CSCE 548

Project 1

Group 9

Member: Andrew Cox, Joy Ray, Sebastian Martin, Xiyuan Zheng

Readme file

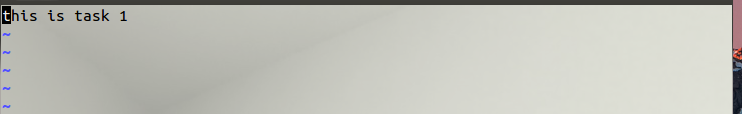
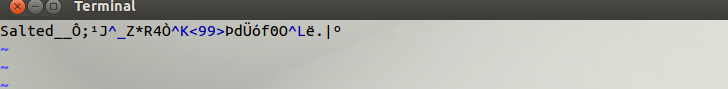
Task 1: Encryption using OpenSSL commands with different cipher and mode options Mode 1 using CBC

1. Mode using CBC
2. Encryption

* **openssl enc -des-ede-cbc -e -in plain.txt -out cipherbin.bin -k password**

this command encrypts a text file with CBC mode and with a password

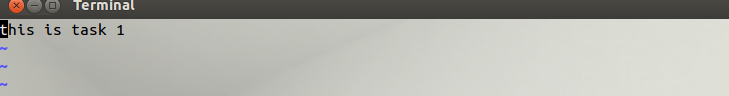
this is the plain text file:

this is what the encrypted file looks like: 

1. Decryption

* **openssl enc -des-ede-cbc -d -in cipherbin.bin -out cipherout.txt -k password**

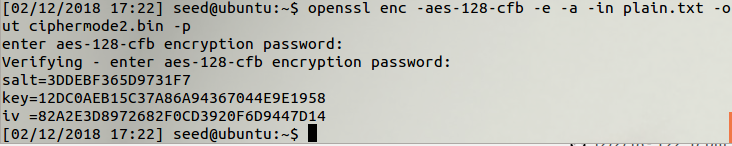
this command decrypts bin file with CBC mode and passed in the password

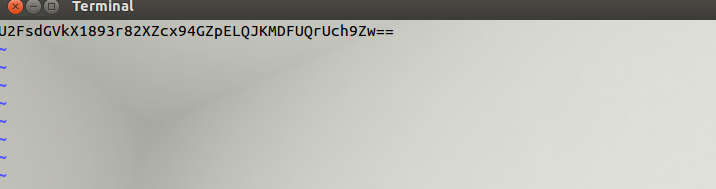
this is what decrypted file looks like:

1. mode 2 using CFB
2. Encryption

* **openssl enc -aes-128-cfb -e -a -in plain.txt -out ciphermode2.bin -p**

this command encrypt the same plain text file with CFB mode, output will be base64 encoded and will print out salt key and iv. Then it will ask and verify the password you create

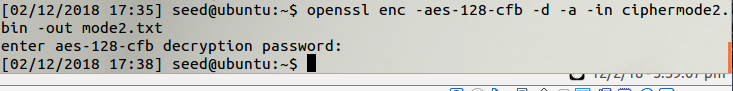


Encrypted file looks like: 

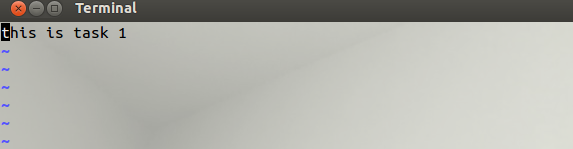
1. Decryption

* **openssl enc -aes-128-cfb -d -a -in ciphermode2.bin -out mode2.txt**

This command will ask you the password you created for encryption



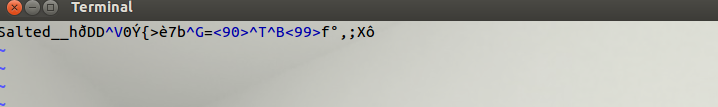
Decrypted file looks like:



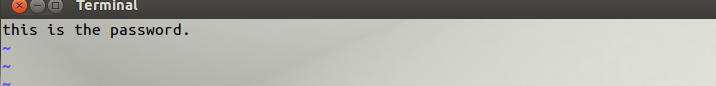
1. Mode 3 using ECB
2. Encryption

* **openssl enc -bf-ecb -salt -in plain.txt -out ciphermode3.bin -pass file:password.txt**

this command using ECB mode, pass in a password that created in password.txt with salt option that stronger the encryption

this is encrypted file looks like:

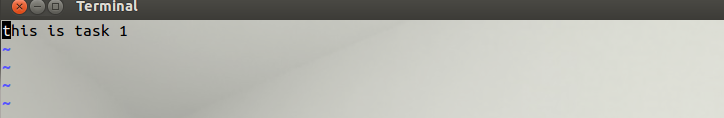
password.txt:



1. Decryption

* **openssl enc -bf-ecb -d -in ciphermode3.bin -out mode3.txt -pass file:password.txt**

this is decrypted file looks like:



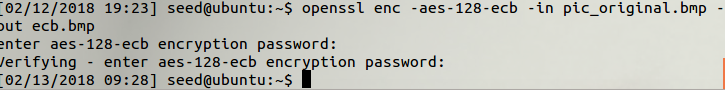
Task 2: Encryption Mode – ECB vs. CBC

For this task, download pic\_original.bmp for testing.

1. ECB mode
2. Encryption

* **openssl enc -aes-128-ecb -in pic\_original.bmp -out ecb.bmp**

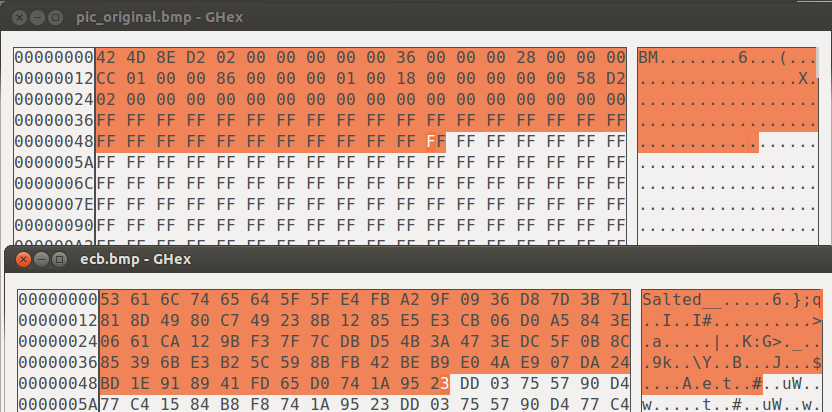
this command uses AES-128 cipher, will ask to create a password.



because its header is bogus, have to use hex editor to replace the first 54 bytes with that of the original picture pic\_original.bmp

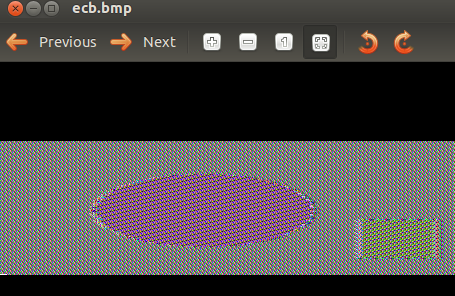
should open up both original and encrypted hex editor by using **ghex pic\_original.bmp** and **ghex ecb.bmp**

the first 54 bytes were highlighted.



1. Display image

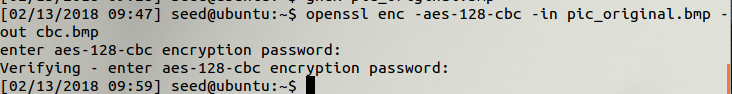
Should have something like this:



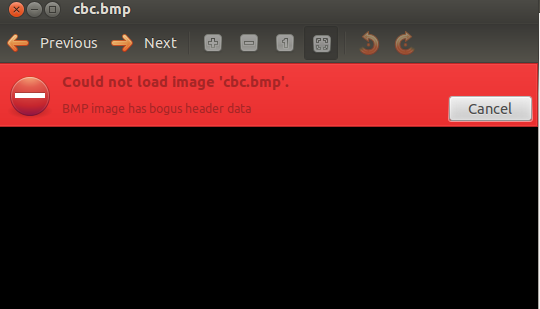
1. Mode CBC
2. Encryption

* **openssl enc -aes-128-cbc -in pic\_original.bmp -out cbc.bmp**

this command uses AES-128 cipher, will ask to create a password.

****

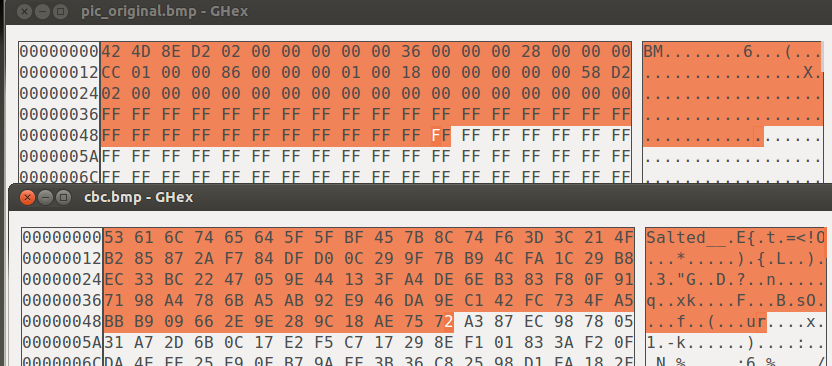
because its header is bogus, if try to display it, will get something like this:



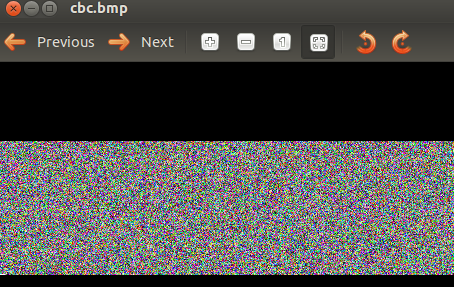
have to use hex editor to replace the first 54 bytes with that of the original picture pic\_original.bmp

should open up both original and encrypted hex editor by using **ghex pic\_original.bmp** and **ghex cbc.bmp**

the first 54 bytes were highlighted



1. Display image

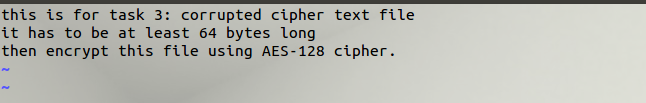
Should have something like this:

Task 3: Encrypted Mode -- Corrupted Cipher Text

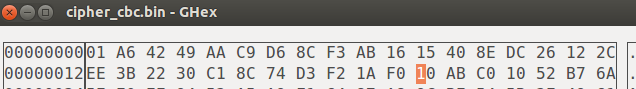
For this task, need to create a at least a 64-byte file, use AES -128 cipher, then change a single bit of the 30th byte to make a corrupted file

1. CBC mode
2. Encryption

* **openssl enc -aes-128-cbc -e -in corrputed.txt -out cipher\_cbc.bin \-K 0011223344556677889aabbccddeeff \-iv 0102030405060708**

This is the original file:

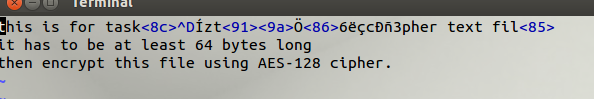
Open hex editor to change the bit, using command **ghex cipher\_cbc.bin**

The 30th byte was 10

Then change it into F0

1. Decryption

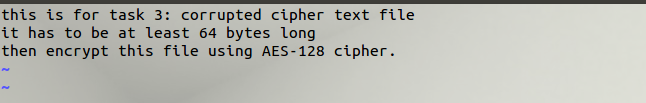
* **openssl enc -aes-128-cbc -d -in cipher\_cbc.bin -out decrypt\_cbc.txt \-K 0011223344556677889aabbccddeeff \-iv 0102030405060708**

Then decrypted, to show the contents of the corrupted file:

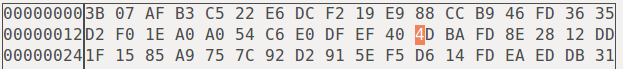
Observation: two places haven’t corrupted.

1. ECB mode
2. Encryption

* **openssl enc -aes-128-ecb -e -in corrupted.txt -out cipher\_ecb.bin \-K 0011223344556677889aabbccddeeff \-iv 0102030405060708**

This is the original file:

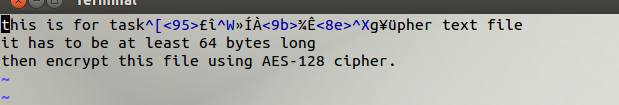
Open hex editor to change the bit, using command **ghex cipher\_ecb.bin**

ECB mode, original file, its 30th byte was 4D.

Then change it into AD

1. Decryption

* **openssl enc -aes-128-ecb -d -in cipher\_ecb.bin \-K 0011223344556677889aabbccddeeff \-iv 0102030405060708**

the content of corrupted file:

Observation: one place has been corrupted.