

Brandon Hough

CPSC 4600

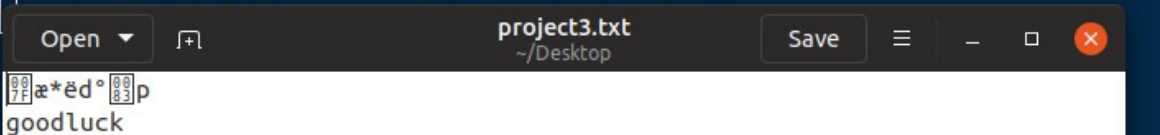
Dr. Yang

Project 3: Testing Different Modes in  
Symmetric Ciphers

## Task 1: Implement DES and AES ciphers (50 points)

I used OFB (output feedback) for encrypting and decrypting with DES3 (Data Encryption Standard) as shown below:

```
brandon@brandons-pc:~/Desktop$ python EncryDecry.py
Would you like to encrypt with AES or DES3? DES3
What algorithm mode would you want to use (ECB, CBC, CFB, OFB, or CTR)? OFB
Please input text to encrypt (8 characters): goodluck
Encrypted Message
=====
7fh e6h 2ah ebh 64h b0h 83h 70h
❖*❖❖❖❖
Decrypted Message
=====
67h 6fh 6fh 64h 6ch 75h 63h 6bh
goodluck
brandon@brandons-pc:~/Desktop$ gedit project3.txt
```



The screenshot shows a Gedit window titled 'project3.txt' with the following content:

```
7fh e6h 2ah ebh 64h b0h 83h 70h
❖*❖❖❖❖
Decrypted Message
=====
67h 6fh 6fh 64h 6ch 75h 63h 6bh
goodluck
```

I used CBC (cipher block chaining) for encrypting and decrypting with AES (Advanced Encryption Standard) as shown below:

```
brandon@brandons-pc:~/Desktop$ python EncryDecry.py
Would you like to encrypt with AES or DES3? AES
What algorithm mode would you want to use (ECB, CBC, CFB, OFB, or CTR)? CBC
Please input text to encrypt (16 characters): crackthisorelse!
Encrypted Message
=====
5dh 1h beh 2h 97h ch 7ah 69h 74h d4h ddh 30h d0h 3fh 36h bah
]00000
zit00?60
Decrypted Message
=====
63h 72h 61h 63h 6bh 74h 68h 69h 73h 6fh 72h 65h 6ch 73h 65h 21h
crackthisorelse!
brandon@brandons-pc:~/Desktop$ gedit project3.txt
```

The screenshot shows the gedit text editor window titled 'project3.txt' with the path '~/Desktop'. The editor contains the text 'crackthisorelse!' on the first line. Above the text, there is a hex dump view showing the corresponding hexadecimal values for each character: 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10. The text is displayed in a monospaced font on a dark background.

## Task 2: Investigate Properties of Modes in DES and AES (50 points)

The modes (ECB, CBC, CFB, OFB, or CTR) and encryption type (DES or AES) that the script will run depends on the given input by the user. The script will ask the user to select a mode, and then the text they want to encrypt. The message must be in blocks of 8/16 (DES/AES) characters based on the encryption type.

	ECB	CBC	CFB	OFB	CTR
Pattern Preservation	DES: Yes AES: Yes	DES: No AES: No	DES: No AES: No	DES: No AES: No	DES: N/A AES: Yes
Error Propagation	DES: Yes AES: Yes	DES: No AES: No	DES: No AES: No	DES: No AES: No	DES: N/A AES: No

Note: I was unable to use the CTR mode with DES.

Pattern preservation - Yes means that the pattern was still clear after the encryption, while no means that the pattern was not still clear after the encryption

Error propagation - Yes means that changing one bit did change the output of the encryption, while no means that changing one bit did not change the output of the encryption

Conclusion - As for pattern preservation, I only noticed that it was found when using ECB mode for AES and DES and with CTR mode for AES. As for error propagation, I was only able to see the plaintext after changing a bit at the end of the message when using ECB mode for AES and DES. With all the other modes, changing one bit ruined the entire message when trying to decrypt it. Below are my test cases that I tested to come up with these results and I have provided screenshots of the terminal and output file I created. I added a padding of 0's at the end if the length was not in 8/16 byte blocks.

[illegible]



[illegible]

[illegible]

[illegible]



### Case 5 (AES & ECB):

[illegible]



### Case 6 (AES & CBC):

[illegible]

### Case 7 (AES & CFB):

[illegible]

### Case 8 (AES & OFB):

[illegible]

[illegible]