Introduction to Symmetric Cryptography and Secret-key encryption

Aim

The aim of this lab is to get familiar with the concepts in secret-key encryption and get familiar with tools to encrypt/decrypt messages.

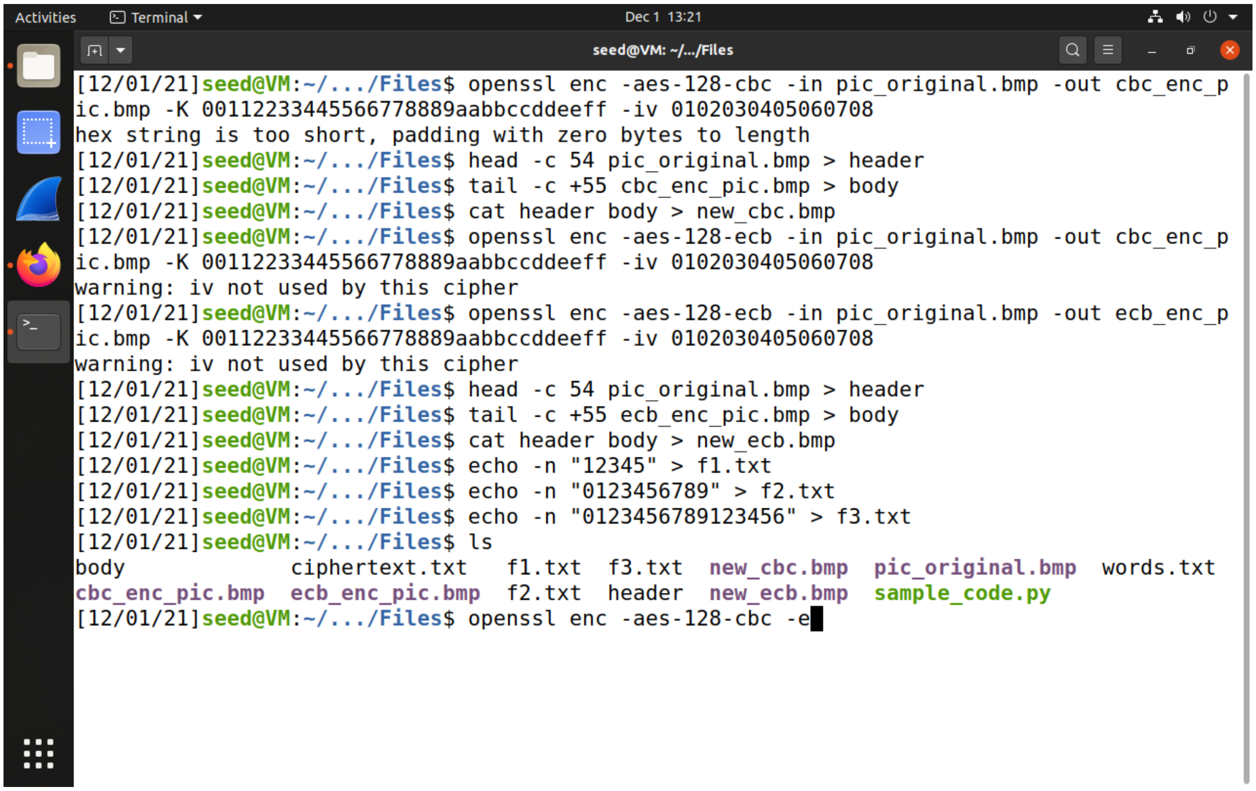
Introduction and Background

The main focus of this lab is to gain first hand experience on encryption algorithms, encryption modes, paddings, and initial vector (IV). To demonstrate these concepts we encrypt a picture using two different encryption modes and compare the results and finally we see how block ciphers pad the data for encryption to fit their block size

Methods

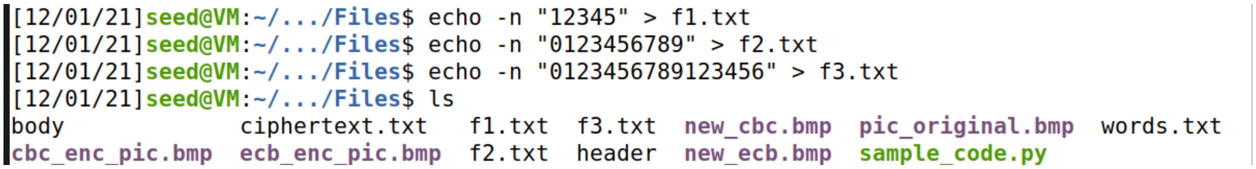
First, we get familiar with various encryption algorithms and modes. We use the `openssl enc` command to encrypt/decrypt a file. We take a text file and encrypt it with the following three different ciphers - `-aes-128-cbc, -bf-cbc, -aes-128-cfb.`

Next, we take a simple picture and encrypt it in a way such that people without the encryption keys cannot know what is in the picture. We encrypt the picture using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes. Now we would like to view the encrypted picture, but for the encrypted files to be considered a legitimate .bmp file we have to take the first 54 bytes which contain the header information about the picture, and replace the header of the encrypted picture with these 54 bytes, offsetting the body by 55 bytes. We do this using the following commands -



```
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -in pic_original.bmp -out cbc_enc_p
ic.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[12/01/21]seed@VM:~/.../Files$ head -c 54 pic_original.bmp > header
[12/01/21]seed@VM:~/.../Files$ tail -c +55 cbc_enc_pic.bmp > body
[12/01/21]seed@VM:~/.../Files$ cat header body > new_cbc.bmp
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-ecb -in pic_original.bmp -out cbc_enc_p
ic.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-ecb -in pic_original.bmp -out ecb_enc_p
ic.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708
warning: iv not used by this cipher
[12/01/21]seed@VM:~/.../Files$ head -c 54 pic_original.bmp > header
[12/01/21]seed@VM:~/.../Files$ tail -c +55 ecb_enc_pic.bmp > body
[12/01/21]seed@VM:~/.../Files$ cat header body > new_ecb.bmp
[12/01/21]seed@VM:~/.../Files$ echo -n "12345" > f1.txt
[12/01/21]seed@VM:~/.../Files$ echo -n "0123456789" > f2.txt
[12/01/21]seed@VM:~/.../Files$ echo -n "0123456789123456" > f3.txt
[12/01/21]seed@VM:~/.../Files$ ls
body             ciphertext.txt   f1.txt  f3.txt  new_cbc.bmp  pic_original.bmp  words.txt
cbc_enc_pic.bmp  ecb_enc_pic.bmp  f2.txt  header  new_ecb.bmp  sample_code.py
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e
```

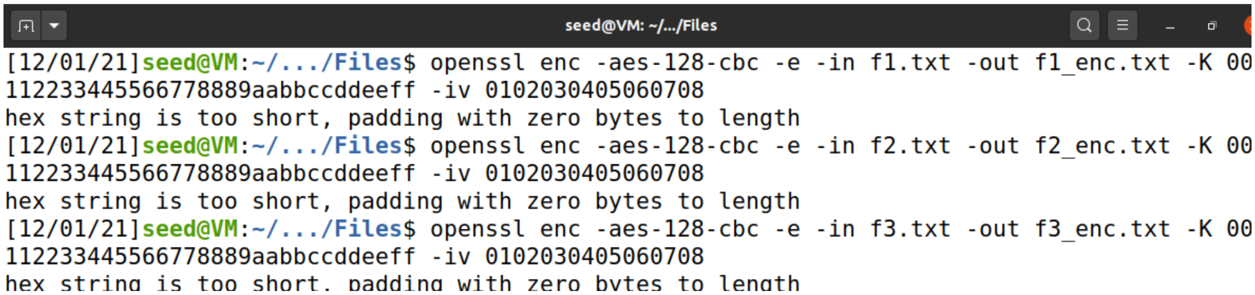
Finally we see the paddings in block ciphers. We first create 3 files f1.txt, f2.txt, f3.txt of size 5, 10, 16.



```
[12/01/21]seed@VM:~/.../Files$ echo -n "12345" > f1.txt
[12/01/21]seed@VM:~/.../Files$ echo -n "0123456789" > f2.txt
[12/01/21]seed@VM:~/.../Files$ echo -n "0123456789123456" > f3.txt
[12/01/21]seed@VM:~/.../Files$ ls
body             ciphertext.txt   f1.txt  f3.txt  new_cbc.bmp  pic_original.bmp  words.txt
cbc_enc_pic.bmp  ecb_enc_pic.bmp  f2.txt  header  new_ecb.bmp  sample_code.py
```

We then use "`openssl enc -aes-128-cbc -e`" to encrypt these three files using 128-bit AES with CBC mode.

Now to check what's added to the padding we decrypt these files using "`openssl enc -aes-128-cbc -nopad -d`". The option "-nopad", disables the padding, i.e., during the decryption, the command will not remove the padded data.

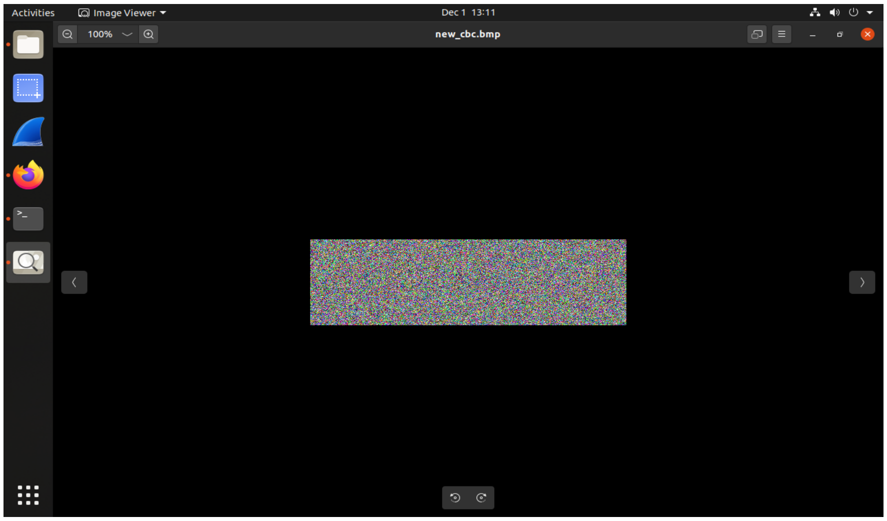


```
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in f1.txt -out f1_enc.txt -K 00
112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in f2.txt -out f2_enc.txt -K 00
112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in f3.txt -out f3_enc.txt -K 00
112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

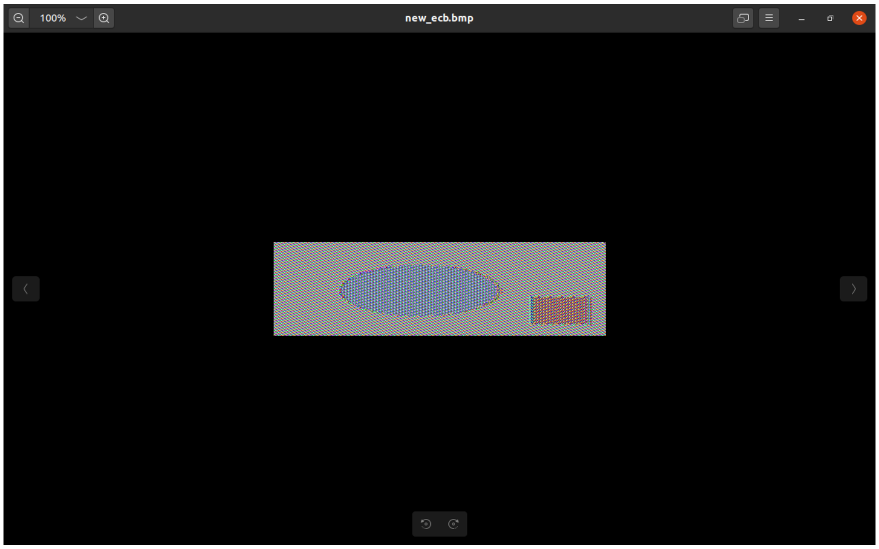
Results and Discussion

For the encrypted pictures we use a picture viewing program to check both encrypted pictures and see if we can derive any meaningful information.

CBC

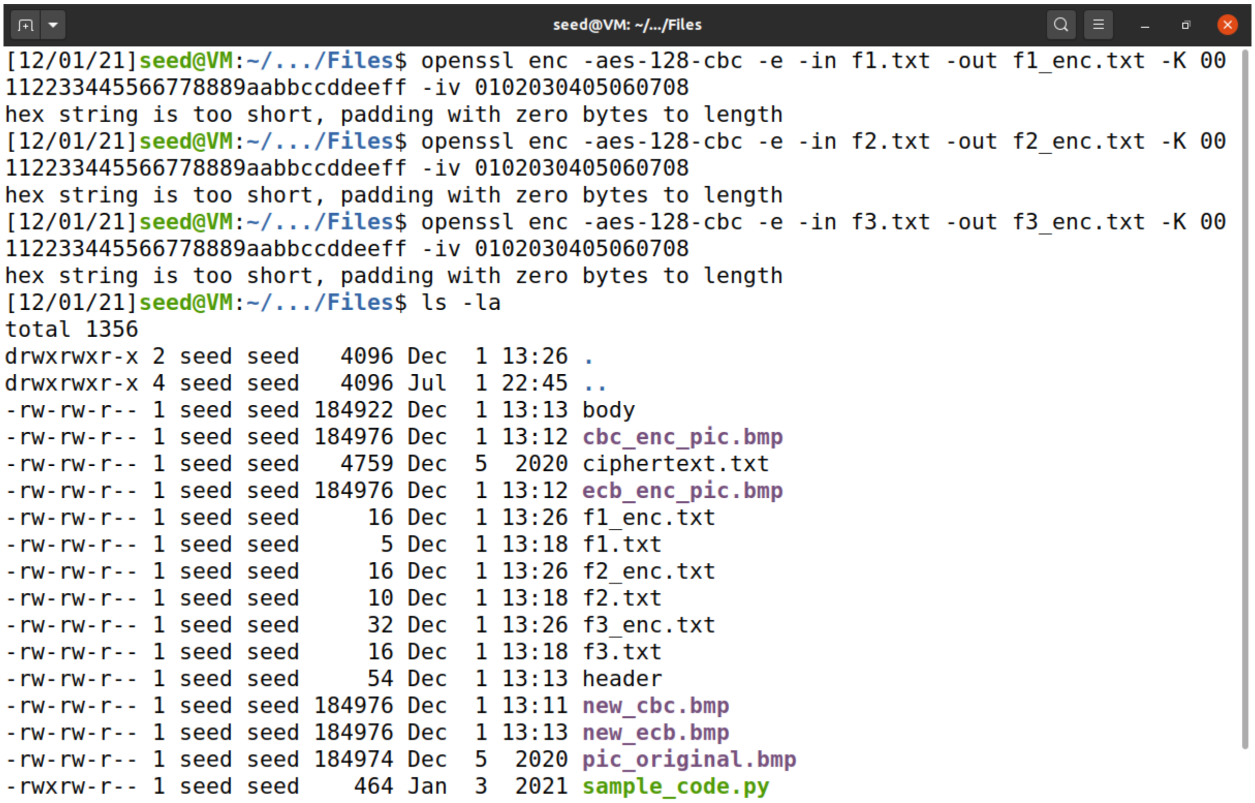


EBC



We see that in case cbc we can not find any relation to the original picture, but in the case of the picture being encrypted through EBC, we can still make out what the original image looked like.

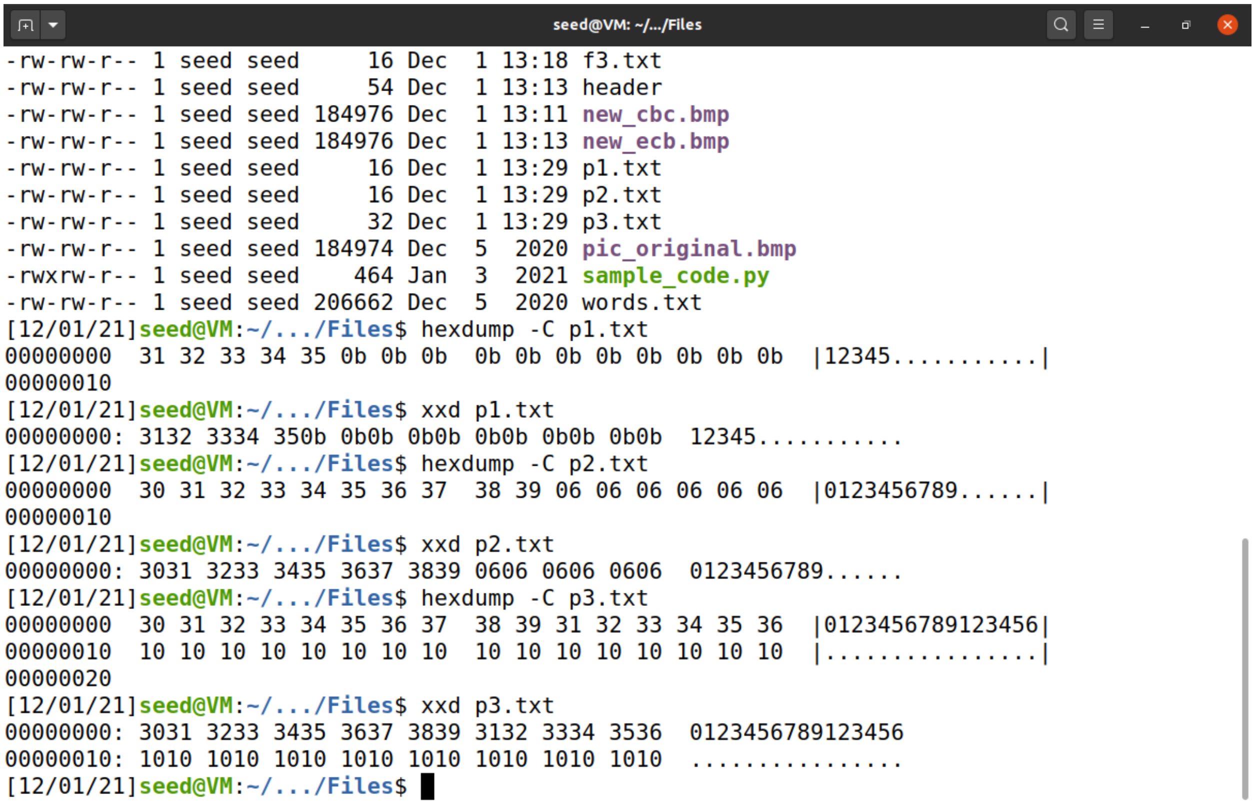
Finally in the task for padding, we compare the size of the original files to the size of the encrypted files.



```
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in f1.txt -out f1_enc.txt -K 00
112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in f2.txt -out f2_enc.txt -K 00
112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[12/01/21]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in f3.txt -out f3_enc.txt -K 00
112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[12/01/21]seed@VM:~/.../Files$ ls -la
total 1356
drwxrwxr-x 2 seed seed   4096 Dec  1 13:26 .
drwxrwxr-x 4 seed seed   4096 Jul  1 22:45 ..
-rw-rw-r-- 1 seed seed 184922 Dec  1 13:13 body
-rw-rw-r-- 1 seed seed 184976 Dec  1 13:12 cbc_enc_pic.bmp
-rw-rw-r-- 1 seed seed   4759 Dec  5  2020 ciphertext.txt
-rw-rw-r-- 1 seed seed 184976 Dec  1 13:12 ecb_enc_pic.bmp
-rw-rw-r-- 1 seed seed     16 Dec  1 13:26 f1_enc.txt
-rw-rw-r-- 1 seed seed      5 Dec  1 13:18 f1.txt
-rw-rw-r-- 1 seed seed     16 Dec  1 13:26 f2_enc.txt
-rw-rw-r-- 1 seed seed     10 Dec  1 13:18 f2.txt
-rw-rw-r-- 1 seed seed     32 Dec  1 13:26 f3_enc.txt
-rw-rw-r-- 1 seed seed     16 Dec  1 13:18 f3.txt
-rw-rw-r-- 1 seed seed     54 Dec  1 13:13 header
-rw-rw-r-- 1 seed seed 184976 Dec  1 13:11 new_cbc.bmp
-rw-rw-r-- 1 seed seed 184976 Dec  1 13:13 new_ecb.bmp
-rw-rw-r-- 1 seed seed 184974 Dec  5  2020 pic_original.bmp
-rwxrw-r-- 1 seed seed    464 Jan  3  2021 sample_code.py
```

We see that file f1.txt was padded with 11 bytes, f2.txt was padded with 6 bytes and f3.txt was padded with 16 bytes.

Now we decrypt these files and use a hex editor to check what was exactly added as padding to these files.



```
-rw-rw-r-- 1 seed seed     16 Dec  1 13:18 f3.txt
-rw-rw-r-- 1 seed seed     54 Dec  1 13:13 header
-rw-rw-r-- 1 seed seed 184976 Dec  1 13:11 new_cbc.bmp
-rw-rw-r-- 1 seed seed 184976 Dec  1 13:13 new_ecb.bmp
-rw-rw-r-- 1 seed seed     16 Dec  1 13:29 p1.txt
-rw-rw-r-- 1 seed seed     16 Dec  1 13:29 p2.txt
-rw-rw-r-- 1 seed seed     32 Dec  1 13:29 p3.txt
-rw-rw-r-- 1 seed seed 184974 Dec  5  2020 pic_original.bmp
-rwxrw-r-- 1 seed seed    464 Jan  3  2021 sample_code.py
-rw-rw-r-- 1 seed seed 206662 Dec  5  2020 words.txt
[12/01/21]seed@VM:~/.../Files$ hexdump -C p1.txt
00000000  31 32 33 34 35 0b 0b 0b  0b 0b 0b 0b 0b 0b 0b 0b  |12345...........|
00000010
[12/01/21]seed@VM:~/.../Files$ xxd p1.txt
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b  12345...........
[12/01/21]seed@VM:~/.../Files$ hexdump -C p2.txt
00000000  30 31 32 33 34 35 36 37  38 39 06 06 06 06 06 06  |0123456789......|
00000010
[12/01/21]seed@VM:~/.../Files$ xxd p2.txt
00000000: 3031 3233 3435 3637 3839 0606 0606 0606  0123456789......
[12/01/21]seed@VM:~/.../Files$ hexdump -C p3.txt
00000000  30 31 32 33 34 35 36 37  38 39 31 32 33 34 35 36  |0123456789123456|
00000010  10 10 10 10 10 10 10 10  10 10 10 10 10 10 10 10  |................|
00000020
[12/01/21]seed@VM:~/.../Files$ xxd p3.txt
00000000: 3031 3233 3435 3637 3839 3132 3334 3536  0123456789123456
00000010: 1010 1010 1010 1010 1010 1010 1010 1010  ................
[12/01/21]seed@VM:~/.../Files$ █
```