

Lab 3: Symmetric encryption & hashing

Task 1: AES Encryption Using Different Modes

1. First Created a plaintext file using the command– echo "This is a simple test msg on lab 3 on symmetric encryption" > plain.txt
2. Then Generate AES Encrypted file for each AES mode

i.CBC-Command used

```
-openssl enc -aes-128-cbc -e -in plain.txt -out cipher_aes_cbc.bin -K  
0011223344556677889aabbccddeeff -iv 010203040506070809000a0b0c0d0121
```

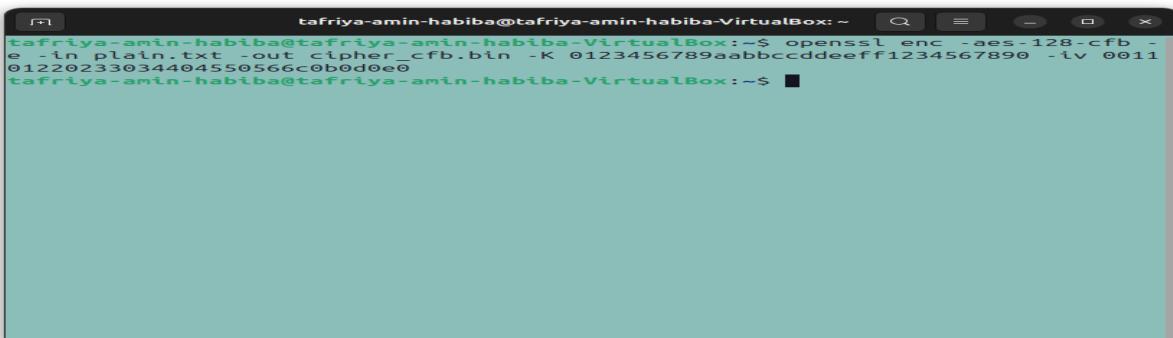


A screenshot of a terminal window titled 'tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox: ~'. The window contains the following text:

```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher_aes_cbc.bin \  
-K 0011223344556677889aabbccddeeff -iv 010203040506070809000a0b0c0d0121  
0a0b0c0d0f001122  
enc: Use -help for summary.  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher_aes_cbc.bin -K 0011223344556677889aabbccddeeff -iv 010203040506070809000a0b0c0d0121  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

ii.CFB-Command used

```
-openssl enc -aes-128-cfb -e -in plain.txt -out cipher_aes_cfb.bin -K  
0123456789aabbccddeeff1234567890 -iv 001101220233034404550566c0b0d0e0
```

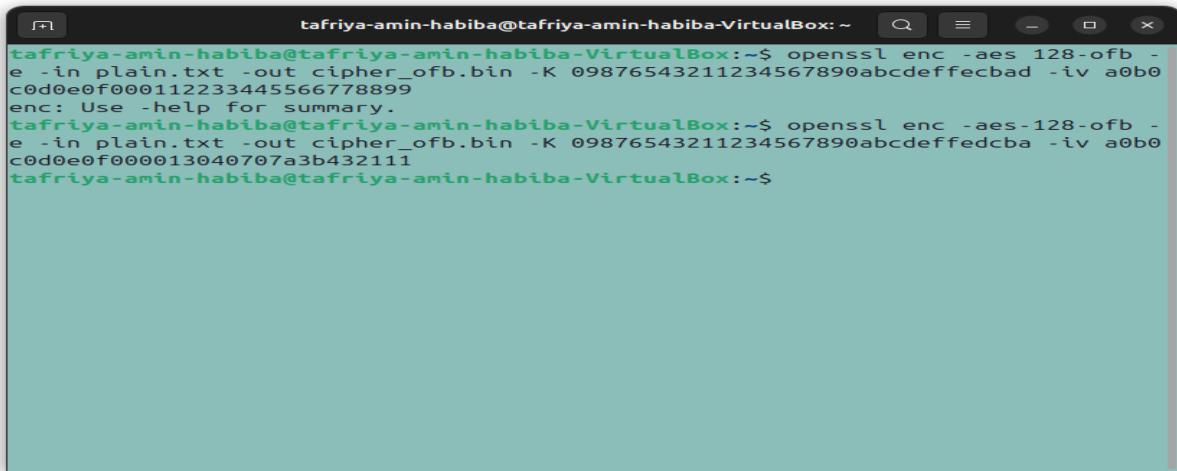


A screenshot of a terminal window titled 'tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox: ~'. The window contains the following text:

```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cfb -e -in plain.txt -out cipher_cfb.bin -K 0123456789aabbccddeeff1234567890 -iv 001101220233034404550566c0b0d0e0  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

iii.OFB-Command used

```
-openssl enc -aes-128-ofb -e -in plain.txt -out cipher_ofb.bin -K  
0123456789aabccddeeff1234567890 -iv 001101220233034404550566c0b0d0e0
```



A screenshot of a terminal window titled "tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox: ~". The window contains the following text:

```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes 128-ofb -  
e -in plain.txt -out cipher_ofb.bin -K 09876543211234567890abcdeffecbad -iv a0b0  
c0d0e0f000112233445566778899  
enc: Use -help for summary.  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ofb -  
e -in plain.txt -out cipher_ofb.bin -K 09876543211234567890abcdeffedcba -iv a0b0  
c0d0e0f000013040707a3b432111  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

3. AES Decryption to Verify

i.CBC

command-openssl enc -aes-128-cbc -d -in plain.txt -out cipher_aes_cbc.bin -K
00112233445566778889aabccddeeff -iv 010203040506070809000a0b0c0d0121

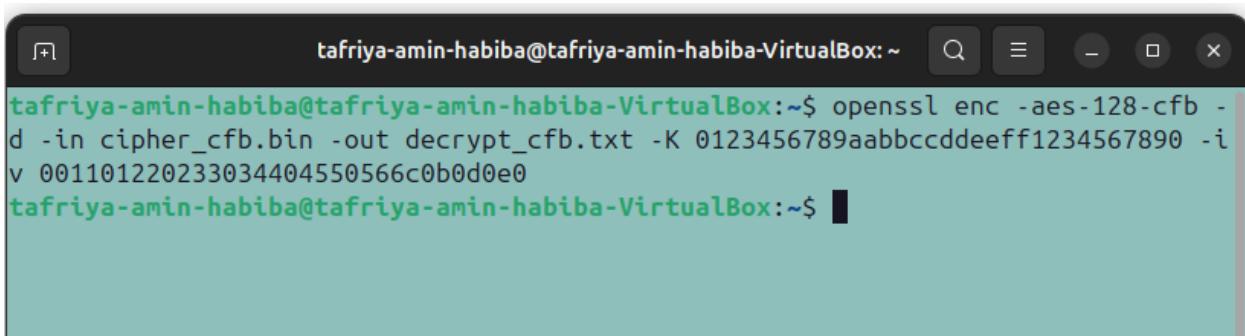


A screenshot of a terminal window titled "tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox: ~". The window contains the following text:

```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cbc -  
d -in cipher_aes_cbc.bin -out decrypt_cbc.txt -K 00112233445566778889aabccddeef  
f -iv 010203040506070809000a0b0c0d0121  
bad decrypt  
4007B5CE9C7F0000:error:1C800064:Provider routines:ossl_cipher_unpadblock:bad dec  
rypt:../providers/implementations/ciphers/ciphercommon_block.c:124:  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

ii.CFB

Command-openssl enc -aes-128-cfb -d -in plain.txt -out cipher_aes_cfb.bin -K 0123456789aabccddeff1234567890 -iv 001101220233034404550566c0b0d0e0



```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cfb -d -in cipher_cfb.bin -out decrypt_cfb.txt -K 0123456789aabccddeff1234567890 -iv 001101220233034404550566c0b0d0e0
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

iii.OFB

Command-openssl enc -aes-128-ofb -d -in plain.txt -out cipher_aes_ofb.bin -K 0123456789aabccddeff1234567890 -iv 001101220233034404550566c0b0d0e0



```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ofb -d -in cipher_ofb.bin -out decrypt_ofb.txt -K 09876543211234567890abcdefedcba -iv a0b0c0d0e0f000013040707a3b432111
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat decrypt_ofb.txt
This is a simple test msg for lab 3 on symmetric encryption
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

4.Verify the Results

Compare the decrypted files with the original plaintext:

cat decrypt_cbc.txt

cat decrypt_cfb.txt

cat decrypt_ofb.txt

Note: Previously we saw bad decryption in CBC that's why when we verify it we are not getting original plaintext

Task:2- Encryption mode - ECB vs CBC



Flower.jpg

1. First Convert image from jpg to bmp

Command-convert image.png to image.bmp

```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ ls
cipher_aes_cbc.bin  decrypt_cfb.txt  Downloads  plain.txt      Templates
cipher_cfb.bin       decrypt_ofb.txt  images.png  plain.txt.save  Videos
cipher_ofb.bin        Desktop        Music      Public
decrypt_cbc.txt      Documents      Pictures   snap
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ convert images.png image.b
mp
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ ls | grep bmp
image.bmp
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ecb -
e -in image.bmp -out image_ecb.bmp -K 00112233445566778899aabbcdddeeff
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ dd if=image.bmp of=header.
bin bs=1 count=54
54+0 records in
54+0 records out
54 bytes copied, 0.0000507857 s, 106 kB/s
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ dd if=image_ecb.bmp of=body_
ecb.bin bs=1 skip=54
47610+0 records in
47610+0 records out
47610 bytes (48 kB, 46 KiB) copied, 0.108102 s, 440 kB/s
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat header.bin body_ecb.bi
n > ecb_final.bmp
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

2. Encryption Commands:

ECB Mode Encryption: openssl enc -aes-128-ecb -e -in pic_original.bmp
-out encrypted_ecb.bmp \ -K 0123456789abcdef0123456789abcdef

CBC Mode Encryption: openssl enc -aes-128-cbc -e -in pic_original.bmp
-out encrypted_cbc.bmp \ -K 0123456789abcdef0123456789abcdef \ -iv
0102030405060708090a0b0c0d0e0f10

3. Header Replacement Commands:

Extract original header

(first 54 bytes) dd if=pic_original.bmp of=header.bin bs=1 count=54

Extract ECB encrypted body (skip header)

```
dd if=encrypted_ecb.bmp of=body_ecb.bin bs=1 skip=54
# Extract CBC encrypted body (skip header)
dd if=encrypted_cbc.bmp of=body_cbc.bin bs=1 skip=54
# Combine original header with ECB encrypted body
cat header.bin body_ecb.bin > ecb_final.bmp
# Combine original header with CBC encrypted body
cat header.bin body_cbc.bin > cbc_final.bmp
```

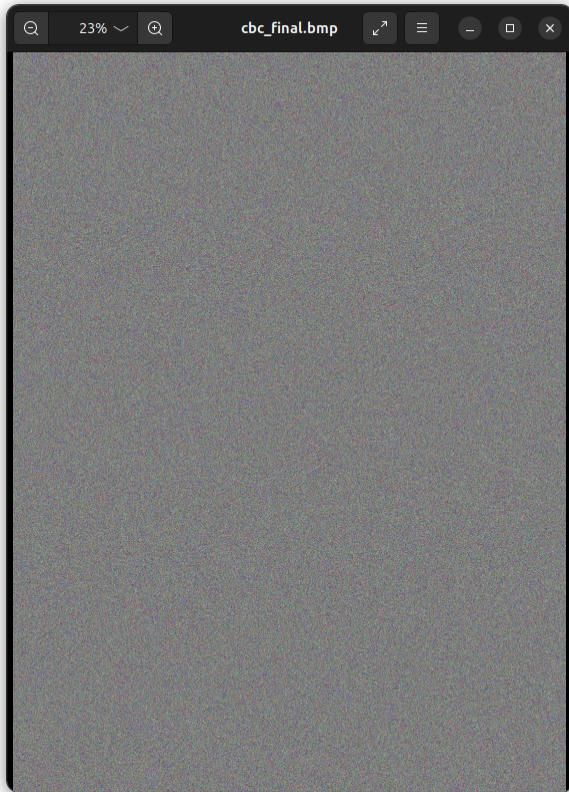
```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$ cd Pictures
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$ openssl enc -aes-128-cbc -e -in flower.bmp -out flower_cbc.bmp -K 00112233445566778899aabbcdddef -iv 0102030405060708090a0b0c0d0e0f00
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$ dd if=flower_cbc.bmp of=body_cbc.bin bs=1 skip=54
23970906+0 records in
23970906+0 records out
23970906 bytes (24 MB, 23 MiB) copied, 22.5856 s, 1.1 MB/s
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$ cat header.bin body_cbc.bin > cbc_final.bmp
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$ eog cbc_final.bmp
```

```
Can't open "flower.bmp" for reading, No such file or directory
40C7867CBF700000:error:80000002:system library:BIO_new_file:No such file or directory:../crypto/bio/bss_file.c:67:calling fopen(flower.bmp, rb)
40C7867CBF700000:error:10000080: BIO routines:BIO_new_file:no such file:../crypto/bio/bss_file.c:75:
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cd Photos
bash: cd: Photos: No such file or directory
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Photos$ cd Pictures
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$ openssl enc -aes-128-ecb -e -in flower.bmp -out flower_ecb.bmp -K 00112233445566778899aabbcdddef
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$ dd if=flower.bmp of=header.bin bs=1 count=54
54+0 records in
54+0 records out
54 bytes copied, 0.000246657 s, 219 kB/s
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$ dd if=flower_ecb.bmp of=body_ecb.bin bs=1 skip=54
23970906+0 records in
23970906+0 records out
23970906 bytes (24 MB, 23 MiB) copied, 37.1229 s, 646 kB/s
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$ cat header.bin body_ecb.bin > ecb_final.bmp
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~/Pictures$
```

ECB



CBC



Observation

During the experiment, I used a color image as input for AES encryption using different modes (ECB and CBC).

After performing encryption, I observed that:

- The AES-ECB encrypted image still partially preserved the structure and color patterns of the original image.
This happened because ECB encrypts identical plaintext blocks into identical ciphertext blocks, which makes patterns visible.
- In contrast, the AES-CBC encrypted image appeared much more random and noise-like, effectively hiding all recognizable patterns.

However, since the image was colorful, the visual difference between ECB and CBC modes was less distinguishable than expected.

The colors and visual textures somewhat blurred the contrast between the two, though CBC still provides stronger security in theory

TASK-3:Encryption mode – corrupted cipher text

1. Test File Creation: -

Filename: myfile.txt - File size: 38 bytes

- Content: This is a test file for my lab task 3.

2. Encryption Commands Used:

ECB Mode: openssl enc -aes-128-ecb -e -in plaintext_task3.txt -out encrypted_ecb_task3.bin \ -K 0123456789abcdef0123456789abcdef

CBC Mode: openssl enc -aes-128-cbc -e -in plaintext_task3.txt -out encrypted_cbc_task3.bin \ -K 0123456789abcdef0123456789abcdef \ -iv 0102030405060708090a0b0c0d0e0f10

CFB Mode: openssl enc -aes-128-cfb -e -in plaintext_task3.txt -out encrypted_cfb_task3.bin \ -K 0123456789abcdef0123456789abcdef \ -iv 0102030405060708090a0b0c0d0e0f10

OFB Mode: openssl enc -aes-128-ofb -e -in plaintext_task3.txt -out encrypted_ofb_task3.bin \ -K 0123456789abcdef0123456789abcdef \ -iv 0102030405060708090a0b0c0d0e0f10

```

wc: plaintext2.txt: No such file or directory
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "This is a test file
for my lab task 3" > myfile.txt

tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ wc -c myfile.txt
38 myfile.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat myfile.txt
This is a test file for my lab task 3
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cbc -
e -in myfile.txt -out encrypted_cbc_task.bin -K 00112233445566778899aabcccddeeff
-iv 010203040506070809000a0b0c0d0e0f
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ecb -
e -in myfile.txt -out encrypted_ecb_task.bin -K 00112233445566778899aabcccddeeff
-iv 010203040506070809000a0b0c0d0e0f
warning: iv not used by this cipher
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ecb -
e -in myfile.txt -out encrypted_ecb_task.bin -K 00112233445566778899aabcccddeeff
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cfb -
e -in myfile.txt -out encrypted_cfb_task.bin -K 00112233445566778899aabcccddeeff
-iv 010203040506070809000a0b0c0d0e0f
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ofb -
e -in myfile.txt -out encrypted_ofb_task.bin -K 00112233445566778899aabcccddeeff
-iv 010203040506070809000a0b0c0d0e0f
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ █

```

3. Corruption Method: - Corrupted byte position: 30th byte (index 29)

) - Method: Flipped one bit using XOR with 0x01

Corruption commands (example for ECB): cp encrypted_ecb_task.bin
corrupted_ecb_task3.bin

BYTE=\$(xxd -s 29 -l 1 -p encrypted_ecb_task3.bin)

NEW_BYTEx=\$((printf "%02x" \$((0x\$BYTE ^ 0x01)))) echo -n -e "\x\$NEW_BYTEx" | dd of=corrupted_ecb_task3.bin bs=1 seek=29 conv=notrunc

[Repeat similar commands for CBC, CFB, OFB]

```
xxd: encrypted_ecb_task.bin: No such file or directory
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ xxd -s 29 -l 1 encrypted_ecb_task.bin
0000001d: d8
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ xxd -p encrypted_ecb_task.bin > ecb_hex.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ nano ecb_hex.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cp encrypted_ecb_task.bin corrupted_ecb_task.bin
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ BYTE=$(xxd -s 29 -l 1 -p encrypted_ecb_task.bin)
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ NEW_BYTE=$(printf "%02x" $((0x$BYTE ^ 0x01)))
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo -n -e "\x$NEW_BYTE" | dd of=corrupted_ecb_task.bin bs=1 seek=29 conv=notrunc 2>/dev/null
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "Original byte 30: 0x$BYTE"
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "Original byte 30: 0x$BYTE"
Original byte 30: 0xd8
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "Corrupted byte 30: 0x$NEW_BYTE"
Corrupted byte 30: 0xd9
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

```
ncrypted_cfb_task.bin)
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ NEW_BYTE=$(printf "%02x" $((0x$BYTE ^ 0x01)))
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echob -n -e "\x$NEW_BYTE" | dd of=corrupted_cfb_task.bin bs=1 seek=29 conv=notrunc 2>/dev/null
Command 'echob' not found, did you mean:
  command 'echo' from deb coreutils (9.4-3ubuntu6.1)
Try: sudo apt install <deb name>
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo -n -e "\x$NEW_BYTE" | dd of=corrupted_cfb_task.bin bs=1 seek=29 conv=notrunc 2>/dev/null
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "CFB - Original: 0x$BYTE,Corrupted: 0x$NEW_BYTE"
CFB - Original: 0xf,Corrupted: 0xe
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ ./ncrypted_ofb_task.bin  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ NEW_BYTEx=$(printf "%02x" $((0x$BYTE ^ 0x01)))  
NEW_BYTEx=b5: command not found  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ NEW_BYTEx=$(printf "%02x" $((0x$BYTE ^ 0x01)))  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo -n -e "\x$NEW_BYTEx" | dd of=corrupted_ofb_task.bin bs=1 seek=29 conv=notrunc 2>/dev/null  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "OFB-Original: 0x$BYTE"  
E,Corrupted: 0x$NEW_BYTEx  
OFB-Original: 0xb4,Corrupted: 0xb5  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

```
bin > cbc_hex.txt  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ nano cbc_hex.txt  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cp encrypted_cbc_task.bin  
corrupted_cbc_task.bin  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ BYTE=$(xxd -s 29 -l 1 -p encrypted_cbc_task.bin)  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ NEW_BYTEx=$(printf "%02x" $((0x$BYTE ^ 0x01)))  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo -n -e "\x$NEW_BYTEx" | dd of=corrupted_cbc_task.bin bs=1 seek=29 conv=notrunc 2>/dev/null  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "Original byte 30: 0x$BYTE"  
Original byte 30: 0x39  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "Corrupted byte 30: 0x$NEW_BYTEx"  
Corrupted byte 30: 0x38  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

4. Decryption Commands:

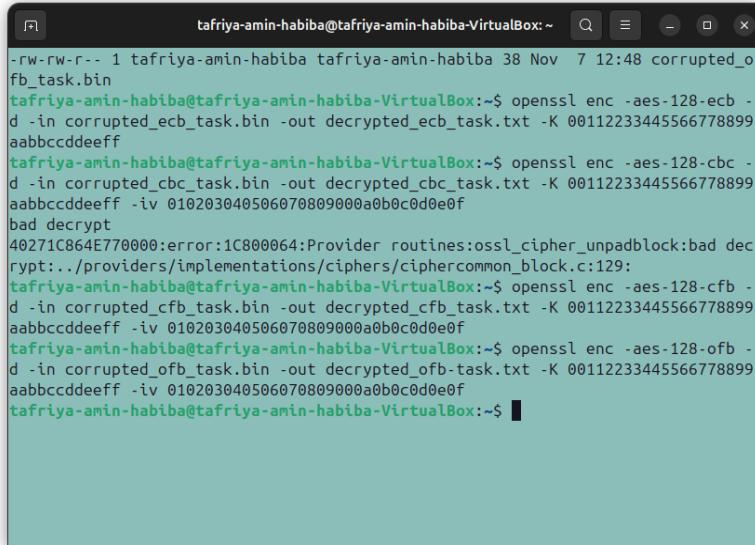
ECB Decryption:

```
openssl enc -aes-128-ecb -d -in corrupted_ecb_task3.bin -out decrypted_ecb_task3.txt \-K 0123456789abcdef0123456789abcdef
```

CBC Decryption: openssl enc -aes-128-cbc -d -in corrupted_cbc_task3.bin -out decrypted_cbc_task3.txt \-K 0123456789abcdef0123456789abcdef \-iv 0102030405060708090a0b0c0d0e0f10

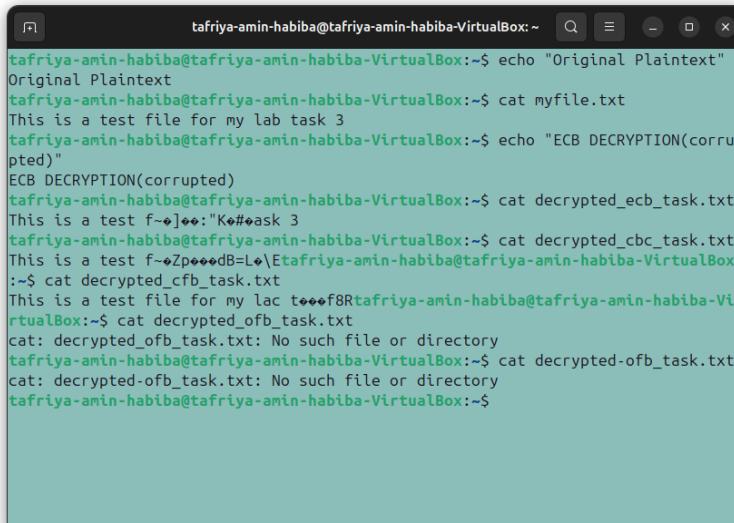
```
CFB Decryption: openssl enc -aes-128-cfb -d -in corrupted_cfb_task3.bin -out decrypted_cfb_task3.txt \ -K 0123456789abcdef0123456789abcdef \ -iv 0102030405060708090a0b0c0d0e0f10
```

```
OFB Decryption: openssl enc -aes-128-ofb -d -in corrupted_ofb_task3.bin -out decrypted_ofb_task3.txt \ -K 0123456789abcdef0123456789abcdef \ -iv 0102030405060708090a0b0c0d0e0f10
```



```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ ls
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ecb -d -in corrupted_ecb_task.bin -out decrypted_ecb_task.txt -K 00112233445566778899 aabbccddeeff
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cbc -d -in corrupted_cbc_task.bin -out decrypted_cbc_task.txt -K 00112233445566778899 aabbccddeeff -iv 010203040506070809000a0b0c0d0e0f
bad decrypt
40271C864E770000:error:1C800064:Provider routines:ossl_cipher_unpadblock:bad decrpyt:../providers/implementations/ciphers/ciphercommon_block.c:129:
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cfb -d -in corrupted_cfb_task.bin -out decrypted_cfb_task.txt -K 00112233445566778899 aabbccddeeff -iv 010203040506070809000a0b0c0d0e0f
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ofb -d -in corrupted_ofb_task.bin -out decrypted_ofb_task.txt -K 00112233445566778899 aabbccddeeff -iv 010203040506070809000a0b0c0d0e0f
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

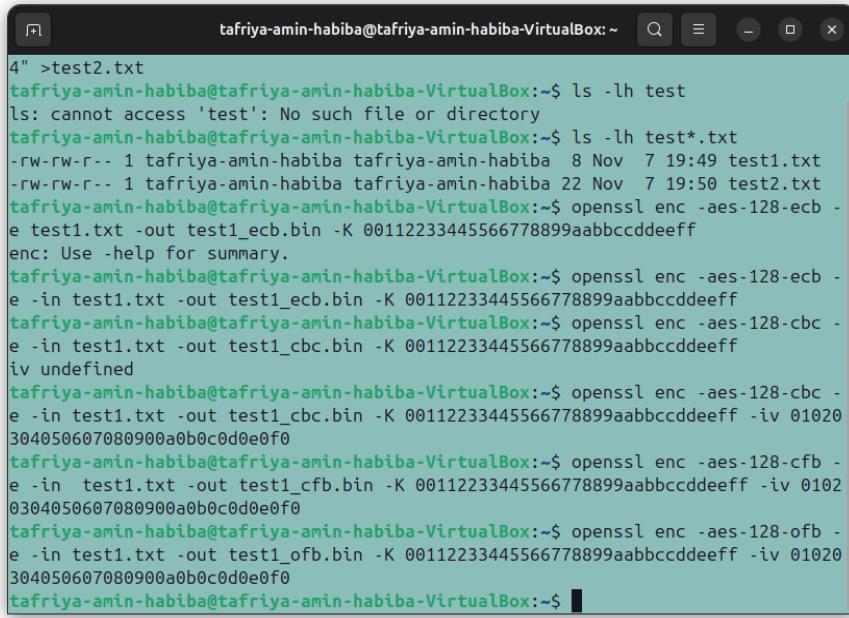
Decryption Outputs from Corrupted Ciphertext



```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "Original Plaintext"
Original Plaintext
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat myfile.txt
This is a test file for my lab task 3
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ echo "ECB DECRYPTION(corrupted)"
ECB DECRYPTION(corrupted)
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat decrypted_ecb_task.txt
This is a test f->]:::"K#ask 3
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat decrypted_cbc_task.txt
This is a test f->Zp***dB=L\|Etafriya-amin-habiba@tafriya-amin-habiba-VirtualBox
:-$ cat decrypted_cfb_task.txt
This is a test file for my lac t***f8Rtafriya-amin-habiba@tafriya-amin-habiba-Vi
rtualBox:-$ cat decrypted_ofb_task.txt
cat: decrypted_ofb_task.txt: No such file or directory
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat decrypted-ofb_task.txt
cat: decrypted-ofb_task.txt: No such file or directory
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

TASK-4:PADDING

1. created two text files with different sizes to test if encryption adds extra bytes (padding) to the files. test1.txt - 8 bytes test2.txt - 22 bytes



```
4" >test2.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ ls -lh test
ls: cannot access 'test': No such file or directory
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ ls -lh test*.txt
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 8 Nov 7 19:49 test1.txt
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 22 Nov 7 19:50 test2.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ecb -e test1.txt -out test1_ecb.bin -K 00112233445566778899aabccddeeff
enc: Use -help for summary.
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ecb -e -in test1.txt -out test1_ecb.bin -K 00112233445566778899aabccddeeff
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cbc -e -in test1.txt -out test1_cbc.bin -K 00112233445566778899aabccddeeff
iv undefined
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cbc -e -in test1.txt -out test1_cbc.bin -K 00112233445566778899aabccddeeff -iv 01020304050607080900a0b0c0d0e0f0
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cfb -e -in test1.txt -out test1_cfb.bin -K 00112233445566778899aabccddeeff -iv 01020304050607080900a0b0c0d0e0f0
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ofb -e -in test1.txt -out test1_ofb.bin -K 00112233445566778899aabccddeeff -iv 01020304050607080900a0b0c0d0e0f0
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

2. Commands:

For test1.txt:

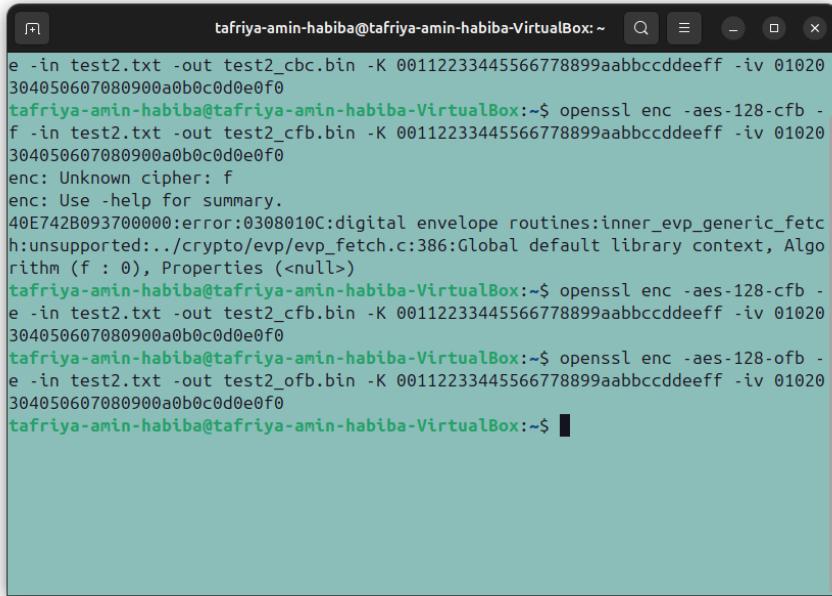
ECB : openssl enc -aes-128-ecb -e -in test1.txt -out test1_ecb.bin \ -K 00112233445566778899aabccddeeff

CBC : openssl enc -aes-128-cbc -e -in test1.txt -out test1_cbc.bin \ -K 00112233445566778899aabccddeeff \ -iv 0102030405060708090a0b0c0d0e0f10

CFB : openssl enc -aes-128-cfb -e -in test1.txt -out test1_cfb.bin \ -K 00112233445566778899aabccddeeff \ -iv 0102030405060708090a0b0c0d0e0f10

OFB : openssl enc -aes-128-ofb -e -in test1.txt -out test1_ofb.bin \ -K 00112233445566778899aabccddeeff \ -iv 0102030405060708090a0b0c0d0e0f10

used similar commands for test2.txt (just replaced test1 with test2)



```
e -in test2.txt -out test2_cbc.bin -K 00112233445566778899aabbccddeeff -iv 01020  
304050607080900a0b0c0d0e0f0  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cfb -  
f -in test2.txt -out test2_cfb.bin -K 00112233445566778899aabbccddeeff -iv 01020  
304050607080900a0b0c0d0e0f0  
enc: Unknown cipher: f  
enc: Use -help for summary.  
40E742B09370000:error:0308010C:digital envelope routines:inner_evp_generic_fetc  
h:unsupported:../crypto/evp/evp_fetch.c:386:Global default library context, Algo  
rithm (f : 0), Properties (<null>)tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-cfb -  
e -in test2.txt -out test2_cfb.bin -K 00112233445566778899aabbccddeeff -iv 01020  
304050607080900a0b0c0d0e0f0  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl enc -aes-128-ofb -  
e -in test2.txt -out test2_ofb.bin -K 00112233445566778899aabbccddeeff -iv 01020  
304050607080900a0b0c0d0e0f0  
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

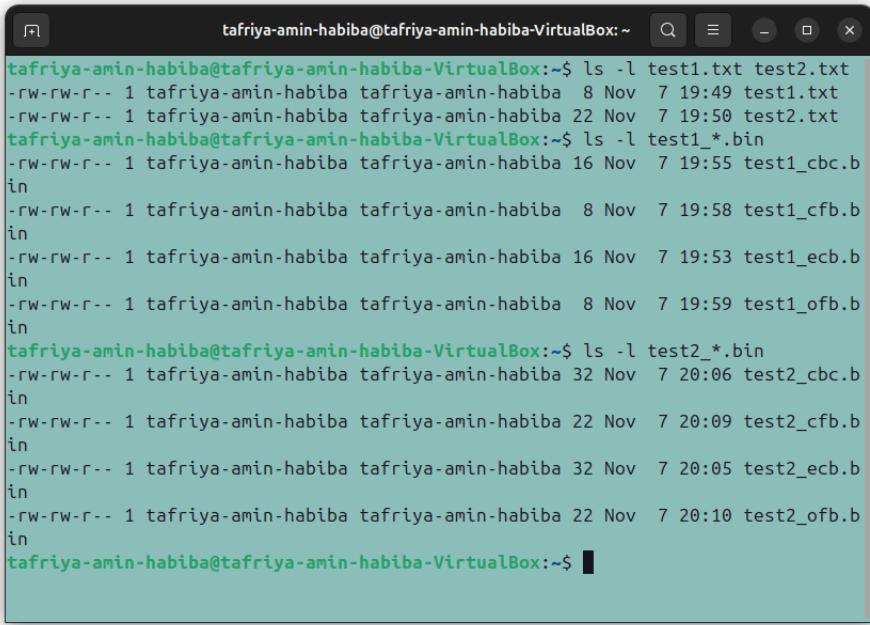
3. Results

: For test1.txt (8 bytes original):

- ECB encrypted: 16 bytes
- CBC encrypted: 16 bytes
- CFB encrypted: 8 bytes
- OFB encrypted: 8 bytes

For test2.txt (22 bytes original):

- ECB encrypted: 32 bytes
- CBC encrypted: 32 bytes
- CFB encrypted: 22 bytes
- OFB encrypted: 22 bytes



A screenshot of a terminal window titled "tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~". The window displays two sets of file listing results. The first set, after the command "ls -l test1.txt test2.txt", shows files test1.txt and test2.txt with sizes of 8 and 22 respectively. The second set, after the command "ls -l test1_*.bin", shows four files: test1_cbc.bin (size 16), test1_ecb.bin (size 16), test1_cfb.bin (size 8), and test1_ofb.bin (size 8). The third set, after the command "ls -l test2_*.bin", shows four files: test2_cbc.bin (size 32), test2_ecb.bin (size 32), test2_cfb.bin (size 22), and test2_ofb.bin (size 22). The terminal prompt is "\$" at the bottom.

```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ ls -l test1.txt test2.txt
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 8 Nov 7 19:49 test1.txt
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 22 Nov 7 19:50 test2.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ ls -l test1_*.bin
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 16 Nov 7 19:55 test1_cbc.bin
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 8 Nov 7 19:58 test1_cfb.bin
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 16 Nov 7 19:53 test1_ecb.bin
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 8 Nov 7 19:59 test1_ofb.bin
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ ls -l test2_*.bin
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 32 Nov 7 20:06 test2_cbc.bin
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 22 Nov 7 20:09 test2_cfb.bin
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 32 Nov 7 20:05 test2_ecb.bin
-rw-rw-r-- 1 tafriya-amin-habiba tafriya-amin-habiba 22 Nov 7 20:10 test2_ofb.bin
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

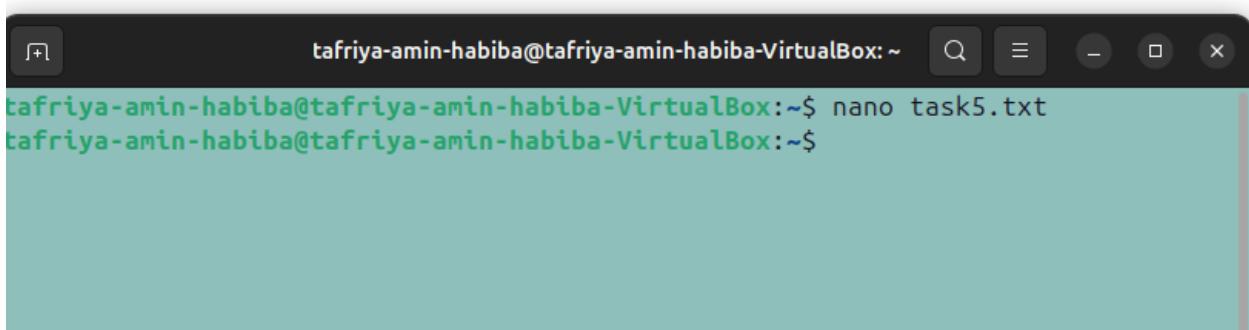
4. What I found:

Modes that need padding: ECB and CBC - These modes made the encrypted file bigger than the original - ECB and CBC added extra bytes to make the file size a multiple of 16 Modes that don't need padding: CFB and OFB - These modes kept the same file size as the original - No extra bytes were added.

From what I understand, ECB and CBC work on complete blocks of 16 bytes. So if the file is not a multiple of 16, they need to add padding to fill the last block. CFB and OFB seem to work differently - they can handle any file size without needing to add extra bytes. I think they work more like stream ciphers that encrypt byte by byte instead of block by block.

TASK 5: Generating Message Digest

1. created a text file called task5.txt with some simple text about cryptography. Then I used OpenSSL to generate hash values for this file using three different algorithms



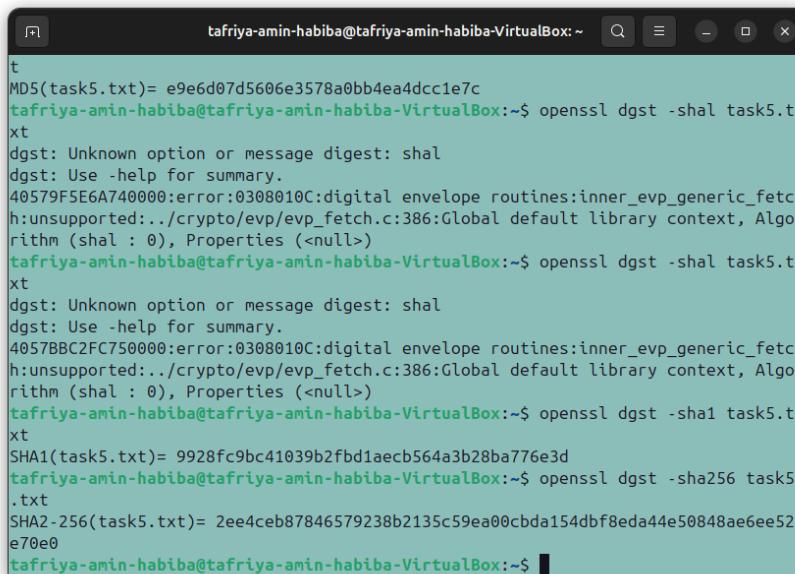
```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ nano task5.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

2. Commands:

MD5 hash: openssl dgst -md5 task5.txt

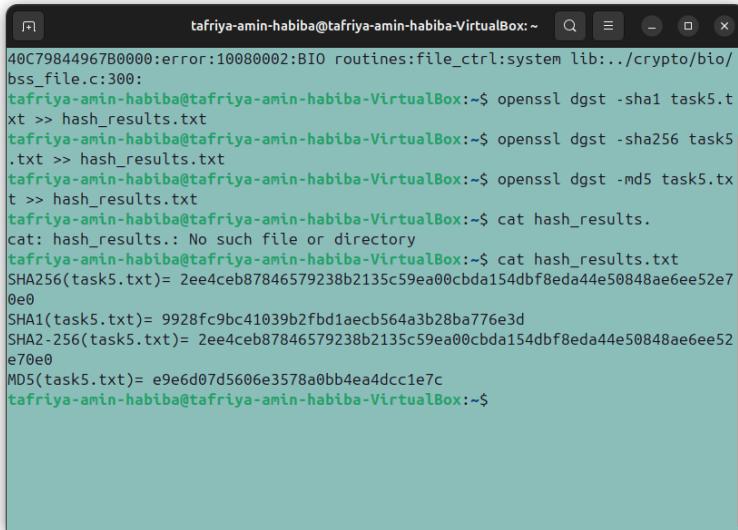
SHA-1 hash: openssl dgst -sha1 task5.txt

SHA-256 hash: openssl dgst -sha256 task5.txt



```
t
MD5(task5.txt)= e9e6d07d5606e3578a0bb4ea4dcc1e7c
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha1 task5.t
xt
dgst: Unknown option or message digest: sha1
dgst: Use -help for summary.
40579F5E6A740000:error:0308010C:digital envelope routines:inner_evp_generic_fetc
h:unsupported:../crypto/evp/evp_fetch.c:386:Global default library context, Algo
rithm (shal : 0), Properties (<null>)
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha1 task5.t
xt
dgst: Unknown option or message digest: sha1
dgst: Use -help for summary.
4057BBC2FC750000:error:0308010C:digital envelope routines:inner_evp_generic_fetc
h:unsupported:../crypto/evp/evp_fetch.c:386:Global default library context, Algo
rithm (shal : 0), Properties (<null>)
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 task5.t
xt
SHA1(task5.txt)= 9928fc9bc41039b2fbfd1aecb564a3b28ba776e3d
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 task5.t
xt
SHA2-256(task5.txt)= 2ee4ceb87846579238b2135c59ea00cbda154dbf8eda44e50848ae6ee52
e70e0
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

Result:



```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha1 task5.txt >> hash_results.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 task5.txt >> hash_results.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -md5 task5.txt >> hash_results.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat hash_results.
cat: hash_results.: No such file or directory
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat hash_results.txt
SHA256(task5.txt)= 2ee4ceb87846579238b2135c59ea00cbda154dbf8eda44e50848ae6ee52e70e0
SHA1(task5.txt)= 9928fc9bc41039b2fb1aecb564a3b28ba776e3d
SHA2-256(task5.txt)= 2ee4ceb87846579238b2135c59ea00cbda154dbf8eda44e50848ae6ee52e70e0
MD5(task5.txt)= e9e6d07d5606e3578a0bb4ea4dcc1e7c
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

Observation:

All three algorithms produced different hash values for the same file, but the hash format looks similar (hexadecimal numbers and letters). The longer hashes (like SHA-256) seem to be more secure because they have more possible combinations. I read that MD5 is considered weak now and SHA-256 is recommended for security. One interesting thing I noticed is that the same file always gives the same hash value when I run the command multiple times

TASK:6- Keyed Hash and HMAC

Generate keyed hash values using HMAC.

HMAC stands for Hash-based Message Authentication Code.

It's a cryptographic technique used to verify both the integrity and authenticity of a message. I've created a text file and generate keyed hashes (HMAC) using different algorithms (MD5, SHA1, SHA256) and different key lengths.

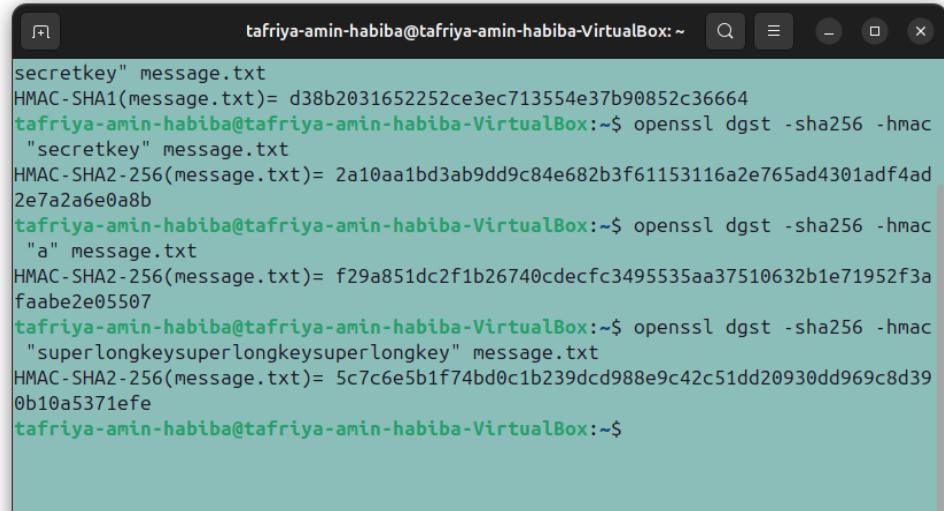
Keys I used "secretkey", "a", "superlongkeysuperlongkeysuperlongkey"

Text = This is my file for keyed hash and HMAC testing. HMAC provides integrity and authentication.

Commands:

```
-openssl dgst -md5 -hmac "secretkey" message.txt
-openssl dgst -sha1 -hmac "secretkey" message.txt
```

```
-openssl dgst -sha256 -hmac " secretkey " message.txt
```



A terminal window titled "tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~". The user runs several OpenSSL commands to generate HMAC signatures for the file "message.txt" using the "secretkey". The output shows four different HMAC values: HMAC-MD5, HMAC-SHA1, HMAC-SHA2-256, and HMAC-SHA2-256 with a long key.

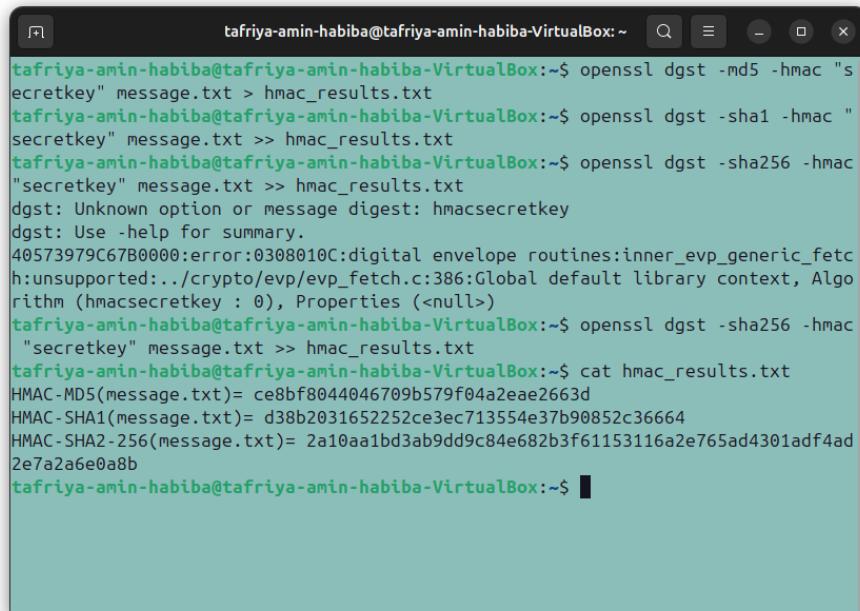
```
secretkey" message.txt
HMAC-SHA1(message.txt)= d38b2031652252ce3ec713554e37b90852c36664
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 -hmac
"secretkey" message.txt
HMAC-SHA2-256(message.txt)= 2a10aa1bd3ab9dd9c84e682b3f61153116a2e765ad4301adf4ad
2e7a2a6e0a8b
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 -hmac
"a" message.txt
HMAC-SHA2-256(message.txt)= f29a851dc2f1b26740cdecfc3495535aa37510632b1e71952f3a
faabe2e05507
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 -hmac
"superlongkeysuperlongkeysuperlongkey" message.txt
HMAC-SHA2-256(message.txt)= 5c7c6e5b1f74bd0c1b239cd988e9c42c51dd20930dd969c8d39
0b10a5371efe
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

Results:

HMAC-MD5(message.txt)= 2ffe8b648e066ad9fb9857abd8e9bf78

HMAC-SHA1(message.txt)= 983a8a67d14fdb853934d0de64f029984a283d07

HMAC-SHA2-256(message.txt)=3a239e80ac59a44c0c9efee21f53ae62d3067fd8d38554fee27d
c093c1f7503



A terminal window titled "tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~". The user attempts to run several OpenSSL commands using the "-md5" option instead of "-sha256", which results in errors. They also attempt to use a long key, which fails due to a library error. Finally, they successfully run the correct SHA256 command and check the results.

```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -md5 -hmac "s
ecretkey" message.txt > hmac_results.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha1 -hmac "
secretkey" message.txt >> hmac_results.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 -hmac
"secretkey" message.txt >> hmac_results.txt
dgst: Unknown option or message digest: hmacsecretkey
dgst: Use -help for summary.
40573979C67B0000:error:0308010C:digital envelope routines:inner_evp_generic_fetc
h:unsupported../../crypto/evp/evp_fetch.c:386:Global default library context, Algo
rithm (hmacsecretkey : 0), Properties (<null>)
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 -hmac
"secretkey" message.txt >> hmac_results.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat hmac_results.txt
HMAC-MD5(message.txt)= ce8bf8044046709b579f04a2eae2663d
HMAC-SHA1(message.txt)= d38b2031652252ce3ec713554e37b90852c36664
HMAC-SHA2-256(message.txt)= 2a10aa1bd3ab9dd9c84e682b3f61153116a2e765ad4301adf4ad
2e7a2a6e0a8b
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

Observation:

HMAC combines key and message to ensure data integrity and authentication. Key length can vary since hashing normalizes it internally.

Do we have to use a fixed key size in HMAC? Why doesn't it need fixed size?

No, HMAC does not require a fixed key size. I tested with keys ranging from 3 characters to over 100 characters, and all of them worked. The HMAC algorithm can handle variable-length keys.

From my understanding, HMAC has an internal process that handles different key sizes. If the key is shorter than the block size, it gets padded. If it's longer than the block size, it gets hashed first to make it shorter. This is why any key length works.

Task 7 – Randomness of One-Way Hash (Bonus 2 Marks)

Demonstrate the avalanche effect in hashing by flipping one bit in input.

Check how much a small change (1 bit flip) in a file affects its hash value for MD5 and SHA256. I created a text file, generated its hash (H1), then changed just one bit in the file and generated te hash again (H2). Then compared the hash functions.

”

Creating the original file:

```
echo"....." <originalText.txt  
echo "....." <modifiedText.txt
```

Commands:

```
-openssl dgst -md5 originalText.txt  
-openssl dgst -md5 modifiedText.txt  
-openssl dgst -sha256 originalText.txt  
-openssl dgst -sha256 modifiedText.txt
```

```
on" > modifiedText.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -md5 original
Text.txt
MD5(originalText.txt)= 0f3afa2c9699297a1a96d161d9685050
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 origi
nalText.txt
SHA2-256(originalText.txt)= 1cd254f24328f8ea89df447cde8fb6e656c32b371a2e55b38265
f11e9514512a
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ xxd originalText.txt
00000000: 7468 6973 2069 7320 6f72 6967 69e6 616c this is original
00000010: 2066 696c 650a file.
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cp original.Text.txt modif
iedText.txt
cp: cannot stat 'original.Text.txt': No such file or directory
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cp originalText.txt modifi
edText.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -md5 modified
Text.txt
MD5(modifiedText.txt)= 0f3afa2c9699297a1a96d161d9685050
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 modif
iedText.txt
SHA2-256(modifiedText.txt)= 1cd254f24328f8ea89df447cde8fb6e656c32b371a2e55b38265
f11e9514512a
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

Results:

MD5(originalText.txt)= cb91361f15283764c0501136f849c609

MD5(modifiedText.txt)= 2a2ee6b0b739e260a49cd45f4ca0e77c

SHA2-256(originalText.txt)=

168135a36be9ee3b10f9895dc770b111af2dabfab46a98b54bf4fddc9641fb99

SHA2-256(modifiedText.txt)=

44cbab28a66f962393ce3939405f7b9e483fa169d245e139036679df5209cbe3

```
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -md5 modified
Text.txt > hash_compare.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -md5 original
Text.txt > hash_compare.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 modif
iedText.txt >> hash_compare.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ openssl dgst -sha256 origi
nalText.txt >> hash_compare.txt
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$ cat hash_compare.txt
MD5(originalText.txt)= 0f3afa2c9699297a1a96d161d9685050
SHA2-256(modifiedText.txt)= 1cd254f24328f8ea89df447cde8fb6e656c32b371a2e55b38265
f11e9514512a
SHA2-256(originalText.txt)= 1cd254f24328f8ea89df447cde8fb6e656c32b371a2e55b38265
f11e9514512a
tafriya-amin-habiba@tafriya-amin-habiba-VirtualBox:~$
```

Observation:

H1 and H2 are completely different, even though I changed only one bit in the file. For both MD5 and SHA256, the new hashes look totally random and have no visible similarity to the original ones. This happens because of the **avalanche effect** — a small change in the input causes a big change in the hash output.

This is important for security because it means that:

- Any tiny change in a file can be easily detected.
- Attackers cannot make small edits to get a similar hash.
- It's very hard to guess or reverse the original data from the hash.