Jinwoo Yom

Jinwoo7@vt.edu

CS 4264, Project 3

“As a Hokie, I will conduct myself with honor and integrity at all times. I will not lie, cheat, or steal, nor will I accept the actions of those who do.”

Project 3: Crypto Lab – Secret-Key Encryption

Jinwoo Yom

– Section 1 – Objectives: Provide a summary of the project objectives.

– Section 2 – Task 1: Using your hex editor, include screenshots of your plain.txt file, as well as screenshots (with labels so it is clear what encryption cipher and mode is being used) for each cipher/mode combination. Include text discussion to explain what you did.

Plain text before Encryption



Figure . Output for plain text before the encrpytion

The text says: I’m late. I’m late. For a very important date. No time to say, "Hello, Goodbye." I’m late, I’m late, I’m late.

Cipher text after Encryption (with leafpad text editor and bless hex editor output)

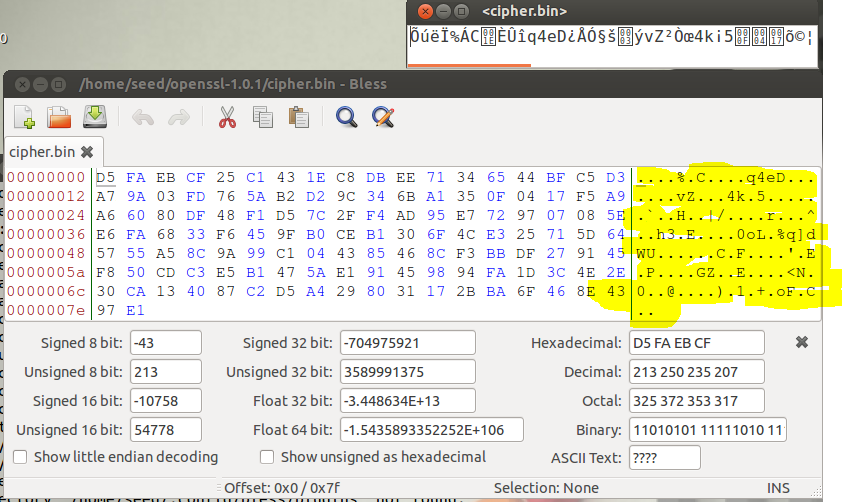


Figure .Output for aes-128-cbc encryption

Command used for aes-128-cbc encryption:

sudo openssl enc -aes-128-cbc -e -in plain.txt -out cipher.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708

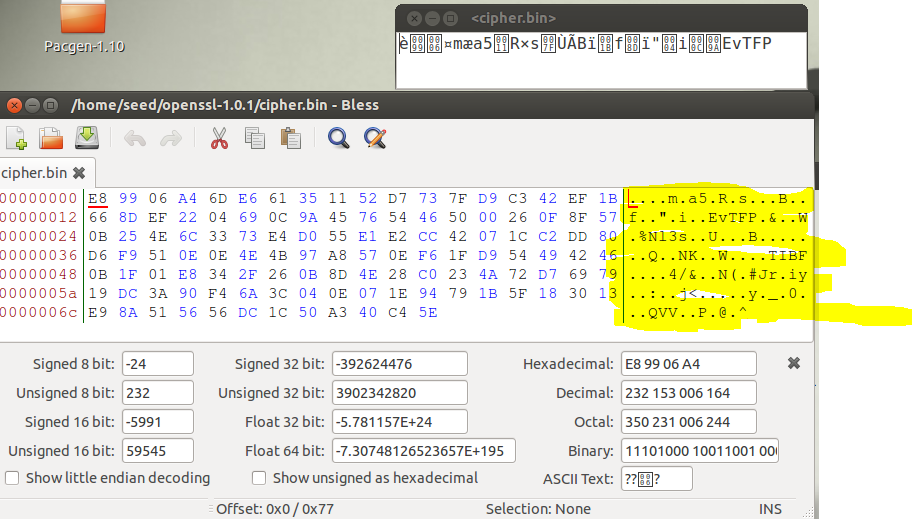


Figure . Output for aes-128-cfb encryption

Command used for aes-128-cfb encryption:

sudo openssl enc -aes-128-cfb -e -in plain.txt -out cipher.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708

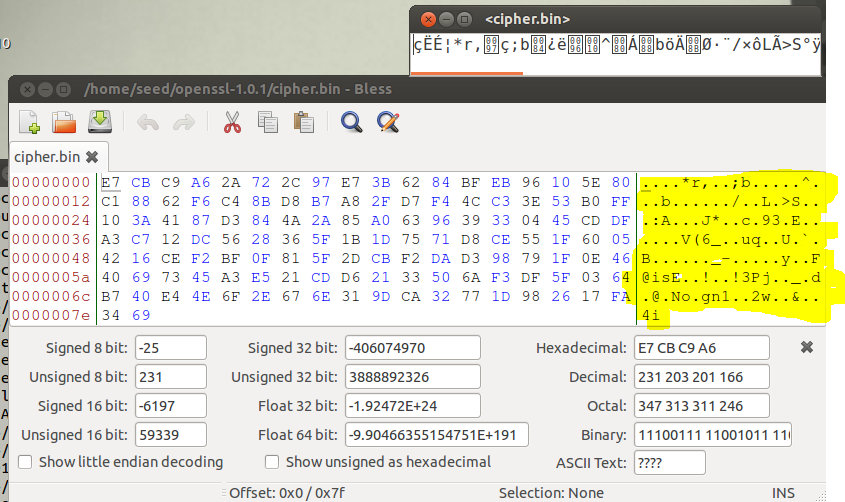


Figure . Output for bf-cbc encryption

Command used for bf-cbc encryption:

sudo openssl enc bf-cbc -e -in plain.txt -out cipher.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708

– Section 3 – Task 2: Take a screenshot of the original image displayed in an image viewing tool (eg Eye of Gnome, eog). Also include screenshots of your encrypted images. Can you derive any useful information about the original picture from the encrypted picture? Please explain your observations, especially noting differences in the image encrypted with ECB and CBC.

In this Task, a picture, pic\_original.bmp, is encrypted using ECB and CBC modes. To do this, You should first encrypt the original picture using the openssl command and save it as a separate bmp file. Once the new file is created, you will notice that it is unable to open. This is due to the new picture having corrupted header. To fix this issue, you must use a hex editor application (ex. ghex or bless) to copy the header (first 54 bytes) over from the original picture to the encrypted picture. Below are the results of this.

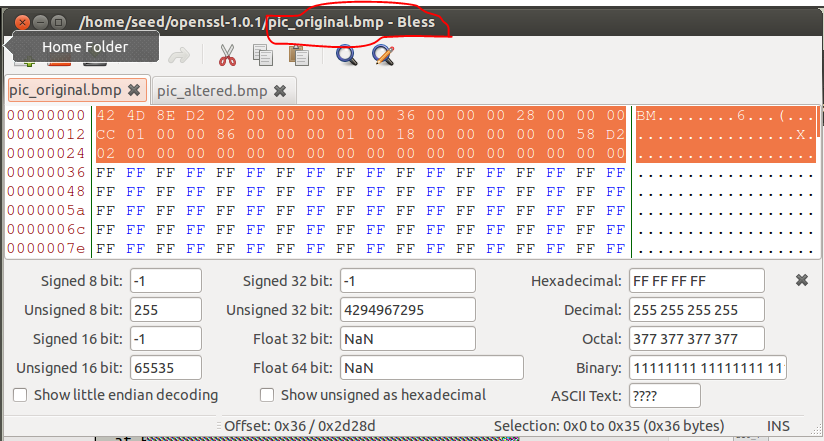


Figure . Hex values for the pic\_original.bmp

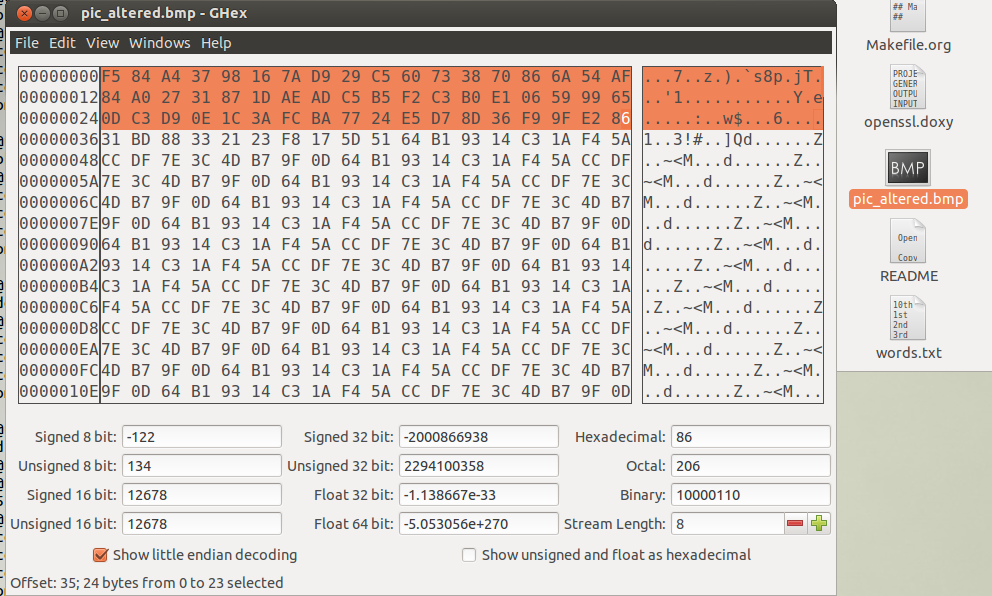


Figure . Modified Hex valves for the pic\_altered.bmp

Original Picture:

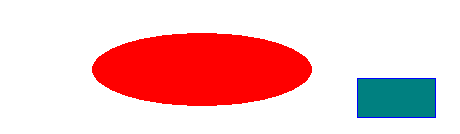


Figure . pic\_riginal.bmp picture

Command used for aes-128-ecb encryption:

sudo openssl enc -aes-128-ecb -e -in pic\_original.bmp -out pic\_altered.bmp -K 1001011 -iv 0010011

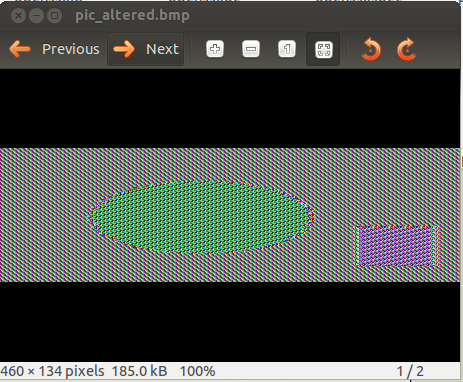


Figure . pic\_altered.bmp (ecb encrypted)

Command used for aes-128-cbc encryption:

sudo openssl enc -aes-128-cbc -e -in pic\_original.bmp -out pic\_altered.bmp -K 1001011 -iv 0010011

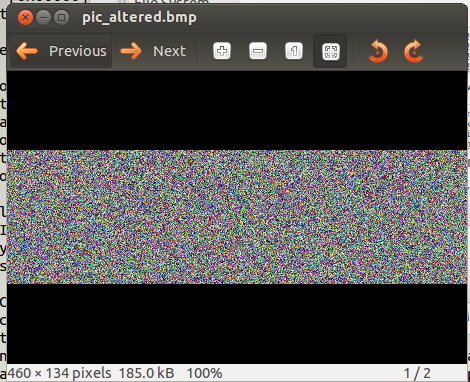


Figure . pic\_altered.bmp (cbc encrypted)

– Section 4 – Task 3: Using your hex editor, include a screenshot of your text file. Take a screenshot of your encrypted file, highlighting the 30th byte. Now take a screenshot after you have corrupted the 30th byte of the encrypted file. Encrypt the corrupted file using each of the requested modes – include screenshots of the encryption commands. Include screenshots of the commands used to decrypt the resulting encryption file. Compare the original plain text file with the decrypted results, noting differences or similarities based on the encryption mode used, and explain what you did.

In this Task, a text file of 97 bytes is created. Then it is encrypted with one of the following encryption modes: ecb, cbc, cfb, and ofb. Once encrypted, 1 bit in the 30th byte is corrupted using a bless (hex editor application). The corrupted file is then decrypted back to the text file and compared to the original text file.

Original text file:

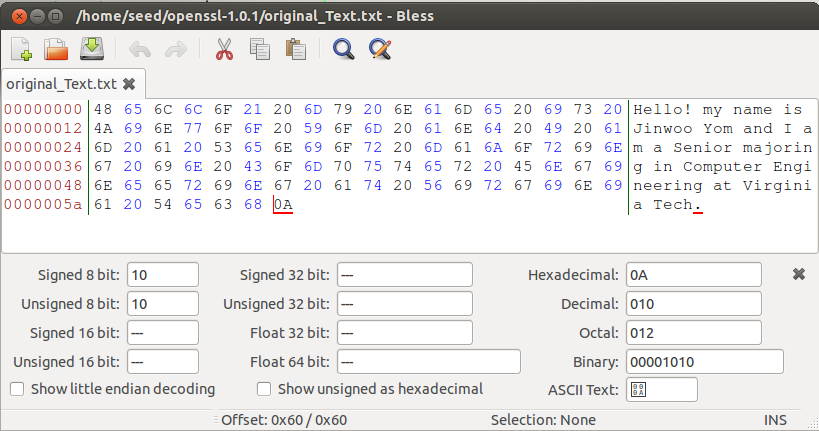


Figure . Original text file

* ECB

Encryption Command used:

C:\Users\Jinwoo\AppData\Local\Microsoft\Windows\INetCacheContent.Word\ecb_encrypt_command.png

Figure . Encryption command for ECB

Encrypted:

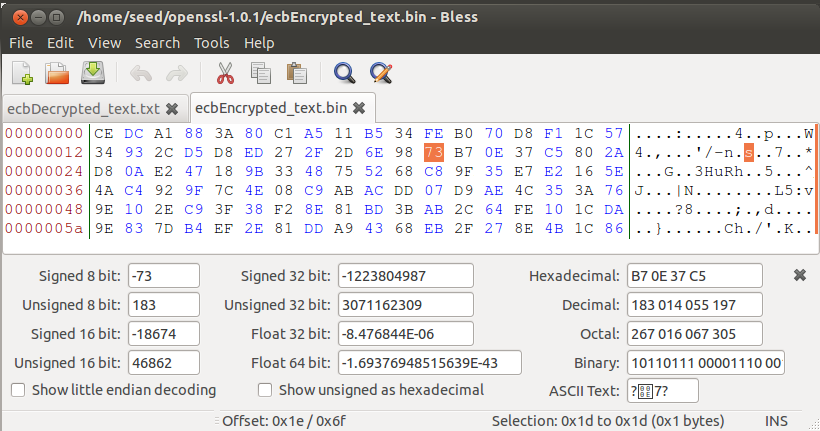


Figure . ECB encrypted text

Corrupted:

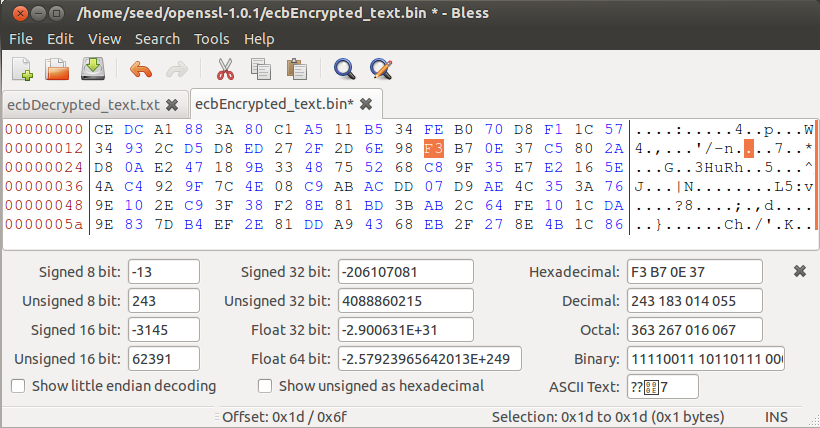


Figure . Corrupted ECB encryption

Decryption Command used:



Figure . ECB decryption command

Decrypted:

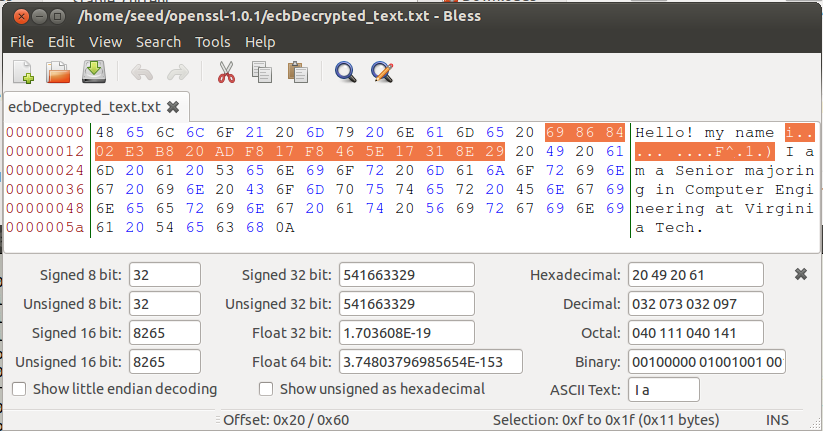


Figure . ECB decrypted text

* CBC

Encryption Command used:



Figure . Encryption command for CBC

Encrypted:

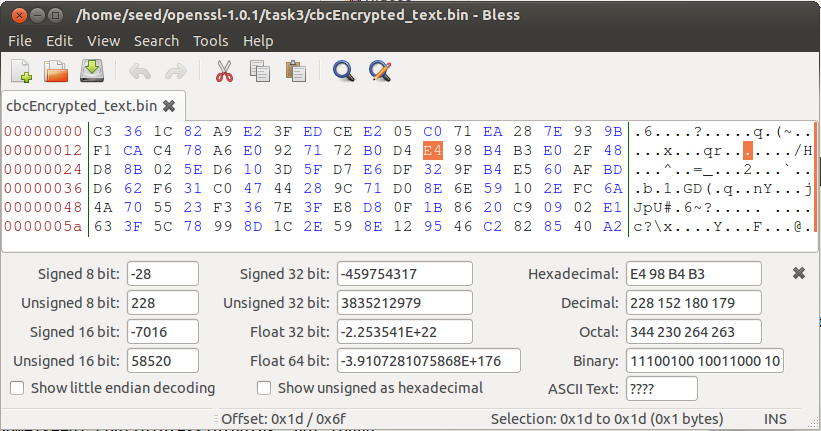


Figure . CBC encrypted text

Corrupted:

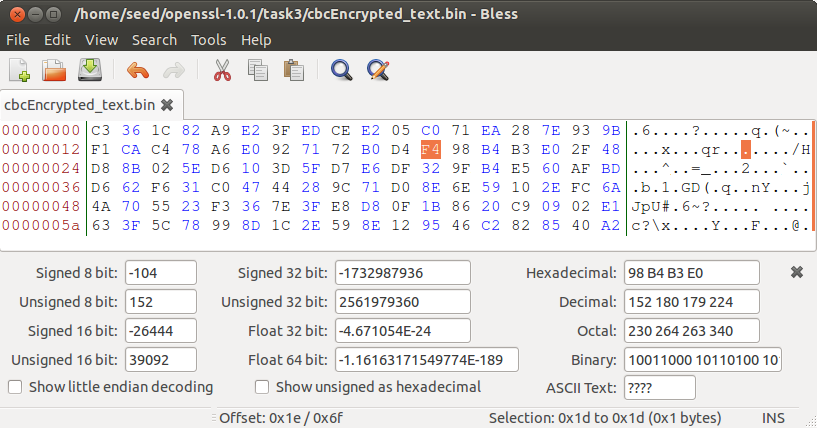


Figure . Corrupted CBC encryption

Decryption Command used:



Figure . CBC decryption command

Decrypted:

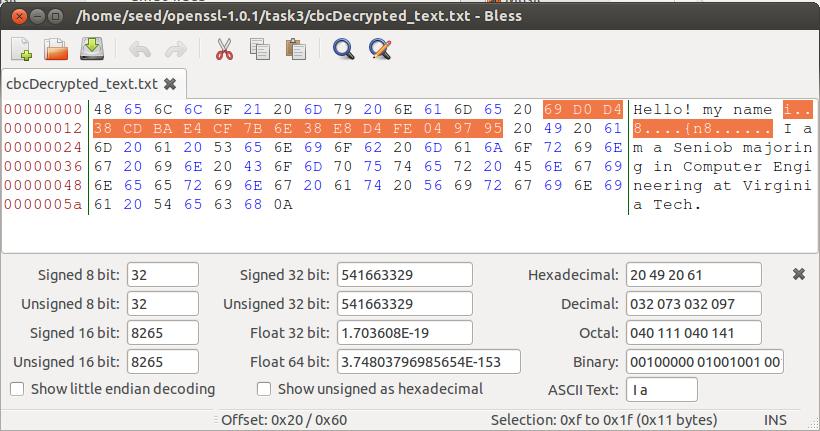


Figure . CBC decrypted text

* CFB

Encryption Command used:



Figure . Encryption command for CFB

Encrypted:

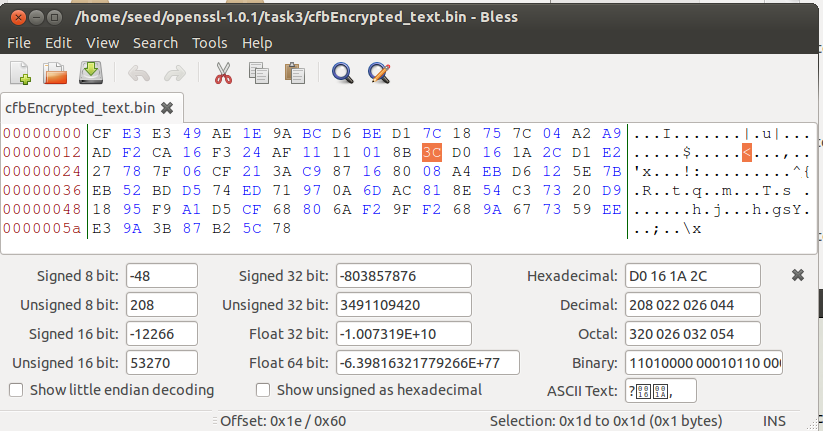


Figure . CFB encrypted text

Corrupted:

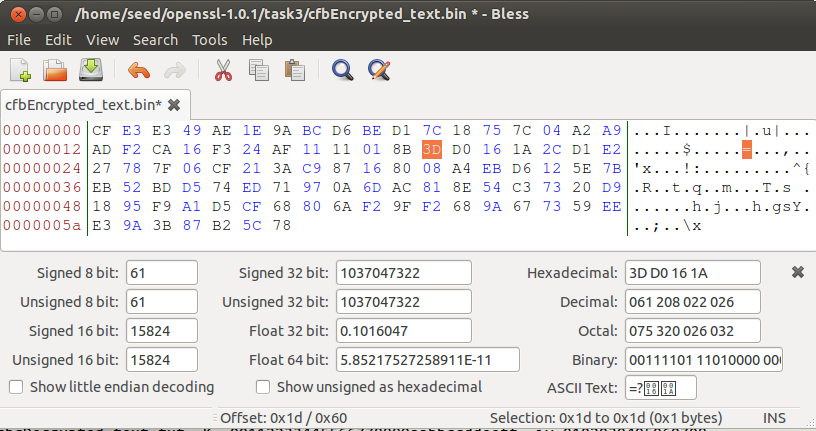


Figure . Corrupted CFB encryption

Decryption Command used:



Figure . CFB decryption command

Decrypted:

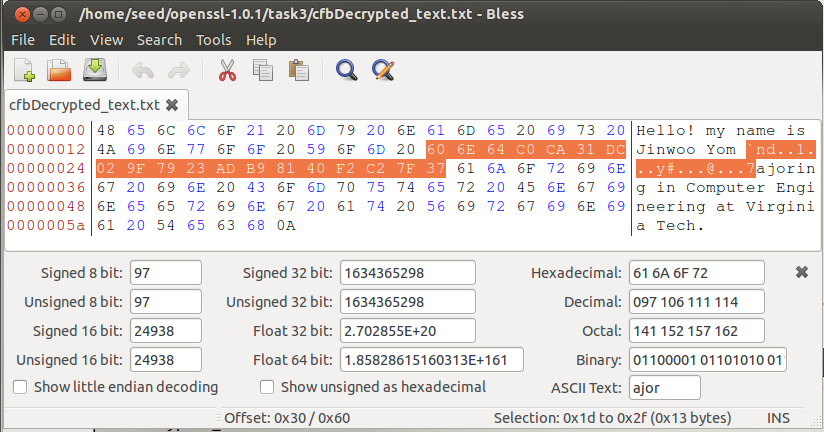


Figure . CFB decrypted text

* OFB

Encryption Command used:



Figure . Encryption command for OFB

Encrypted:

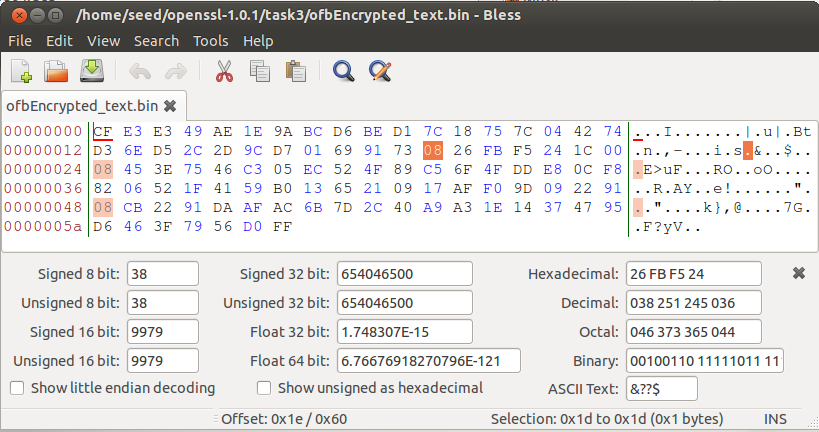


Figure . OFB encrypted text

Corrupted:

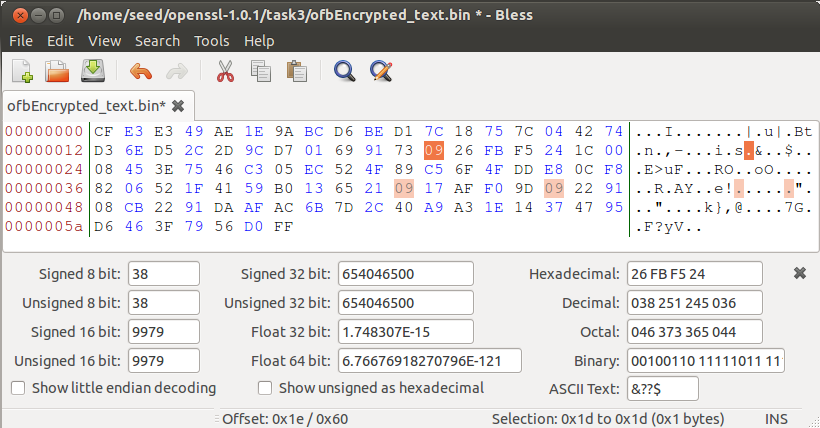


Figure . Corrupted OFB encryption

Decryption Command used:



Figure . OFB decryption command

Decrypted:

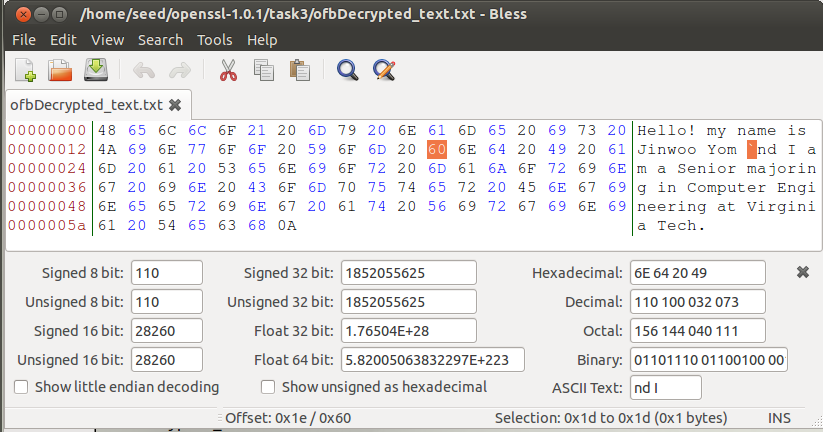


Figure . OFB decrypted text

After comparisons, I noticed that corrupting a bit for cbc and ecb affected 17 bytes of the text. Similarly, corrupting a bit for cfb affected 19 bytes of the text. These tells me that the encriptions depends on the surrounding bit to encrypt and decrypt. In ofb encryption, However, corrupting a bit affected only one byte which tells me that the encryption does not depend on the surrounding bytes.

– Section 5 – Task 4: Using your hex editor, take screenshots of your test text files, screenshots of your encryption commands, and screenshots of the encrypted results. You can edit (draw on, highlight, etc) the files to illustrate your experimental results. Include text discussions to explain what you did.

– Section 6 – Task 5: Include your program’s source code and provide screenshots of sample output, as well as a screenshot showing the key your program found, and why you think that is they key.

– Section 7 – Task 6: Task 6.A: Document and describe your experiment with entropy avail. Task 6.B: Document how you can get /dev/random to unblock and to print out random data. Task 6.C: Run urandom several times, and document whether it will block or not.

– Section 8 – Conclusions: Briefly discuss the outcome of your experiments and any problems or aspects that do not work properly; what you learned by doing this project; and any experiences that were particularly good or bad. Also, specify the approximate number of hours that you devoted to the project. (The number of hours is just for the instructor to assess the suitability of this project assignment.)