Lab 3: Cryptography using OpenSSL

Name: Amit Kumar Sharma

Registration: 2020831009

---

1. Objectives

- Practice symmetric encryption modes (AES: ECB, CBC, CFB, OFB) using OpenSSL.
- Encrypt images and text files using AES.
- Demonstrate encryption/decryption with corrupted files.
- Perform hashing using MD5, SHA-1, SHA-256, and SHA-512.
- Use HMAC with short, long, and very short keys.
- Compare original and modified file hashes.

---

2. Tasks and Commands

Task 1: AES Encryption of Text Files

CBC Encryption:

```
openssl enc -aes-128-cbc -e -in plain.txt -out out-cbc.bin -K
00112233445566778899aabbccddeeff -iv 0102030405060708
```

ECB Encryption:

```
openssl enc -aes-128-ecb -e -in plain.txt -out out-ecb.bin -K
00112233445566778899aabbccddeeff
```

CFB Encryption:

```
openssl enc -aes-128-cfb -e -in plain.txt -out out-cfb.bin -K
00112233445566778899aabbccddeeff -iv 0102030405060708
```

Task 2: Image Encryption

ECB Mode:

```
openssl enc -aes-128-ecb -e -in pic_original.bmp -out pic_ecb.bin -K
00112233445566778899aabbccddeeff
```

CBC Mode:

```
openssl enc -aes-128-cbc -e -in pic_original.bmp -out pic_cbc.bin -K
00112233445566778899aabbccddeeff -iv 0102030405060708
```

Task 3: Text Encryption/Decryption with Corruption

ECB Mode

Encryption:

```
openssl enc -aes-128-ecb -e -in long_text.txt -out encrypted_ecb.bin
-K 00112233445566778889aabbccddeeff
```

Decryption of Corrupted File:

```
openssl enc -aes-128-ecb -d -in corrupted_ecb.bin -out
corrupted_ecb.txt -K 00112233445566778889aabbccddeeff
```

CBC Mode

Encryption:

```
openssl enc -aes-128-cbc -e -in long_text.txt -out encrypted_cbc.bin
-K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Decryption of Corrupted File:

```
openssl enc -aes-128-cbc -d -in corrupted_cbc.bin -out
corrupted_cbc.txt -K 00112233445566778889aabbccddeeff -iv
0102030405060708
```

CFB Mode

Encryption:

```
openssl enc -aes-128-cfb -e -in long_text.txt -out encrypted_cfb.bin
-K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Decryption:

```
openssl enc -aes-128-cfb -d -in corrupted_cfb.bin -out
corrupted_cfb.txt -K 00112233445566778889aabbccddeeff -iv
0102030405060708
```

OFB Mode

Encryption:

```
openssl enc -aes-128-ofb -e -in long_text.txt -out encrypted_ofb.bin
-K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Decryption:

```
openssl enc -aes-128-ofb -d -in corrupted_ofb.bin -out
corrupted_ofb.txt -K 00112233445566778889aabbccddeeff -iv
0102030405060708
```

Task 4: Short Text Encryption

ECB:

```
openssl enc -aes-128-ecb -e -in short.txt -out short_ecb.bin -K
00112233445566778889aabbccddeeff
```

CBC:

```
openssl enc -aes-128-cbc -e -in short.txt -out short_cbc.bin -K
00112233445566778889aabbccddeeff -iv 0102030405060708
```

CFB:

```
openssl enc -aes-128-cfb -e -in short.txt -out short_cfb.bin -K
00112233445566778889aabbccddeeff -iv 0102030405060708
```

OFB:

```
openssl enc -aes-128-ofb -e -in short.txt -out short_ofb.bin -K
00112233445566778889aabbccddeeff -iv 0102030405060708
```

Task 5: Hashing

MD5

```
openssl dgst -md5 hash_sample.txt
```

Output: `8c840e2a4cf16f39c00f05ae0bebb02a`

SHA-1

```
openssl dgst -sha1 hash_sample.txt
```

Output: `91eb45b52babf4500cc578e6492e1dad0342d34f`

SHA-256

```
openssl dgst -sha256 hash_sample.txt
```

Output:
`f2a3af0327329e7d3b74501f36cedf018f7b3126c28647d9eae221142503bc4c`

SHA-512

```
openssl dgst -sha512 hash_sample.txt
```

Output:
`8544fe3dfdaa96c8d4d645c5d22004b52fc787395c441e54eecf7d51d73c534b25df
bfe15ca1bc583260a06d13a1a07f52a4d9252c129ed45ac295f84b5fef1a`

Task 6: HMAC

Short Key

```
openssl dgst -md5 -hmac "mykey" hmac_test.txt
```

Output: `3729da33b418a9d4125c5e27c229b945`

```
openssl dgst -sha1 -hmac "mykey" hmac_test.txt
```

Output: `be3a0e2f20613fa47ccfe36c518823532a4e556b`

```
openssl dgst -sha256 -hmac "mykey" hmac_test.txt
```

Output:
`5394e084fbe91c6d398873a68293f1a393d86670285d8c9beed212148b1c68eb`

Long Key

```
openssl dgst -md5 -hmac
"thisisaverylongkeythatismuchlongerthanexpected" hmac_test.txt
```

Output: `6abda667b8eb6da1d66486a877ef25fa`

Very Short Key

```
openssl dgst -sha256 -hmac "a" hmac_test.txt
```

Output:
`58c9a50e26b503c80d39e106cb588b9cf8a8c28b36f1af05267f841857f750ca`

Observation: HMAC does not require a fixed key size. The key is internally hashed to match block size.

Task 7: File Hash Comparison

Original File

```
openssl dgst -md5 hash_original.txt > H1_md5.txt
openssl dgst -sha256 hash_original.txt > H1_sha256.txt
```

Modified File

```
openssl dgst -md5 hash_modified.txt > H2_md5.txt
openssl dgst -sha256 hash_modified.txt > H2_sha256.txt
```

Bit Comparison (Python)

- **MD5:** 65/128 bits same (≈50.78%)
- **SHA-256:** 127/256 bits same (≈49.61%)

Observation: Even a small change in the file drastically changes the hash, demonstrating the **avalanche effect**.