Lab-3 Report — AES Modes of Operation (ECB, CFB, OFB)

Objective

To understand and implement AES-128 encryption in ECB, CFB, and OFB modes using OpenSSL,
observe the difference between block and stream modes, and analyze error propagation when ciphertext or image data is corrupted.

Task 1 — Generate Key and IV

Goal: Create a 16-byte random AES key and initialization vector (IV).

Commands:

```
openssl rand -hex 16 > key.hex
openssl rand -hex 16 > iv.hex
$key = (Get-Content key.hex -Raw).Trim()
$iv  = (Get-Content iv.hex  -Raw).Trim()
```

Sample Output:

Key: bfbb9a8bf514af6a18242454b498b00c
IV : aaf9a8115e5ec048b887848eadd9619b

Task 2 — Encrypt Plaintext File (ECB / CFB / OFB)

Goal: Encrypt a plaintext file using AES-128 in three different modes.

Commands:

"This is a sample message for AES mode tests." | Out-File -FilePath plain.txt -Encoding ascii -NoNewline

openssl enc -aes-128-ecb -K $key -in plain.txt -out enc_ecb.bin
openssl enc -aes-128-cfb -K $key -iv $iv -in plain.txt -out enc_cfb.bin
openssl enc -aes-128-ofb -K $key -iv $iv -in plain.txt -out enc_ofb.bin

Result Verification:

Get-Item .\enc_*.bin | Format-Table Name,Length

Observation:
All ciphertext files were successfully created.
ECB applies padding by default, while CFB and OFB operate as stream modes without padding.

Task 3 — Ciphertext Corruption Impact

Goal: Flip a single byte in each ciphertext to observe error propagation.

Helper Function & Commands:

```
function Corrupt-File {
  param([string]$In,[string]$Out,[int]$Offset=8,[byte]$Xor=0xFF)
  $b = [IO.File]::ReadAllBytes($In)
  $b[$Offset] = $b[$Offset] -bxor $Xor
  [IO.File]::WriteAllBytes($Out,$b)
}
```

Corrupt-File "enc_ecb.bin" "enc_ecb_corrupted.bin" 8
Corrupt-File "enc_cfb.bin" "enc_cfb_corrupted.bin" 8
Corrupt-File "enc_ofb.bin" "enc_ofb_corrupted.bin" 8

Decryption & Observation:

openssl enc -d -aes-128-ecb -K $key -in enc_ecb_corrupted.bin
openssl enc -d -aes-128-cfb -K $key -iv $iv -in enc_cfb_corrupted.bin
openssl enc -d -aes-128-ofb -K $key -iv $iv -in enc_ofb_corrupted.bin

Findings:

ECB: corruption affects only one block.

CFB: corruption partially spreads forward.

OFB: corruption continues through the stream.

Task 4 — Prepare Image for Encryption

Goal: Split BMP file into header (54 bytes) and body.

Commands:

```
$bytes = [IO.File]::ReadAllBytes("pic_original.bmp")
[IO.File]::WriteAllBytes("header.bin", $bytes[0..53])
[IO.File]::WriteAllBytes("body.bin",    $bytes[54..($bytes.Length-1)])
```

Validation:

```
(Get-Item header.bin).Length
(Get-Item body.bin).Length
```

Result:
Header = 54 bytes, Body ≈ 151,680 bytes.

Task 5 — Encrypt Image Body (ECB / CFB / OFB)

Goal: Encrypt `body.bin` using AES and reconstruct encrypted images.

Commands:

```
openssl enc -aes-128-ecb -K $key -in body.bin -out body_ecb.bin -nopad
openssl enc -aes-128-cfb -K $key -iv $iv -in body.bin -out body_cfb.bin
```

```
openssl enc -aes-128-ofb -K $key -iv $iv -in body.bin -out body_ofb.bin

$hdr = [IO.File]::ReadAllBytes("header.bin")
$ecb = [IO.File]::ReadAllBytes("body_ecb.bin")
$cfb = [IO.File]::ReadAllBytes("body_cfb.bin")
$ofb = [IO.File]::ReadAllBytes("body_ofb.bin")

[IO.File]::WriteAllBytes("image_ecb.bmp", $hdr + $ecb)
[IO.File]::WriteAllBytes("image_cfb.bmp", $hdr + $cfb)
[IO.File]::WriteAllBytes("image_ofb.bmp", $hdr + $ofb)
```

Observation:

ECB: clear block patterns visible.

CFB / OFB: encrypted images appear random with no visible structure.

Task 6 — Image Corruption and Error Propagation

Goal: Modify one byte in each encrypted image body and observe visual impact.

Commands:

```
Corrupt-File "body_ecb.bin" "body_ecb_corrupted.bin" 1000
Corrupt-File "body_cfb.bin" "body_cfb_corrupted.bin" 1000
Corrupt-File "body_ofb.bin" "body_ofb_corrupted.bin" 1000

$hdr = [IO.File]::ReadAllBytes("header.bin")
[IO.File]::WriteAllBytes("image_ecb_corrupted.bmp", $hdr + (Get-Content
body_ecb_corrupted.bin -Encoding Byte))
[IO.File]::WriteAllBytes("image_cfb_corrupted.bmp", $hdr + (Get-Content
body_cfb_corrupted.bin -Encoding Byte))
[IO.File]::WriteAllBytes("image_ofb_corrupted.bmp", $hdr + (Get-Content
body_ofb_corrupted.bin -Encoding Byte))
```

Result :

Mode	Corruption Spread	Visual Effect
ECB	Single block only	Small localized distortion
CFB	Nearby blocks affected	Partial noise
OFB	Continuous propagation	Random streak through image

Task 7 — Verification and Integrity Check

Goal: Verify headers remain identical while body hashes differ after corruption.

Commands:

```
Get-FileHash .\body_ecb.bin -Algorithm SHA256
Get-FileHash .\body_ecb_corrupted.bin -Algorithm SHA256

$hdr1 = [IO.File]::ReadAllBytes(".\image_ecb.bmp")[0..53]
$hdr2 = [IO.File]::ReadAllBytes(".\image_ecb_corrupted.bmp")[0..53]
[System.Collections.StructuralComparisons]::StructuralEqualityComparer.Equals($hdr1,$hdr2)
```

Result Example:

```
True   # Header identical
Different SHA256 hashes for full images
```