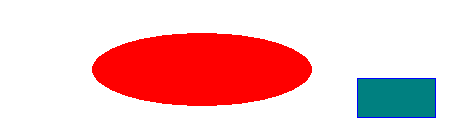
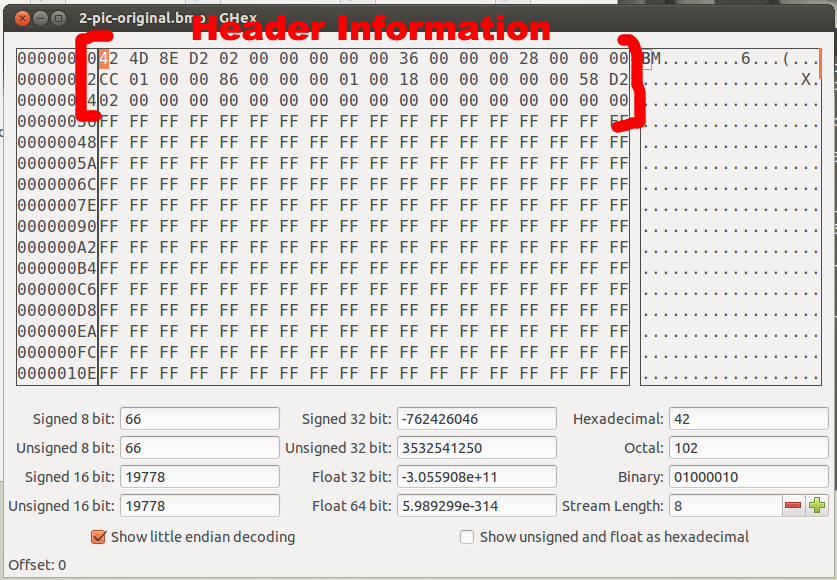
**TASK 2:**

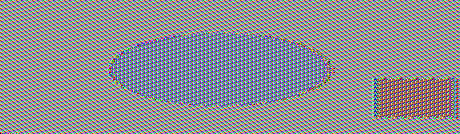
1. 2-pic-original.bmp contains a simple picture that I would like to encrypt, so people that I don't know cannot see the original image. Here's what it looks like: 
2. This time, I'll encrypt using AES-128-ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes.

**AES-128-ECB**

Our goal is to create an encrypted, but also a viewable .bmp picture. The first thing we should try is to get rid of header information that resides in .bmp file. Also this information need to be erased as it does not relate to generating the picture.

* Utilize ghex to cut unrelated bytes in the file and save the file and save the bytes into your clipboard/notes/etc. 
* I saved mine as 2-pic-altered.bmp
  + Encrypt 2-pic-altered.bmp using the command used. I saved mine as 2-pic-aes-128-ecb
  + You can't open and view the encrypted picture as a .bmp yet! To solve this problem, we have to give the encryption its header encryption (the info we cut out into the clip-board).
  + Open 2-pic-aes-128-ecb.bmp in ghex and insert the header information on the first three memory locations (0000000000, 00000002, 00000004).
  + That's it, the image should look like this:

**ECB**



**CBC**

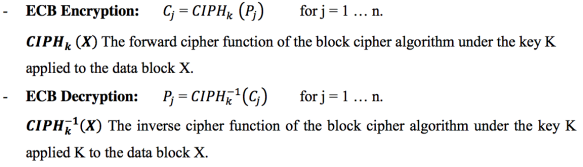


Cool!

**3.  Encryption mode – ECB vs. CBC**

**Encryption – ECB**

The Electronic Codebook (ECB) mode is a confidentiality mode that features the message is divided into blocks, and each block is encrypted separately. The Electronic Codebook (ECB) mode is defined as follows (Evans, Bond, & Bement, 2001):

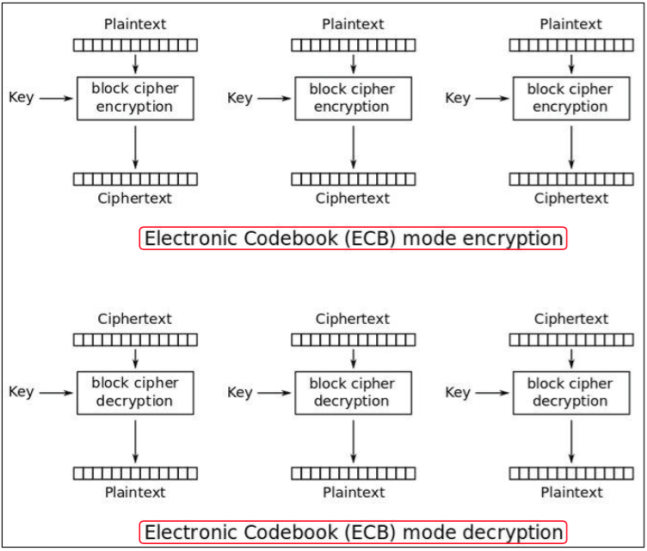
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-41-47.png)

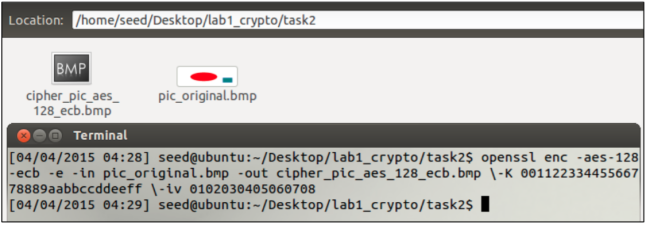
In ECB encryption, the forward cipher function is applied directly and independently to each block of the plaintext. The resulting sequence of output blocks is the cipher text.

In ECB decryption, the inverse cipher function is applied directly and independently to each block of the cipher text. The resulting sequence of output blocks is the plaintext.

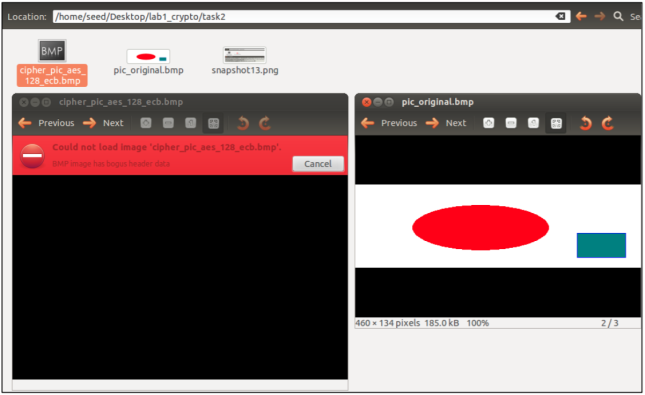
With ECB, if the same b-bit block of plaintext appears more than once in the message, it always produces the same cipher text. Because of this, for lengthy messages, the ECB mode may not be secure.

The figure below show both encryption and decryption processes of ECB (Block cipher mode of operation, 2015).

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-43-38.png)

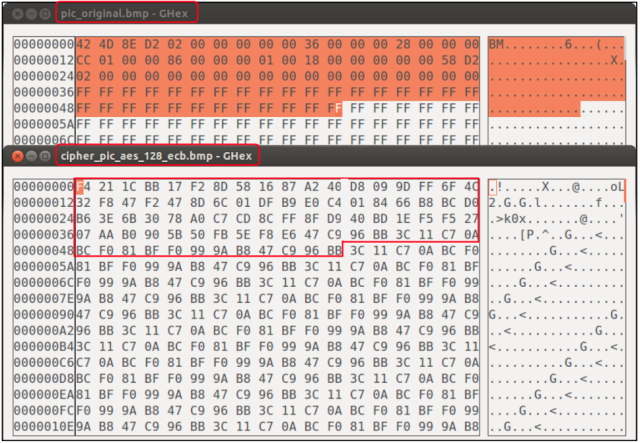
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-44-18.png)

I use the image provided by the website of the lab, pic\_original.bmp. I then encrypted the image using the electronic codebook encryption mode.

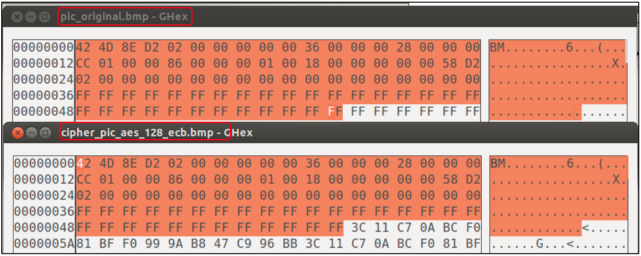
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-45-02.png)

When I try to view the encrypted image using the software to display it, it show error “BMP image has bogus header data” because the .bmp file, the first 54 bytes contain the header information about the picture, I have to set it correctly, so the encrypted file can be treated as a legitimate .bmp file. I use ghex editor tool to change those 54 bytes.

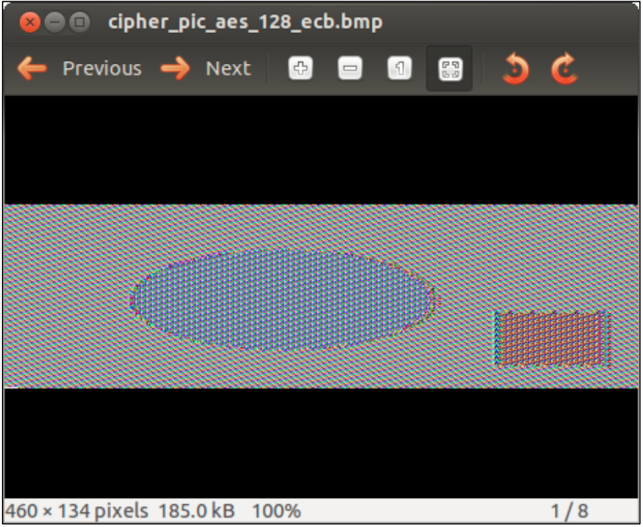
I opened two files, pic\_original.bmp and cipher\_pic\_aes\_128\_ecb.bmp of images with ghex. The selected areas in the pic\_original.bmp are the 54 bytes values that we need to put in the header of cipher\_pic\_aes\_128\_ecb.bmp; I also drew a rectangle around the 54 bytes that need to change.

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-45-50.png)

Here is the bytes that I changed:

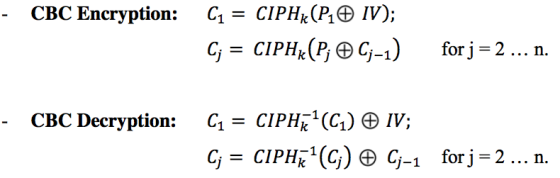
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-46-00.png)

After changed the header of the cipher\_pic\_aes\_128\_ecb.bmp, I can view this file with image software.

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-47-30.png)

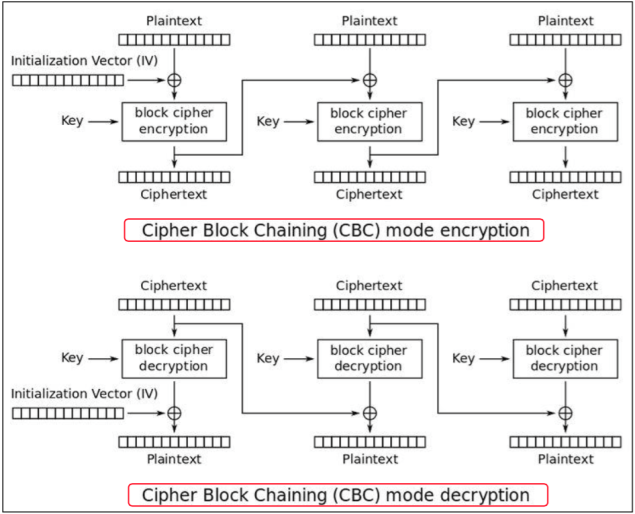
**Encryption – CBC**

The Cipher Block Chaining (CBC) mode is a confidentiality mode whose encryption process features the combining (“chaining”) of the plaintext blocks with the previous ciphertext blocks. The CBC mode requires an IV to combine with the first plaintext block. The IV need not be secret, but it must be unpredictable; The CBC mode is defined as follows (Evans, Bond, & Bement, 2001):

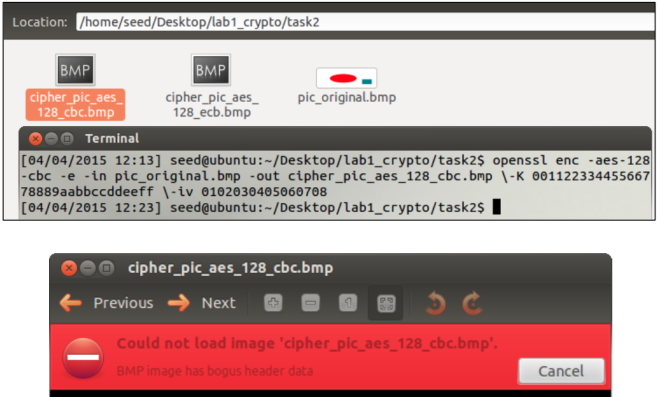
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-48-28.png)

In CBC mode, each block of plaintext is XORed with the previous cipher text block before being encrypted. This way, each cipher text block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.

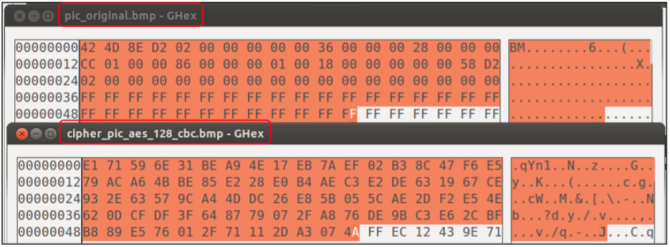
The figure below show both encryption and decryption processes of CBC (Block cipher mode of operation, 2015)

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-49-13.png)

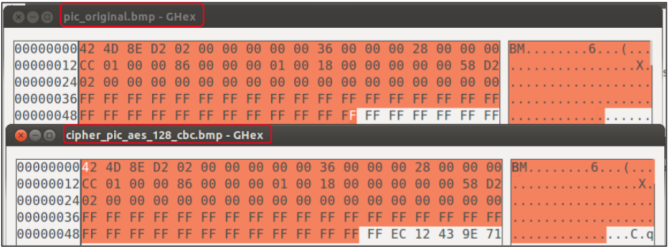
I encrypted the pic\_original.bmp with the encryption mode cbc.

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-49-59.png)

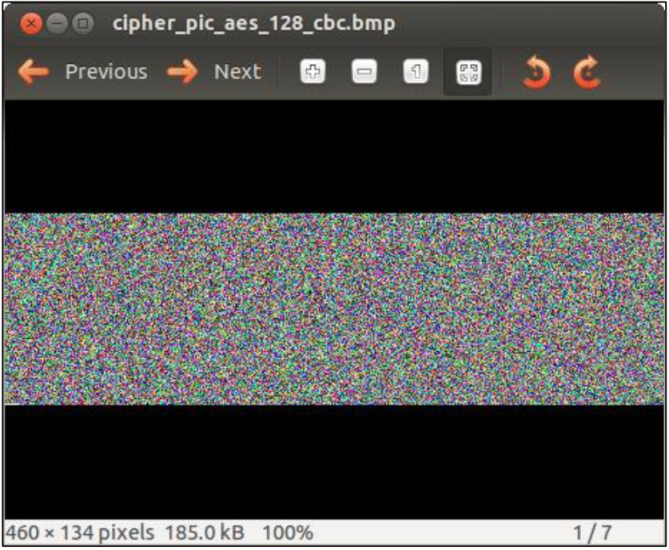
In order to be able to view the image, I have to change the 54 bytes header of the encrypted image, cipher\_pic\_aes\_128\_cbc.bmp.

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-50-57.png)

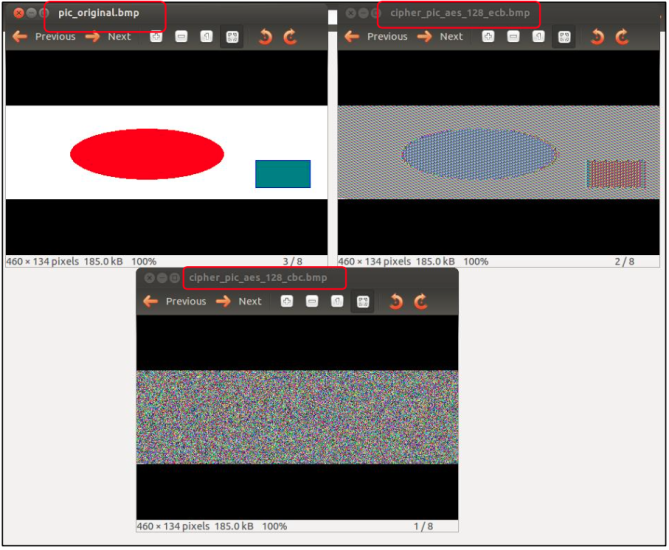
Here is the contents of both files after I changed the header bytes of the encrypted image.

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-51-07.png)

Here is the encrypted image viewed by the image software viewer.

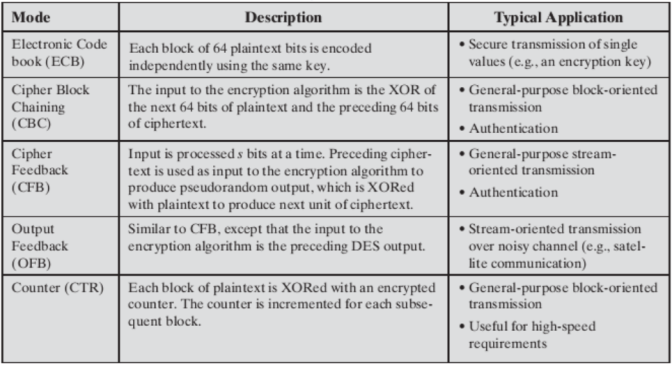
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-52-09.png)

**Observation**

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-52-24.png)

The disadvantage of ECB method is that when identical plaintext blocks are encrypted into identical cipher text blocks; thus, data patterns are not well hidden. So no confidentiality is provided. On the other hand, CBC provides more secure mechanism because it used the operation XOR before encryption of each cipher text block.

Here is the brief description of each encryption mode (William & Brown, 2012):

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-28-at-22-53-45.png)

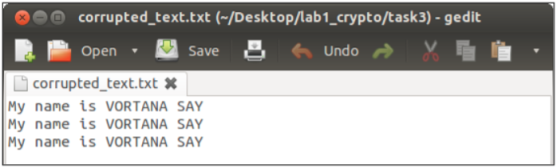
In this part, we discussed mainly the different encryption methods. In part 2, I am going to discuss the next tasks which are , Encryption Mode – Corrupted Cipher Text, Padding, and Programming using the Crypto Library.

# Crypto Lab – Secret-Key Encryption (Part 2)

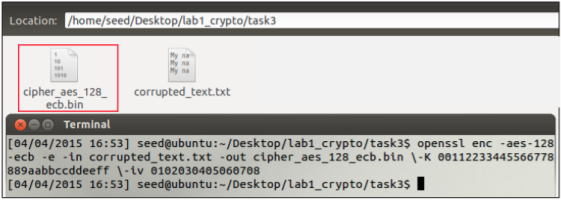
[May 30, 2015](https://vortana.wordpress.com/2015/05/30/crypto-lab-secret-key-encryption-part-2/) [vortana say](https://vortana.wordpress.com/author/vortanasay/)[Leave a comment](https://vortana.wordpress.com/2015/05/30/crypto-lab-secret-key-encryption-part-2/#respond)

In the previous part, I showed the first two big parts of the crypto lab of the SEED Lab. In this part, I am going to discuss the rest of the tasks in the labs.

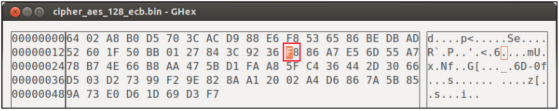
**3.  Encryption Mode – Corrupted Cipher Text**

  
I created a plain text file whose size is 69 bytes. I will encrypt this file using AES-128 and encryption mode, ECB, CBC, CFB, and OFB respectively. After that I will change a single bit of the 30th byte, 1E in hexadecimal value, so that I can get the corrupted encrypted file. Then I will decrypt the corrupted encrypted file using its encryption mode and explain the differences of the results.

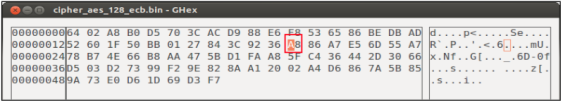
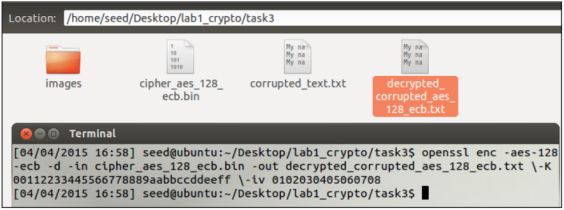
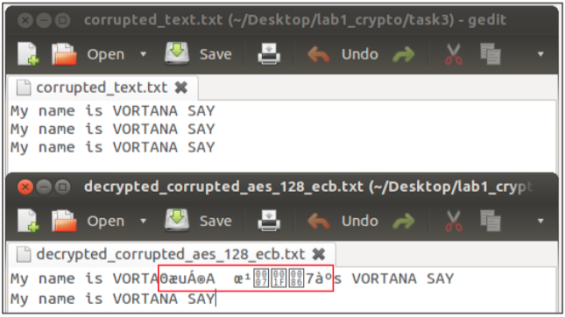
**Encryption mode – ECB**

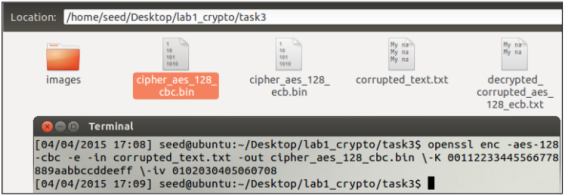
I encrypted the plain text, called corrupted\_text.txt.  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-33-13.png)

* Original encrypted file

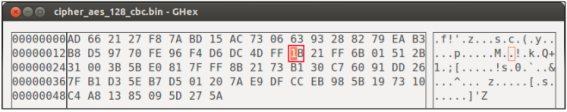
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-36-16.png)

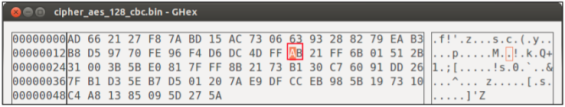
* Corrupted encrypted file

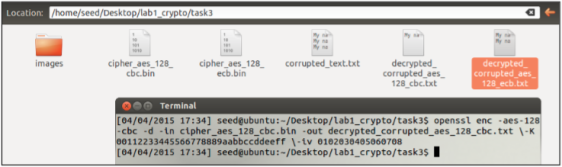
  
In the original encrypted file, the position 30th byte is 1E corresponds to the value F8, I changed the single bit of this value from F to A so the value of 30th byte now is A8 then I saved the file. Consequently, I got a corrupted encrypted file.  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-37-58.png)  
I decrypted the encrypted file, cipher\_aes\_128\_ecb.bin which is became corrupted. I received a decrypted file, decrypted\_corrupted\_aes\_128\_ecb.txt.  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-38-58.png)

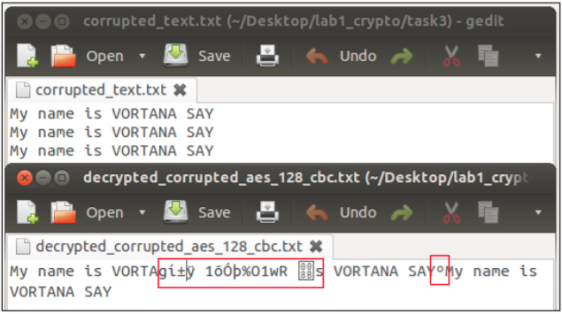
**Encryption mode – CBC**  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-39-37.png)

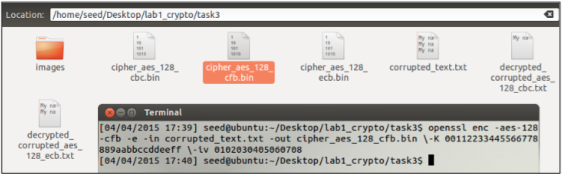
Original encrypted file

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-40-57.png)

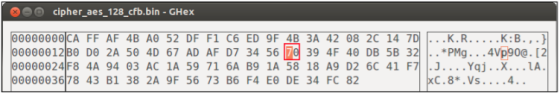
Corrupted encrypted file  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-41-35.png)

In the original encrypted file, the position 30th byte is 1E corresponds to the value 1B, I changed the single bit of this value from 1 to A so the value of 30th byte now is AB then I saved the file. Consequently, I got a corrupted encrypted file.  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-42-10.png)

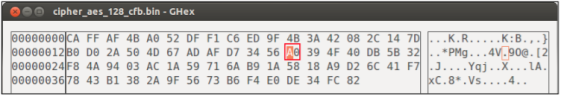
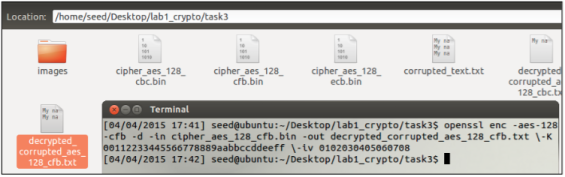
Here is the result:  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-42-47.png)

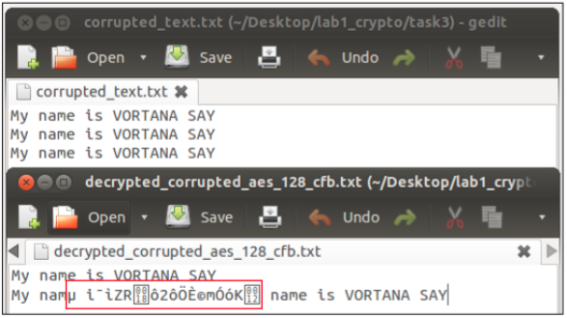
**Encryption mode – CFB**  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-43-35.png)

* Original encrypted file

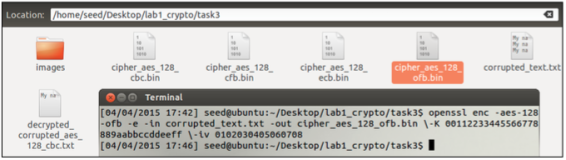
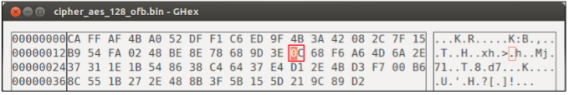
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-44-20.png)

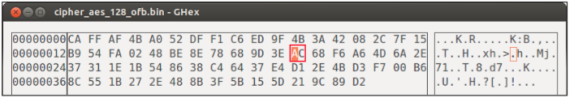
* Corrupted encrypted file

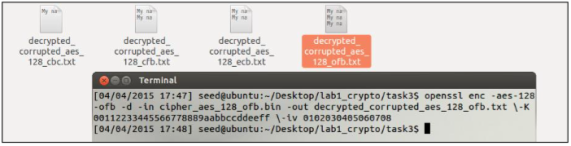
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-45-19.png)  
In the original encrypted file, the position 30th byte is 1E corresponds to the value 70, I changed the single bit of this value from 7 to A so the value of 30th byte now is A0 then I saved the file. Consequently, I got a corrupted encrypted file.  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-46-06.png)

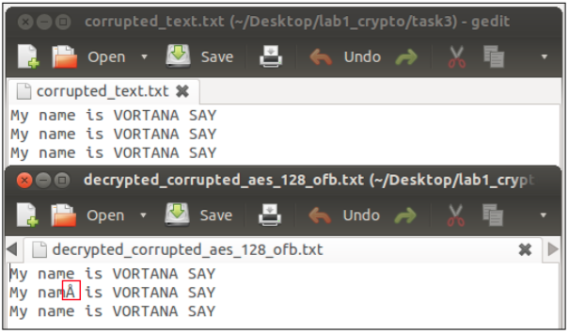
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-46-39.png)

**Encryption mode – OFB**

[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-47-17.png)  
Original encrypted file  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-47-33.png)

Corrupted encrypted file  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-48-42.png)

In the original encrypted file, the position 30th byte is 1E corresponds to the value 0C, I changed the single bit of this value from 0 to A so the value of 30th byte now is AC then I saved the file. Consequently, I got a corrupted encrypted file.  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-49-21.png)

Here is the result:  
[](https://vortana.files.wordpress.com/2015/05/screen-shot-2015-05-29-at-22-49-30.png)

**Conclusion**  
From the results above I can conclude that:

* In ECB mode, only one block is affected when any problem in a cipher text happens; furthermore; moreover, each block is decrypted independently. However, the corrupted bit of the 30th byte in cipher text block 8 bytes might spread to all n bits in plaintext block 8 bytes since we do the decryption one block at a time.
* In CBC mode, there was affect in two blocks.
* In CFB mode, there is problem in n / r number of blocks. Therefore, the error propagation criterion is poorer (Modes of Operation of Block Ciphers).
* In OFB mode, the feedback is only in the key-generation system. If the single digit of the 30th byte corrupted, then in plain text that only that byte or character is corrupted.**Thus, only OFB mode shows the most promising result and almost all the texts are recovered.**