Alexander Watson

CS 4351

Programming Assignment 4

**Task 1: Caesar Cipher**

1. **Decrypt**

Using a decryption setup, we will decrypt the message using ascii lowercase and making the original text lowercase as well.

**A screenshot of a computer screen

Description automatically generated**

Running the code with a few keys will show that the 4th shift completes the cipher.

**A screenshot of a computer code

Description automatically generated**

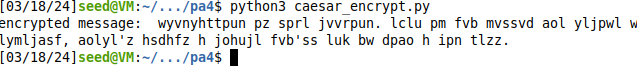
1. **Encrypt**

For the encryption I will use the same logic but in the reverse:

A screen shot of a computer

Description automatically generated

So instead of moving the position to the plain text we are shifting the plain text’s position by 7 as instructed. Running the script I get the following text:



**Task 2: Frequent Analysis**

1. **Relative Frequency**

Using CrypTool this is the relative frequency of the letters in the cypher text:

A screenshot of a graph

Description automatically generated

1. **Frequency Decryption**

The following is an example of the relative frequency of the English language:

A screenshot of a graph

Description automatically generatedThe Cipher Text:

lrvmnir bpr sumvbwvr jx bpr lmiwv yjeryrkbi jx qmbm wi bpr xjvni mkd ymibrut jx irhx wi bpr riirkvr jx ymbinlmtmipw utn qmumbr dj w ipmhh but bj rhnvwdmbr bpr yjeryrkbi jx bpr qmbm mvvjudwko bj yt wkbrusurbmbwjk lmird jk xjubt trmui jx ibndt wb wi kjb mk rmit bmiq bj rashmwk rmvp yjeryrkb mkd wbi iwokwxwvmkvr mkd ijyr ynib urymwk nkrashmwkrd bj ower m vjyshrbr rashmkmbwjk jkr cjnhd pmer bj lr fnmhwxwrd mkd wkiswurd bj invp mk rabrkb bpmb pr vjnhd urmvp bpr ibmbr jx rkhwopbrkrd ywkd vmsmlhr jx urvjokwgwko ijnkdhrii ijnkd mkd ipmsrhrii ipmsr w dj kjb drry ytirhx bpr xwkmh mnbpjuwbt lnb yt rasruwrkvr cwbp qmbm pmi hrxb kj djnlb bpmb bpr xjhhjcwko wi bpr sujsru msshwvmbwjk mkd wkbrusurbmbwjk w jxxru yt bprjuwri wk bpr pjsr bpmb bpr riirkvr jx jqwkmcmk qmumbr cwhh urymwk wkbmvb

**Task 3: DES**

Encryption: DES uses 64-bit blocks that use a key length of 56-bits which is derived from the block using 8th -bit as a parity bit, which is the XOR of the previous 7th-bit. The plaintext undergoes a set of substitution and then permutations before finally being encrypted through the XOR function

Decryption: In short, decryption is done by using the same steps but the subkeys are used in reverse order.

**Task 4: AES**

AES has a fixed block size of 128 bits, key length starting from the block size of 128 bits and then growing to either 192 bits or 256 bits. The key length depends on the number of rounds. 10 rounds for 128, 12 for 192, and 14 rounds for 256-bit kyes. The algorithm works by initializing the round through a XOR operation, then using a lookup table provides a nonlinear substitution, treating the bits like a matrix for each block the rows are shifted, and the columns are mixed, the key is XOR and added again. Finally, the AES algorithm repeats the substitution, row shifting, and key addition.

**Task 5: Encryption w/ Different ciphers/modes**

The first mode I chose to use is the aes-256 cypher which converted the text into both unrecognizable symbols of the terminal as well as non-English symbols such as Chinese and Arabic.

****

For the next encryption I used DES which produced a bunch of special characters, still including different language characters but also varying different types of other characters as well as a lot of blank space throughout the text.



Finally, I used the seed encryption mode and it produced a similar setup to AES-256 however, it appears to be making less whitespace and filling in more characters with special characters rather than alphanumeric characters.



**Task 6: ECB vs. CBC**

For this task there were multiple attempts at encrypting the image then combining the header with the encrypted body. The image became filled with noise each time.

A screenshot of a computer program

Description automatically generated

A green oval on a black background

Description automatically generated

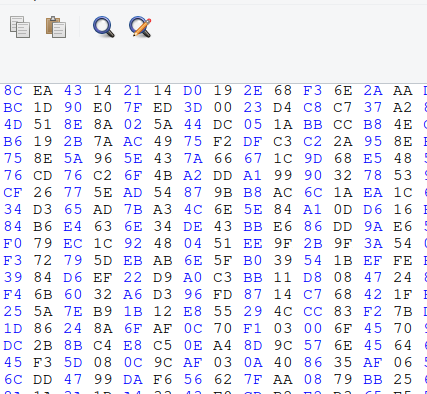
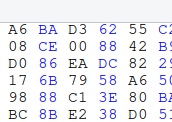
**Task 7: Error Propagation – Corrupted Cipher Text**

For this task I encoded the Shakespearean play Hamlet in different encryption modes.

A white background with black and blue text

Description automatically generated

Using bless I changed the 55th byte in all of the files here are two examples (problems with screenshot program on personal device):



Note: on the second line in both images the values are changed to 00.

My predictions:

ECB – the block containing the corrupted 55th byte will be corrupted so the rest will be fine.

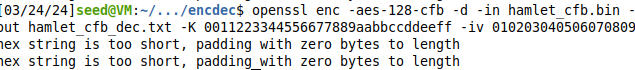
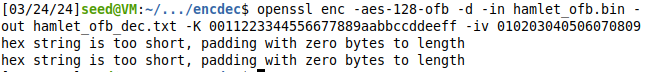
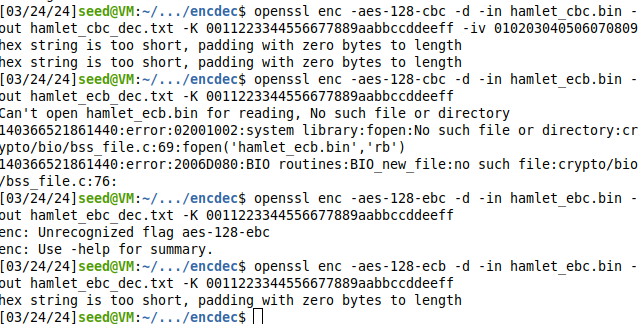
CBC – the block containing the corrupted 55th byte and the next block will be corrupted.

CFB – The corruption will be similar to CBC.

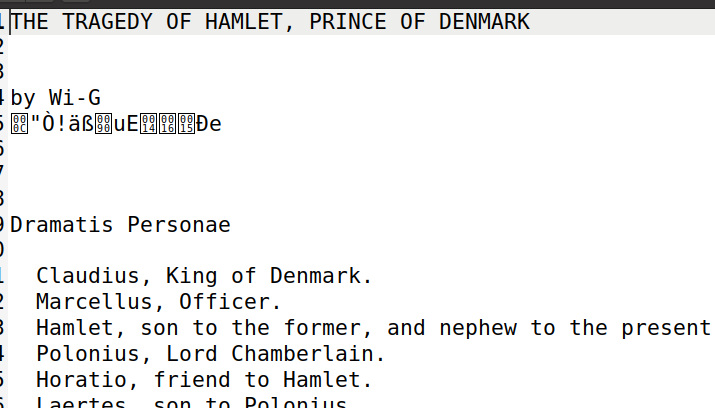
OFB – The corruption will be similar to ECB.

Results:

The following are the commands I used to decrypt the text



The following decryption was done in the order: CBC, ECB, OFB, and CFB.



This is the ECB and according to my predictions the OFB should be like it.

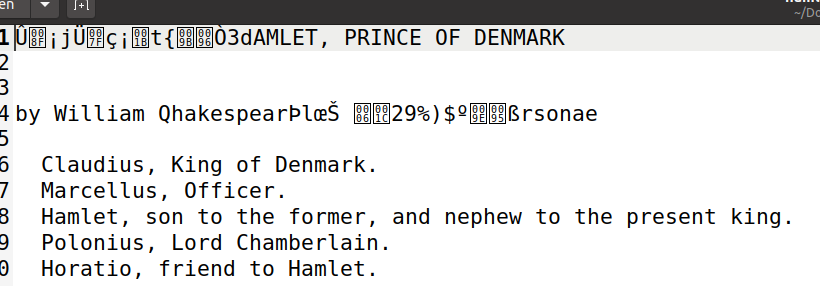


Unfortunately, I think there was a problem with encrypting and decrypting the file after multiple attempts I got similar results to the above.

A screenshot of a computer

Description automatically generated

However, the next being CBC and CFB being very similar to each other.



I believe that the prediction stands, that the corrupted bit in CBC and CFB more than just the block that the corrupted bit was in.