Lab Manual 3: Symmetric Encryption & Hashing (Task 1 and Task 2 Report)

Task 1: AES Encryption Using Different Modes

Objective: To encrypt a text file using AES-128 cipher with three different encryption modes and verify the encryption by performing decryption.

Method:

Step 1: Created a text file named plain.txt with multiple lines of text content, then saved the file in the working directory

Step 2: Encryption using three different Modes

Mode 1: AES-128-CBC (Cipher Block Chaining)

Encryption Command: openssl enc -aes-128-cbc -e -in plain.txt -out cipher_cbc.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708

Decryption Command: openssl enc -aes-128-cbc -d -in cipher_cbc.bin -out decrypted_cbc.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708

Mode 2: AES-128-CFB (Cipher Feedback)

Encryption Command: openssl enc -aes-128-cfb -e -in plain.txt -out cipher_cfb.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708

Decryption Command: openssl enc -aes-128-cfb -d -in cipher_cfb.bin -out decrypted_cfb.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708

Mode 3: AES-128-OFB (Output Feedback)

Encryption Command: openssl enc -aes-128-ofb -e -in plain.txt -out cipher_ofb.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708

Decryption Command: openssl enc -aes-128-ofb -d -in cipher_ofb.bin -out decrypted_ofb.txt -K 00112233445566778899aabbccddeeff -iv 0102030405060708

Results

1. All three encryption modes successfully encrypted the plain text file
2. Decryption restored the original content correctly in all three modes

3. The encrypted files (cipher_cbc.bin, cipher_cfb.bin, cipher_ofb.bin) contain unreadable binary data
4. All decrypted files matched the original plain.txt content exactly

Observations:

1. **CBC Mode**: Requires both key and IV; provides good security through block chaining
2. **CFB Mode**: Operates as a stream cipher; requires both key and IV
3. **OFB Mode**: Also operates as a stream cipher; requires both key and IV

Task 2: Encryption Mode - ECB vs CBC

Objective: To compare ECB and CBC encryption modes by encrypting a BMP image and observing the visual differences in the encrypted outputs.

Method:

Step 1:

1. Converted the original image to BMP format using Paint application
2. Created a working directory at: C:\Users\USER\Desktop\Lab\Task2
3. Changed PowerShell directory to the Task2 folder using: cd C:\Users\USER\Desktop\Lab\Task2

Step 2:

Encryption Command: openssl enc -aes-128-ecb -e -in lemurbmp.bmp -out lemur_ecb.enc -K 00112233445566778899aabbccddeeff

Header Replacement Process:

1. Opened the original lemurbmp.bmp file in HxD hex editor
2. Selected Edit , then Select Block
3. Copied the first 54 bytes (BMP header information)
4. Opened the encrypted file lemur_ecb.enc in HxD
5. Selected the first 54 bytes of the encrypted file
6. Pasted the copied header from the original file
7. Saved the file as lemurview_ecb.bmp

Step 3: CBC Mode Encryption

Encryption Command: openssl enc -aes-128-cbc -e -in lemurbmp.bmp -out lemur_cbc.enc -K 00112233445566778899aabbccddeeff -iv 0102030405060708090a0b0c0d0e0f10

Header Replacement Process:

1. Opened the original lemurbmp.bmp file in HxD hex editor
2. Selected and copied the first 54 bytes
3. Opened the encrypted file lemur_cbc.enc in HxD
4. Selected the first 54 bytes of the encrypted file
5. Pasted the copied header from the original file
6. Pressed Ctrl+S to save
7. Saved the file as lemurview_cbc.bmp

Results and Observations

ECB Mode Results:

1. The ECB-encrypted image (lemurview_ecb.bmp) reveals visible patterns from the original image
2. Identical plaintext blocks produce identical ciphertext blocks
3. The outline and structure of the original image remain visible

CBC Mode Results:

1. The CBC-encrypted image (lemurview_cbc.bmp) appears completely randomized
2. No visible patterns or structure from the original image
3. The image looks like random noise/static
4. No useful information can be derived about the original image content

Analysis:

Why ECB Mode is Insecure for Images:

1. ECB encrypts each block independently without any chaining
2. Identical plaintext blocks always produce identical ciphertext blocks
3. Images often contain large areas of uniform or similar colors
4. These repetitive patterns are preserved in the encrypted output
5. An attacker can derive significant information about the image structure

Why CBC Mode is More Secure:

1. CBC uses an Initialization Vector (IV) and chains blocks together
2. Each plaintext block is XORed with the previous ciphertext block before encryption
3. Identical plaintext blocks produce different ciphertext blocks
4. The chaining mechanism ensures that patterns are not preserved
5. The encrypted output appears completely random

Security Implications:

1. ECB mode should never be used for encrypting images or any data with patterns
2. CBC mode provides confidentiality by hiding visual patterns
3. The choice of encryption mode significantly impacts security
4. Block chaining mechanisms (like CBC) are essential for semantic security