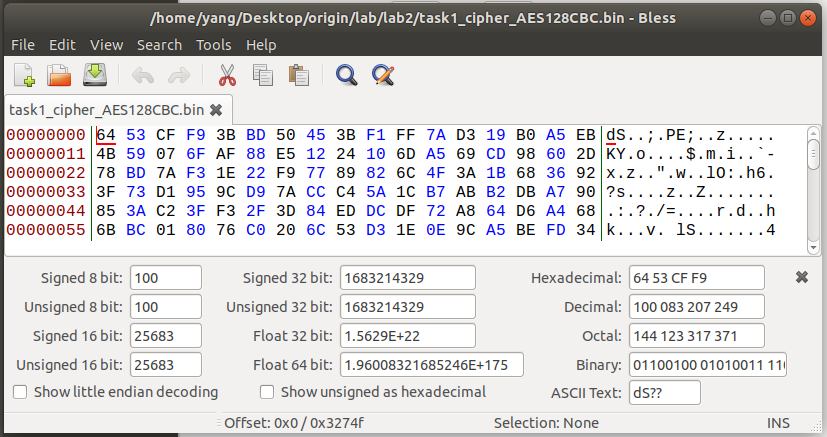
Task 2

In this task, we chose AES-128-CBC, BF-CFB, and DES to encrypt the file we made named “words.txt”.

AES-128-CBC

Firstly, we used AES-128-CBC to encrypt the file and input as “task1\_cipher\_ AES128CBC.bin”, set the key is 00112233445566778889aabbccddeeff and set iv is 01020304050607080910111213141516. Then we opened the encrypted file and can see that the content of file is unreadable.

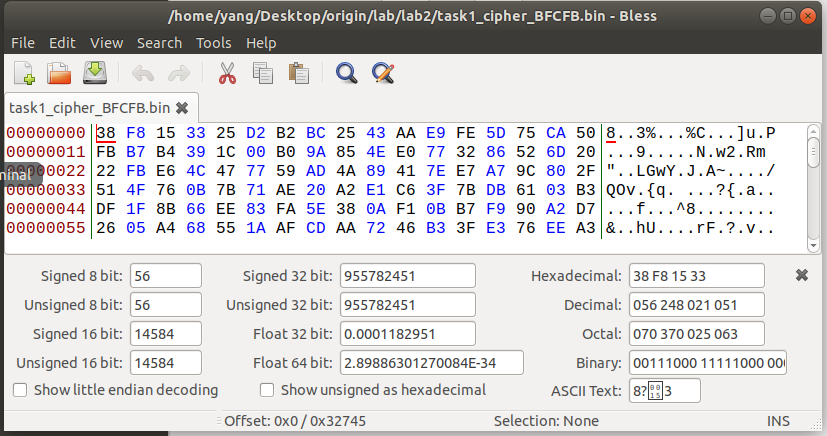




BF-CFB

Then, we tried the BF-CFB encryption , output is “task1\_cipher\_BFCFB.bin”, set key is 00112233445566778889aabbccddeeff and set iv is 0102030405060708. The file was successfully encrypted.

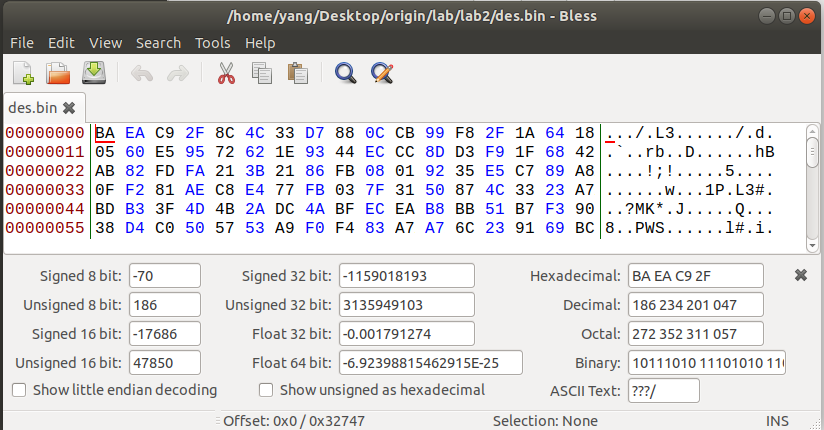




DES

Finally, we tried DES to encrypt the file, using key 0011223344556677 and iv 0102030405060708.





Task 4

1

First, we created a 7-byte file named “t4.txt”. use this file as material to try the encryption.



Then we encrypted the file with ECB, CBC, CFB, OFB respectively and get the output “t4ecb.bin”,”t4cbc.bin”,”t4cfb.bin”,”t4ofb.bin”.

The key is 00112233445566778899aabbccddeeff and the iv is 010203040506070809101112131415

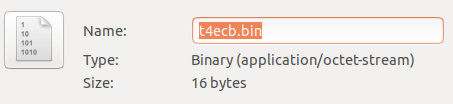


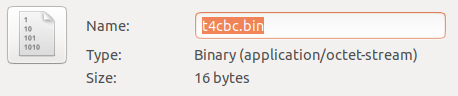


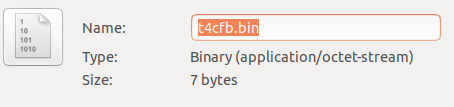


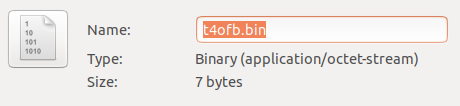


We can see that use ECB and CBC will have 9 bytes padding and use CFB and OFB will not have any padding.









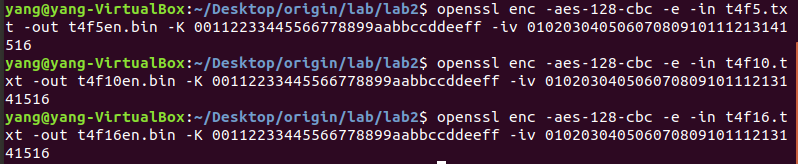
CFB and OFB mode are the mode that first get the encrypted iv and using the encrypted iv XOR the plaintext to encrypt the messages. That means that these two modes can encrypt any length of message. No padding can still encrypt the plaintext messages.

2

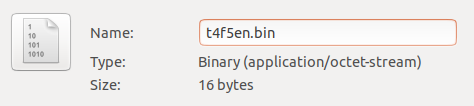
We created the files using “echo -n” command and we got 5-byte file “t4f5.txt”, 10-byte file “t4f10.txt”, and 16-byte file “t4f16.txt”

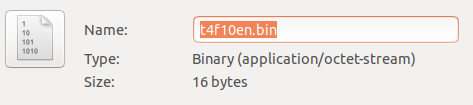


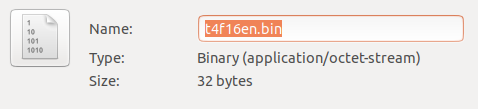
Then we used CBC to encrypt the files respectively and output are “t4f5.bin”, “t4f10.bin”, and “t4f16.bin”, using the same key 00112233445566778899aabbccddeeff and the same iv 01020304050607080910111213141516.



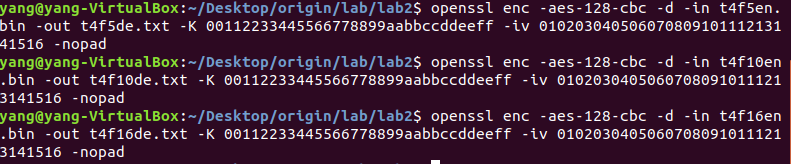
Then we can see that all three files have padding. The 5-byte file and 10-byte became 16-byte, and 16-byte file became 32-byte.





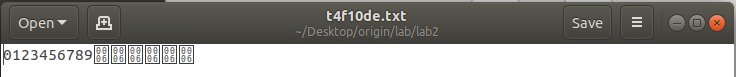


Then we decrypted the files and the output are “t4f5de.bin”, “t4f10de.bin” and “t4f16de.bin”.



Open the file we can see these results.







Then we use “hexdump” to open the files. The content of file are displayed like this.

