntu:~$ openssl aes-128-cbc -in task3.txt -out task3cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/06/2018 11:30] seed@ubuntu:~$
```

Joshua Braden
Yesenia Valles
DFSC 4317 Lab 5

3. Corrupt the 30th byte with a hex editor.
30th byte was originally D5, changed it to D4.

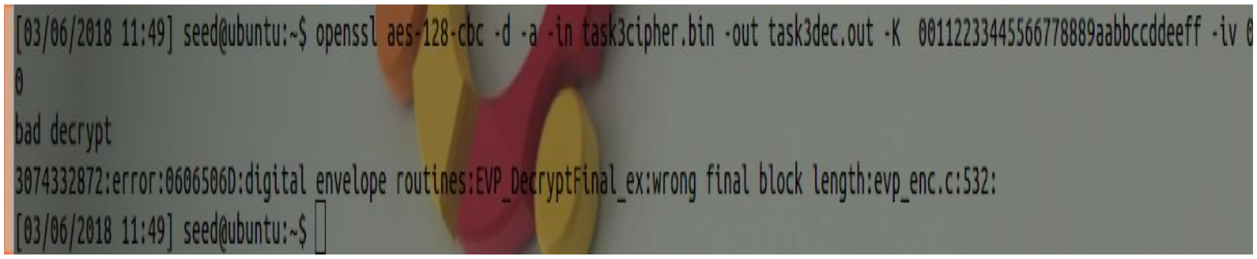


```
[03/06/2018 11:29] seed@ubuntu:~$ openssl aes-128-cbc -in task3.txt -out task3cipher.bin -K  00112233445566778889aabbccddeeff -iv 0102030405060708
[03/06/2018 11:30] seed@ubuntu:~$ 
```

task3cipher.bin - GHex

```
000000005B 79 0E CC DC 41 92 29 2C B6 79 A1 4A 15 F9 BD F3 87 [y...A.),.y.J.....
0000001256 86 39 F5 F7 F0 FA A8 4F 15 67 D4 1E 8F 11 80 04 D2 V.9.....O.g.......
0000002495 99 26 34 2B 95 34 4E FC 54 36 59 15 D9 70 98 25 1A ..&4+.4N.T6Y..p.%.
000000366A 3E 21 B0 AE 15 2A 36 D5 E7 0D DA 6F 1E AC 54 02 82 j>!...*6....o..T..
000000483F 8F 44 D1 8E 20 11 52                               ?.D.. .R
```

4. Run a decrypt



```
[03/06/2018 11:49] seed@ubuntu:~$ openssl aes-128-cbc -d -a -in task3cipher.bin -out task3dec.out -K  00112233445566778889aabbccddeeff -iv 0
0
bad decrypt
3074332872:error:0606506D:digital envelope routines:EVP_DecryptFinal_ex:wrong final block length:evp_enc.c:532:
[03/06/2018 11:49] seed@ubuntu:~$ 
```

When running the decryption, it resulted in a bad decrypt because the byte was corrupted by us!

(1) How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively?
Please answer this question before you conduct this task, and then find out whether your answer is correct or wrong after you finish this task. (2) Please explain why. (3) What are the implication of these differences?

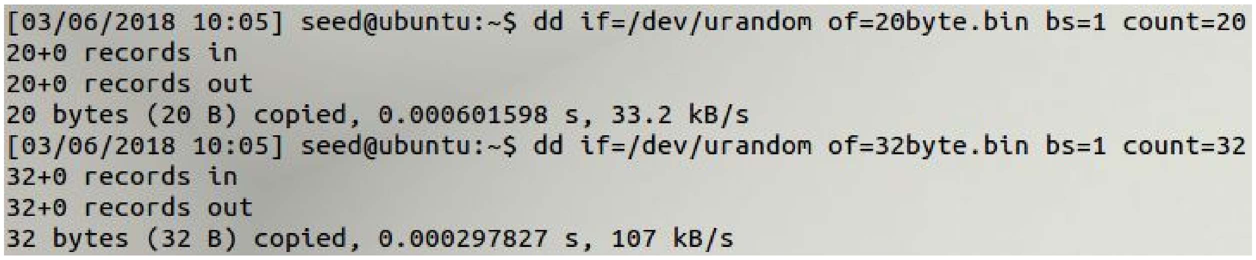
ECB	All but 1 corrupted block
CBC	All but 2 corrupted blocks (N-2)
CFB	All but 2 corrupted blocks
OFB	All but 1 corrupted block

The multiple block error propagation is because the of the block modes' dependency on the prior block, since it serves as the initialization for the current encryption. OFB and ECB do not use previous blocks during encryption, so only 1 block would be corrupted. Because of these differences in the encryption process, the damaged data amount can be predicted.

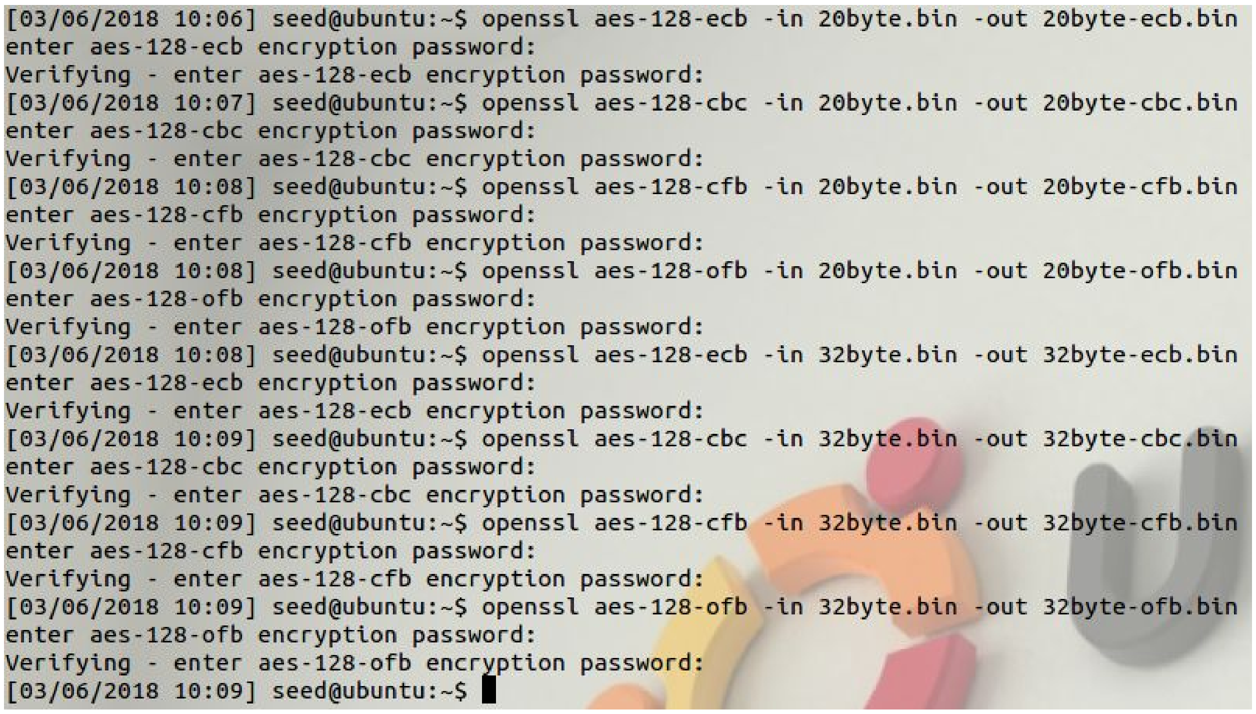
Joshua Braden
Yesenia Valles
DFSC 4317 Lab 5

Task 4: Padding

To test padding in encryption modes, created a 20-byte and 32-byte file and encrypted them with ECB, CBC, CFB, and OFB encryption modes:

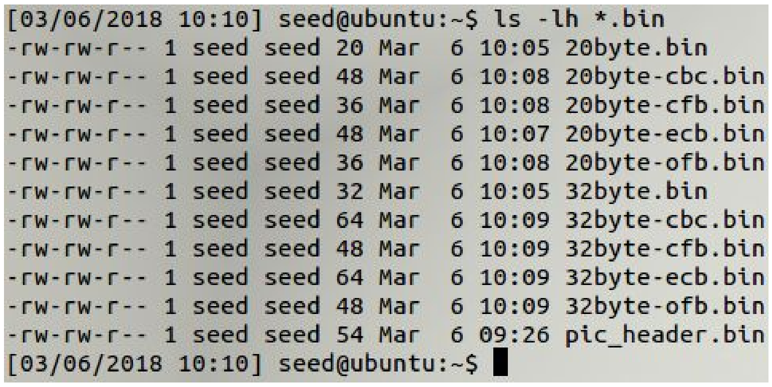


```
[03/06/2018 10:05] seed@ubuntu:~$ dd if=/dev/urandom of=20byte.bin bs=1 count=20
20+0 records in
20+0 records out
20 bytes (20 B) copied, 0.000601598 s, 33.2 kB/s
[03/06/2018 10:05] seed@ubuntu:~$ dd if=/dev/urandom of=32byte.bin bs=1 count=32
32+0 records in
32+0 records out
32 bytes (32 B) copied, 0.000297827 s, 107 kB/s
```



```
[03/06/2018 10:06] seed@ubuntu:~$ openssl aes-128-ecb -in 20byte.bin -out 20byte-ecb.bin
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
[03/06/2018 10:07] seed@ubuntu:~$ openssl aes-128-cbc -in 20byte.bin -out 20byte-cbc.bin
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[03/06/2018 10:08] seed@ubuntu:~$ openssl aes-128-cfb -in 20byte.bin -out 20byte-cfb.bin
enter aes-128-cfb encryption password:
Verifying - enter aes-128-cfb encryption password:
[03/06/2018 10:08] seed@ubuntu:~$ openssl aes-128-ofb -in 20byte.bin -out 20byte-ofb.bin
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
[03/06/2018 10:08] seed@ubuntu:~$ openssl aes-128-ecb -in 32byte.bin -out 32byte-ecb.bin
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
[03/06/2018 10:09] seed@ubuntu:~$ openssl aes-128-cbc -in 32byte.bin -out 32byte-cbc.bin
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
[03/06/2018 10:09] seed@ubuntu:~$ openssl aes-128-cfb -in 32byte.bin -out 32byte-cfb.bin
enter aes-128-cfb encryption password:
Verifying - enter aes-128-cfb encryption password:
[03/06/2018 10:09] seed@ubuntu:~$ openssl aes-128-ofb -in 32byte.bin -out 32byte-ofb.bin
enter aes-128-ofb encryption password:
Verifying - enter aes-128-ofb encryption password:
[03/06/2018 10:09] seed@ubuntu:~$
```

The resulting files:



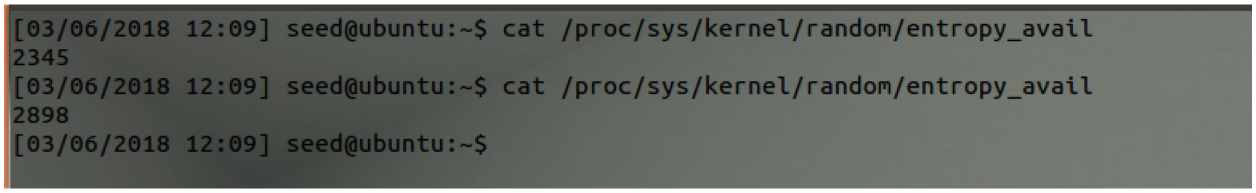
```
[03/06/2018 10:10] seed@ubuntu:~$ ls -lh *.bin
-rw-rw-r-- 1 seed seed 20 Mar  6 10:05 20byte.bin
-rw-rw-r-- 1 seed seed 48 Mar  6 10:08 20byte-cbc.bin
-rw-rw-r-- 1 seed seed 36 Mar  6 10:08 20byte-cfb.bin
-rw-rw-r-- 1 seed seed 48 Mar  6 10:07 20byte-ecb.bin
-rw-rw-r-- 1 seed seed 36 Mar  6 10:08 20byte-ofb.bin
-rw-rw-r-- 1 seed seed 32 Mar  6 10:05 32byte.bin
-rw-rw-r-- 1 seed seed 64 Mar  6 10:09 32byte-cbc.bin
-rw-rw-r-- 1 seed seed 48 Mar  6 10:09 32byte-cfb.bin
-rw-rw-r-- 1 seed seed 64 Mar  6 10:09 32byte-ecb.bin
-rw-rw-r-- 1 seed seed 48 Mar  6 10:09 32byte-ofb.bin
-rw-rw-r-- 1 seed seed 54 Mar  6 09:26 pic_header.bin
[03/06/2018 10:10] seed@ubuntu:~$
```

It appears that when using aes-128, the CFB and OFB modes pad 16 bytes to the end of the file, while CBC and ECB pad until a multiple of 8 is reached.

Joshua Braden
Yesenia Valles
DFSC 4317 Lab 5

Task 5: Pseudo Random Number Generation

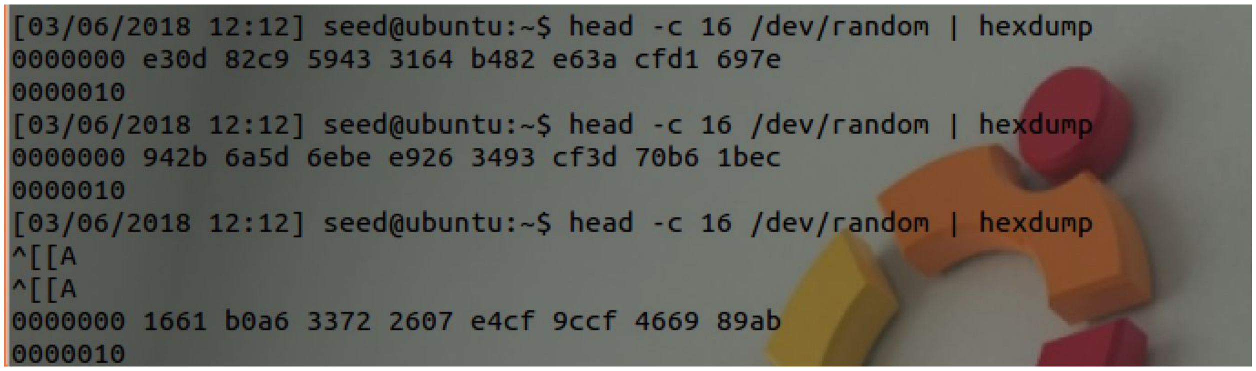
5a. Determine how much randomness you initially have and then perform random actions, see the difference.



```
[03/06/2018 12:09] seed@ubuntu:~$ cat /proc/sys/kernel/random/entropy_avail
2345
[03/06/2018 12:09] seed@ubuntu:~$ cat /proc/sys/kernel/random/entropy_avail
2898
[03/06/2018 12:09] seed@ubuntu:~$
```

Entropy increased.

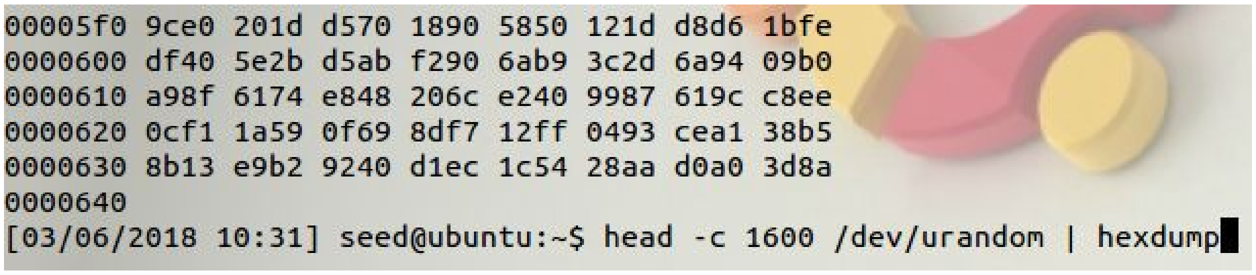
5b. Get Random Numbers from /dev/random



```
[03/06/2018 12:12] seed@ubuntu:~$ head -c 16 /dev/random | hexdump
0000000 e30d 82c9 5943 3164 b482 e63a cfd1 697e
0000010
[03/06/2018 12:12] seed@ubuntu:~$ head -c 16 /dev/random | hexdump
0000000 942b 6a5d 6ebe e926 3493 cf3d 70b6 1bec
0000010
[03/06/2018 12:12] seed@ubuntu:~$ head -c 16 /dev/random | hexdump
^[[A
^[[A
0000000 1661 b0a6 3372 2607 e4cf 9ccf 4669 89ab
0000010
```

At some point, there was a pause, indicating that entropy was depleted.

5c. Get Pseudo Random Numbers from /dev/urandom



```
00005f0 9ce0 201d d570 1890 5850 121d d8d6 1bfe
0000600 df40 5e2b d5ab f290 6ab9 3c2d 6a94 09b0
0000610 a98f 6174 e848 206c e240 9987 619c c8ee
0000620 0cf1 1a59 0f69 8df7 12ff 0493 cea1 38b5
0000630 8b13 e9b2 9240 d1ec 1c54 28aa d0a0 3d8a
0000640
[03/06/2018 10:31] seed@ubuntu:~$ head -c 1600 /dev/urandom | hexdump
```

Unlike the previous command, when using /dev/urandom source depletion is never encountered.