Assignment 3: Secret Key Crypto
Computer Security
Marco Alanis
10/29/2023

Introduction

Assignment 3 of CS 4351: Computer Security, taught by Dr. Deepak Tosh, aimed to provide students with a hands-on experience in secret-key encryption, exploring various encryption algorithms, modes, and their behavior when decrypting data. This assignment also involved introducing errors into the encrypted data to observe how different encryption modes handle decryption when the data has been altered.

Task 1: Encryption using Different Ciphers and Modes

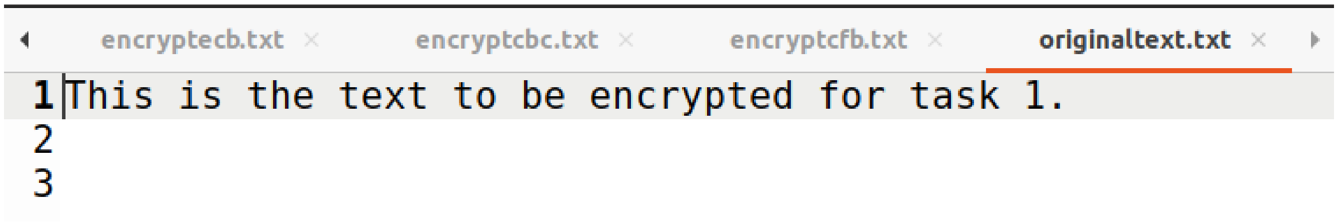
In Task 1, we experimented with three different encryption modes: ECB (Electronic Code Book), CBC (Cipher Block Chaining), and CFB (Cipher Feedback). We encrypted and decrypted text files using these modes.

Observations for Task 1:

Regardless of the encryption mode used, when we decrypted the files, they were successfully returned to their original state.

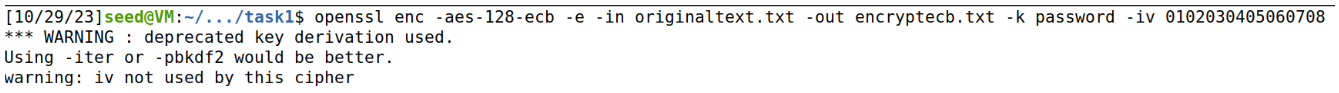
Notably, every encrypted file began with "Salted," indicating the use of a salt value in the encryption process. The "Salted" keyword is added to the beginning of the ciphertext to indicate the presence of a salt value used to derive the encryption key

Original File:

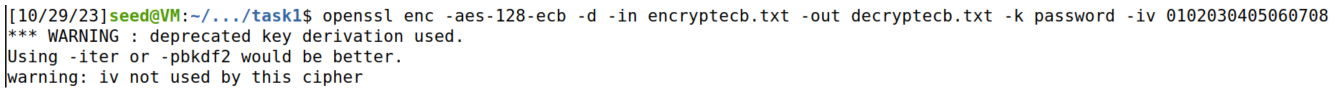


```
This is the text to be encrypted for task 1.
```

Encrypt & Decrypt ECB

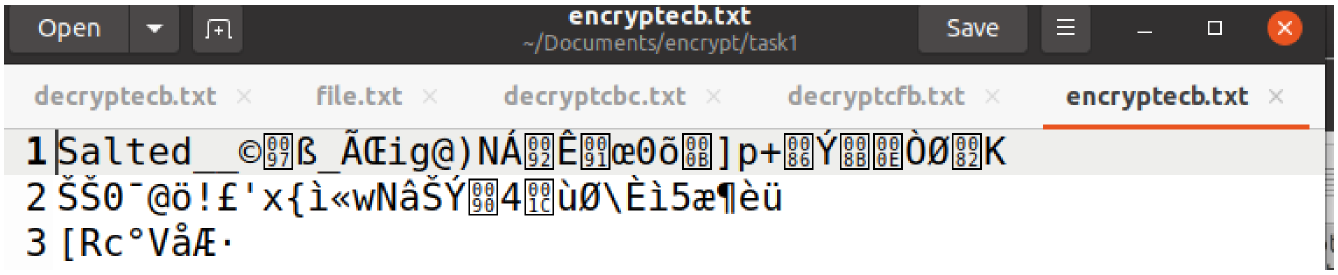


```
[10/29/23]seed@VM:~/.../task1$ openssl enc -aes-128-ecb -e -in originaltext.txt -out encryptecb.txt -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
warning: iv not used by this cipher
```



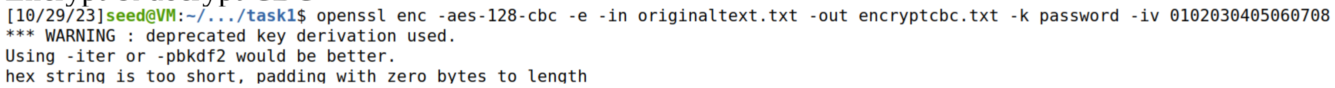
```
[10/29/23]seed@VM:~/.../task1$ openssl enc -aes-128-ecb -d -in encryptecb.txt -out decryptecb.txt -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
warning: iv not used by this cipher
```

Encrypted File:

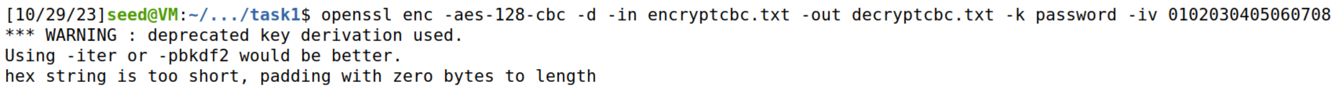


```
Salted__©0097ß_ÃŒig@)NÁ0092Ê0091œ0õ000B]p+0086Ý008B000EÒØ0082K
ŠŠ0¯@ö!£'x{ì«wNâŠÝ00904001CùØ\Èì5æ¶èü
[Rc°VåÆ·
```

Encrypt & decrypt CBC

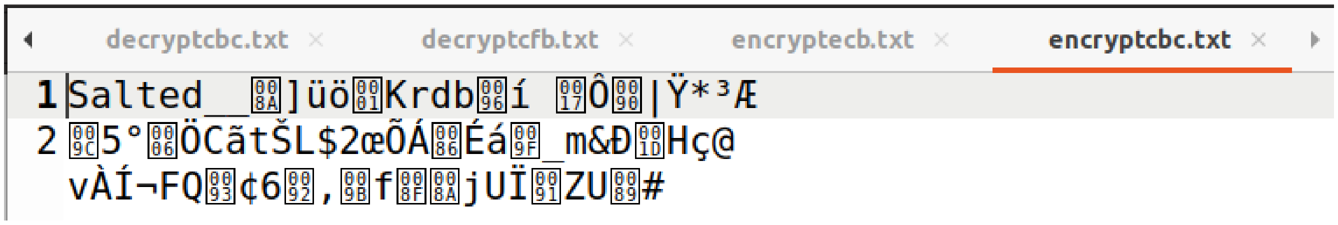


```
[10/29/23]seed@VM:~/.../task1$ openssl enc -aes-128-cbc -e -in originaltext.txt -out encryptcbc.txt -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
```



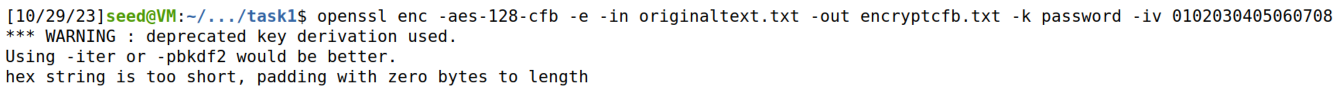
```
[10/29/23]seed@VM:~/.../task1$ openssl enc -aes-128-cbc -d -in encryptcbc.txt -out decryptcbc.txt -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
```

Encrypted File:

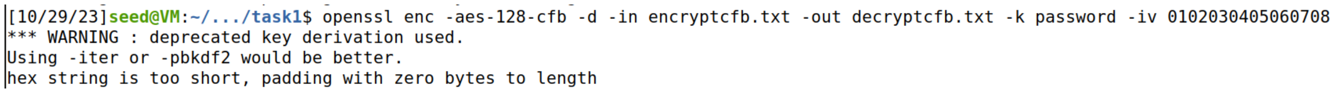


```
Salted__008A]üö0081Krdb0096í 0017Ô0098|Ÿ*³Æ
009C5°0086ÖCãtŠL$2œÕÁ0086Éá009F_m&Ð001DHç@
vÀÍ¬FQ0093¢60092,009Bf008F008AjUÏ0091ZU0089#
```

Encrypt & Decrypt CFB

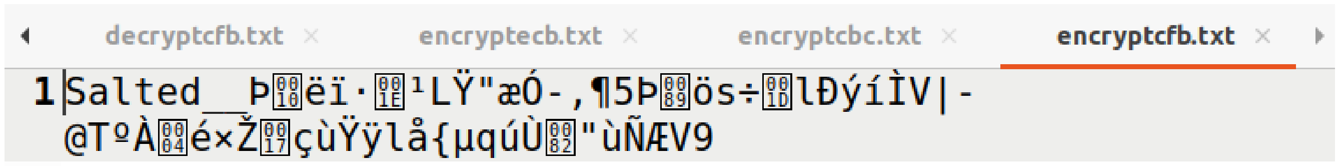


```
[10/29/23]seed@VM:~/.../task1$ openssl enc -aes-128-cfb -e -in originaltext.txt -out encryptcfb.txt -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
```



```
[10/29/23]seed@VM:~/.../task1$ openssl enc -aes-128-cfb -d -in encryptcfb.txt -out decryptcfb.txt -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
```

Encrypted file:



```
Salted__Þ0010ëï·001E¹LŸ"æÓ-,¶5Þ0089ös÷001DlÐýíÌV|-
@TºÀ0004é×Ž0017çùŸÿlå{µqúÙ0082"ùÑÆV9
```

Things to note for task 1: Every file starts with Salted

Task 2 Encryption Mode: ECB VS CBS

For this task, we encrypted a picture with two different algorithms the ECB and CBS and then carried out several actions to understand their behavior.

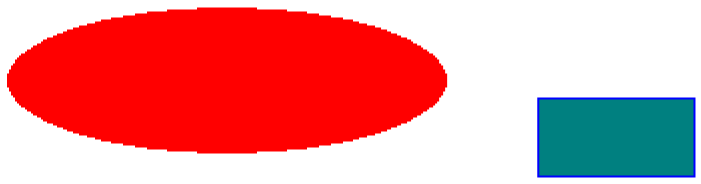
Observations for Task 2:

-ECB (Electronic Code Book) mode, the encrypted picture showed some similarities to the original, indicating that patterns in the image were preserved to some extent. The colors changed, but the shapes were somewhat recognizable.

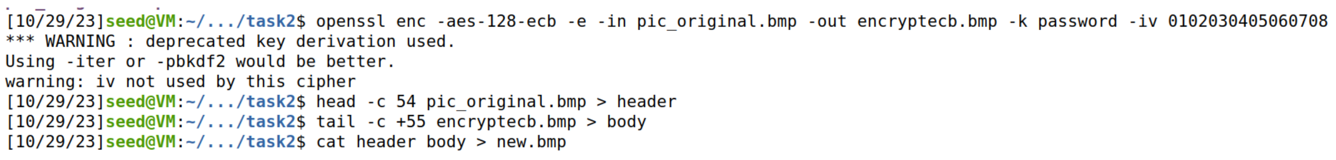
- **CBC (Cipher Block Chaining) mode**, the encrypted picture appeared completely unrecognizable. Both colors and shapes were heavily distorted.

- However, when decrypting the files, both the ECB and CBC encrypted files successfully returned to the original picture, indicating that the decryption process was able to recover the original data.

Original Picture:

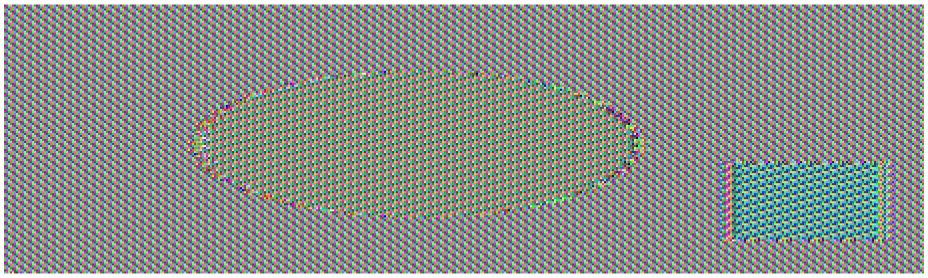


Encrypt ECB:

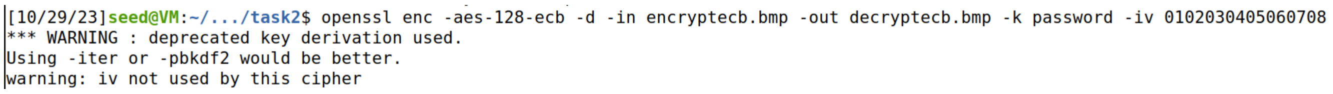


```
[10/29/23]seed@VM:~/.../task2$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out encryptecb.bmp -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
warning: iv not used by this cipher
[10/29/23]seed@VM:~/.../task2$ head -c 54 pic_original.bmp > header
[10/29/23]seed@VM:~/.../task2$ tail -c +55 encryptecb.bmp > body
[10/29/23]seed@VM:~/.../task2$ cat header body > new.bmp
```

ECB File Result:

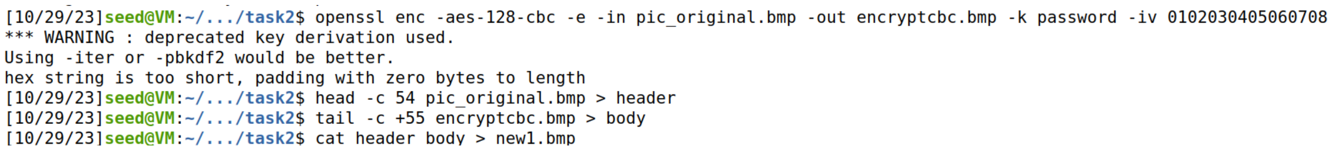


ECB file Decrypt



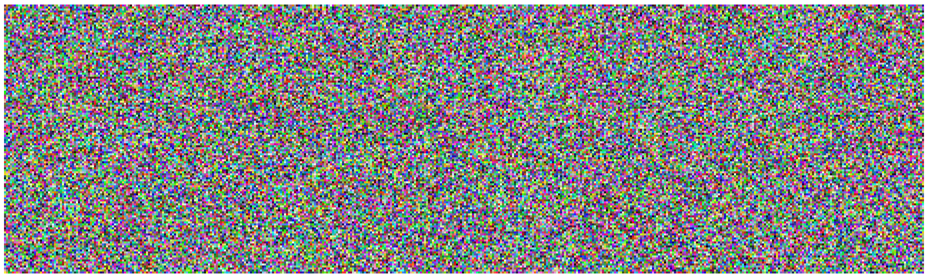
```
[10/29/23]seed@VM:~/.../task2$ openssl enc -aes-128-ecb -d -in encryptecb.bmp -out decryptecb.bmp -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
warning: iv not used by this cipher
```

Encrypt CBC

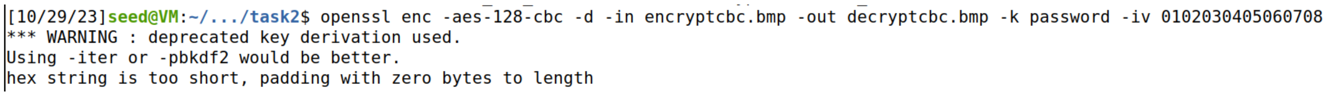


```
[10/29/23]seed@VM:~/.../task2$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out encryptcbc.bmp -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
[10/29/23]seed@VM:~/.../task2$ head -c 54 pic_original.bmp > header
[10/29/23]seed@VM:~/.../task2$ tail -c +55 encryptcbc.bmp > body
[10/29/23]seed@VM:~/.../task2$ cat header body > new1.bmp
```

CBC Ecrypt Result



Decrypt CBS



```
[10/29/23]seed@VM:~/.../task2$ openssl enc -aes-128-cbc -d -in encryptcbc.bmp -out decryptcbc.bmp -k password -iv 0102030405060708
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
```

Task 3:

In this task, we introduced an error into the ciphertext by corrupting a single bit in the 55th byte of the encrypted file. We then attempted to decrypt the corrupted file using different encryption modes (ECB, CBC, and CFB) to understand how error propagation affects each mode.

Observations for Task 3:
- **ECB (Electronic Code Book) Mode:** Since ECB mode processes each block of plaintext independently, changing a single bit in one block only affects that block during decryption. As a result, the error in the 55th byte only impacted a portion of the decrypted text, but the rest of the data remained intact. This means that the impact of the error is localized, and most of the information can be recovered.

- **CBC (Cipher Block Chaining) Mode:** In CBC mode, changing a single bit in one block not only affects that block but also has a cascading effect on subsequent blocks due to the XOR operation with the previous ciphertext block. This error propagation leads to a significant loss of information in the decrypted text. The error causes the entire decrypted content to be corrupted, making it challenging to recover any meaningful information.

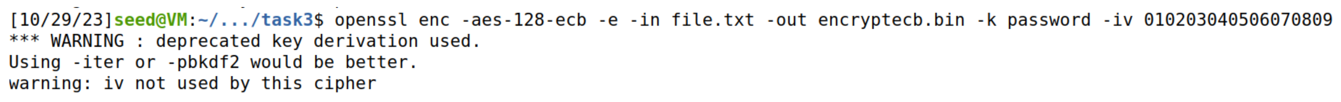
- CFB (Cipher Feedback) Mode: CFB mode, like CBC, propagates errors through subsequent blocks. However, the error impact is not as extensive as in CBC mode. In this case, some information can still be recovered, but a portion of the data is still lost due to the error.

Summary

Summary: The error propagation property of encryption modes significantly affects the ability to recover information from corrupted ciphertext. ECB mode localizes the error impact, making it possible to recover most of the information. CBC mode has a wide error propagation, leading to a loss of the entire decrypted content, while CFB mode falls somewhere in between, with partial information recovery.

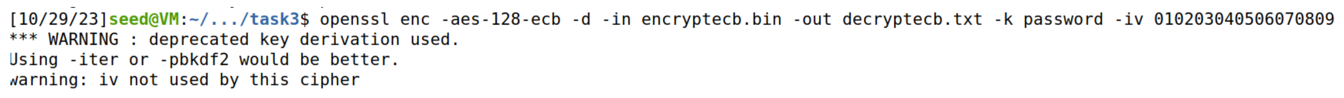
Conclusion: This lab assignment provided valuable insights into secret key cryptography, different encryption modes, and their error propagation properties. Understanding these concepts is essential for ensuring the security and integrity of encrypted data.

Encrypt ecb



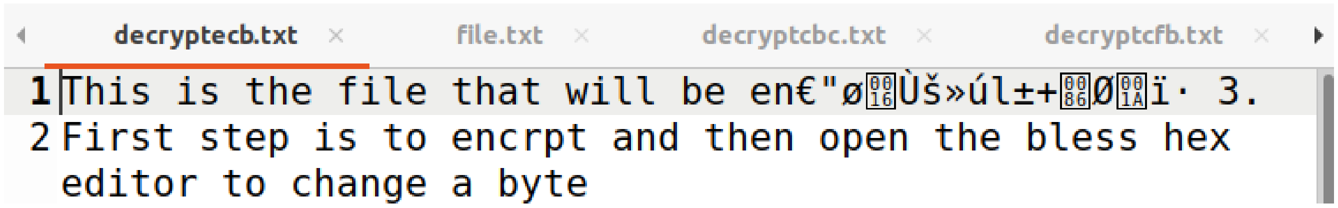
```
[10/29/23]seed@VM:~/.../task3$ openssl enc -aes-128-ecb -e -in file.txt -out encryptecb.bin -k password -iv 010203040506070809
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
warning: iv not used by this cipher
```

Decrypt ecb



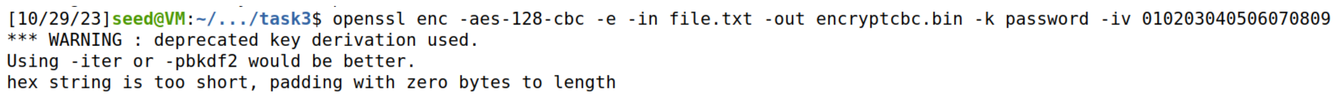
```
[10/29/23]seed@VM:~/.../task3$ openssl enc -aes-128-ecb -d -in encryptecb.bin -out decryptecb.txt -k password -iv 010203040506070809
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
warning: iv not used by this cipher
```

Decrypted File:



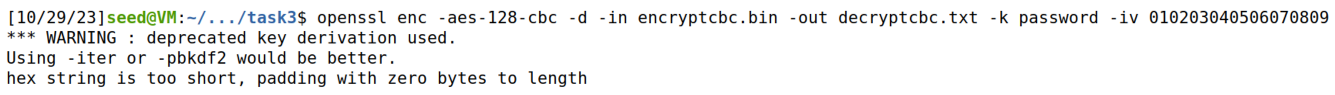
```
decryptecb.txt    file.txt    decryptcbc.txt    decryptcfb.txt
1 This is the file that will be en€"ø0016Ùš»úl±+0086Ø001Aï· 3.
2 First step is to encrpt and then open the bless hex
  editor to change a byte
```

Encrypt cbc



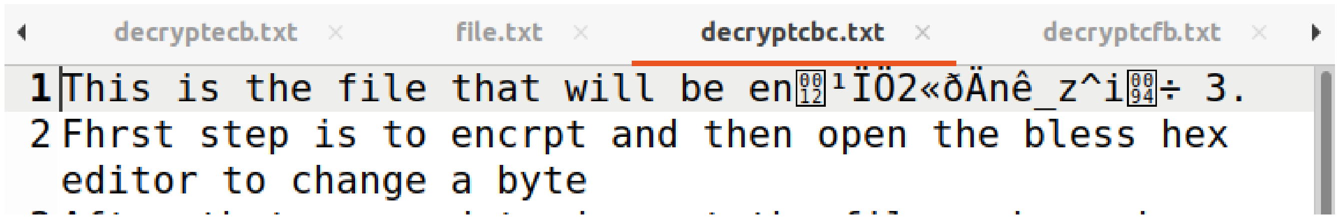
```
[10/29/23]seed@VM:~/.../task3$ openssl enc -aes-128-cbc -e -in file.txt -out encryptcbc.bin -k password -iv 0102030405060708090
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
```

Decrypt cbc



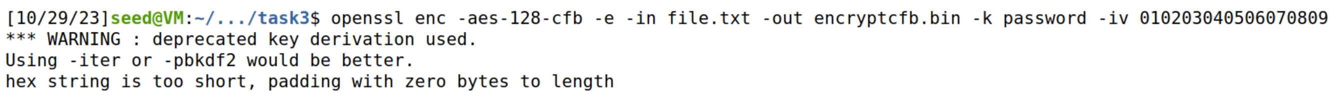
```
[10/29/23]seed@VM:~/.../task3$ openssl enc -aes-128-cbc -d -in encryptcbc.bin -out decryptcbc.txt -k password -iv 0102030405060708090
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
```

Decrypted File:



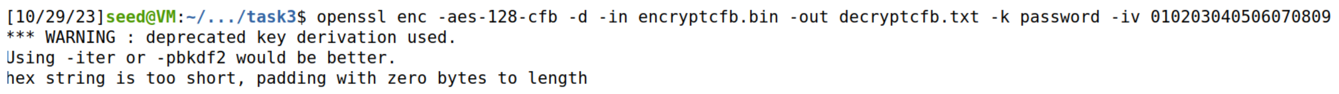
```
decryptecb.txt    file.txt    decryptcbc.txt    decryptcfb.txt
1 This is the file that will be en0012¹ÏÖ2«ðÄnê_z^i0094÷ 3.
2 Fhrst step is to encrpt and then open the bless hex
  editor to change a byte
```

Encrypt cfb



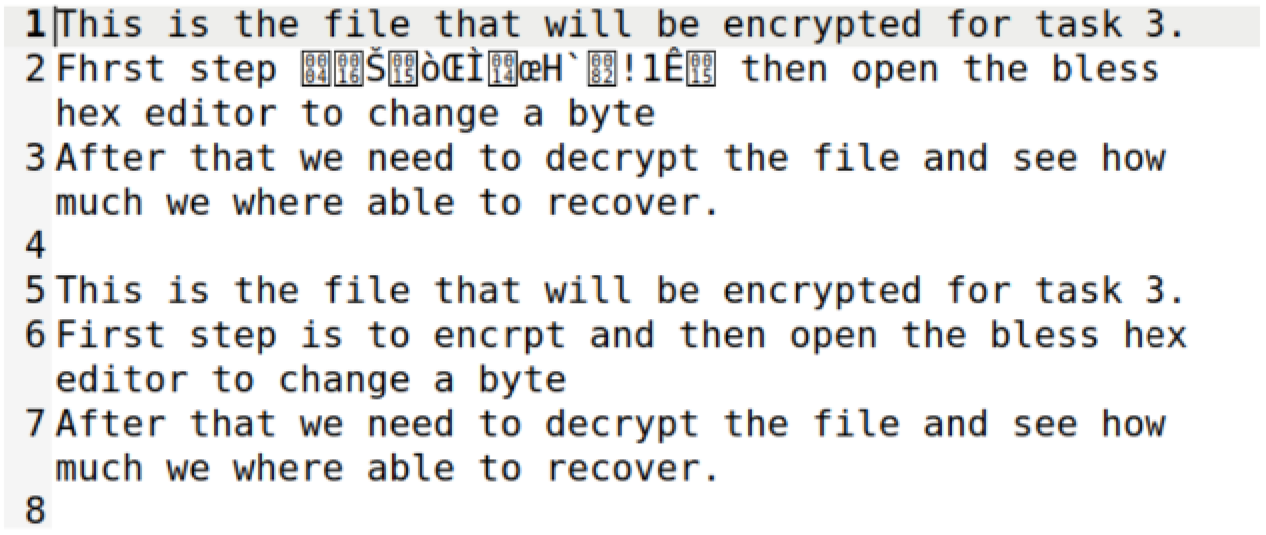
```
[10/29/23]seed@VM:~/.../task3$ openssl enc -aes-128-cfb -e -in file.txt -out encryptcfb.bin -k password -iv 0102030405060708090
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
```

Decrypt cfb



```
[10/29/23]seed@VM:~/.../task3$ openssl enc -aes-128-cfb -d -in encryptcfb.bin -out decryptcfb.txt -k password -iv 0102030405060708090
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
hex string is too short, padding with zero bytes to length
```

Decrypted File:



```
This is the file that will be encrypted for task 3.
Fhrst step ŠòŒÌœH`!1Ê then open the bless hex editor to change a byte
After that we need to decrypt the file and see how much we where able to recover.

This is the file that will be encrypted for task 3.
First step is to encrpt and then open the bless hex editor to change a byte
After that we need to decrypt the file and see how much we where able to recover.

```