Assignment – 1

AES overview :

AES is a symmetric block cipher standarized by NIST in 2001. It is used thourgh out the world for securing sensitive data by replacing the older Data Encryption Standard.

It supports key sizes of 128, 192, 256 bit keys. AES inloves a number of rounds which implements a same set of operation again and again to produce a ciphertext at the end.

10 rounds ----- 128 bit keys
12 rounds ----- 192 bit keys
14 rounds ----- 256 bit keys

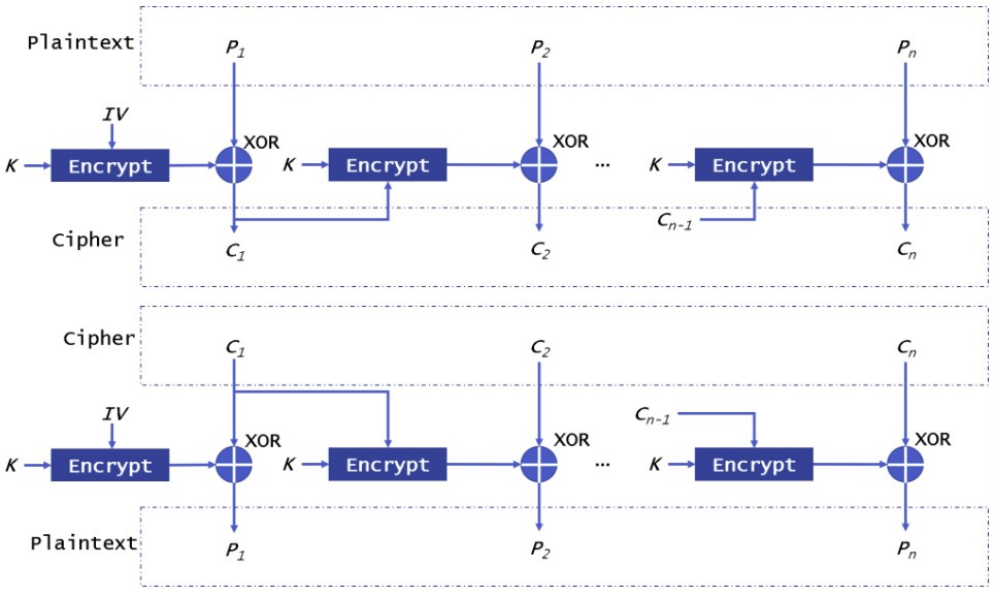
Each round of AES contains the below main steps :

1) SubBytes
2) Shift Rows
3) Mix Columns
4) AddRoundKey

The last round of AES does not do the mix columns step. The decryption use the inverse operation of all the above steps.

AES CFB Mode :

Cipher Feeback Mode is the one of the way we can use the AES. Cipher FeedBack (CFB) mode converts a block cipher into a stream cipher by allowing the encryption of individual bytes rather than fixed-size blocks. It uses the previous ciphertext segment as the input for the encryption process, generating a keystream that is XORed with the plaintext to produce the ciphertext.



In CFB mode, each block of ciphertext depends on the one that came before it (or the IV for the first block). This means the blocks are chained together. If one block of ciphertext is changed, it messes up the next block when you try to decrypt. Unlike ECB, which works on blocks separately and can show patterns, CFB hides those patterns by linking everything. The downside is that encryption has to be done one block at a time. Decryption, though, can be done in parallel since all the ciphertext blocks are already there.

CFB mode needs an Initialization Vector (IV) to get started, since there's no previous ciphertext for the very first block. The IV is first encrypted, then combined (using XOR) with the first block of plaintext to create the first block of ciphertext. For security, the IV has to be unique and unpredictable each time you encrypt. If you reuse the same IV with the same key, attackers can spot patterns in the ciphertext and might even figure out parts of the plaintext by comparing different encrypted messages. The IV doesn't have to stay secret, but it's critical that it's never reused.

CFB mode handles errors better than CBC in some cases. If a single bit of ciphertext gets corrupted during transmission, only the same bit in the corresponding plaintext block will be affected. The next plaintext block will be completely corrupted, but the error won't spread beyond that. This makes CFB more resistant to transmission errors than CBC, where a single error can mess up many blocks.

Output of programin execution :

```
sudharsanan@spongebob:~/Desktop/TPM-Assesment$ python3 main.py
AES-128 CFB Mode Encryption/Decryption (from scratch)
Enter plaintext: I am Sudharsanan K
Enter 128-bit key (hex, 32 chars): 2b7e151628aed2a6abf7158809cf4f3c
Enter IV (hex, 32 chars) [leave blank to auto-generate]:
Generated IV (hex): 088d38d3633f06000000000000000000
Ciphertext (hex): 56594ed72c7793902314cfb116f1295fcae09d06e1a5ce26a1a004476d353a73
Recovered Plaintext: I am Sudharsanan K
Success: Plaintext recovered correctly!
sudharsanan@spongebob:~/Desktop/TPM-Assesment$
```

Performance Measurement :

```
sudharsanan@spongebob:~/Desktop/TPM-Assesment$ python3 performance_analysis.py
AES-128 CFB Performance Analysis
Plaintext size: 1048576 bytes

Results:
+-------------------+-------------------+
| Operation         | Time (seconds)    |
+-------------------+-------------------+
| Encryption        | 3.915621          |
| Decryption        | 3.929945          |
+-------------------+-------------------+
Ciphertext size: 1048576 bytes

Table for report:
| Operation   | Time (s)   | Size (bytes) |
| Encrypt     | 3.915621 | 1048576        |
| Decrypt     | 3.929945 | 1048576        |

Check 'ciphertext.bin' and 'recovered.txt' for output files.
sudharsanan@spongebob:~/Desktop/TPM-Assesment$
```