

# BLOCK CIPHER MODES OF OPERATION

In this chapter, we will discuss the different modes of operation of a block cipher. These are procedural rules for a generic block cipher. Interestingly, the different modes result in different properties being achieved which add to the security of the underlying block cipher.

A block cipher processes the data blocks of fixed size. Usually, the size of a message is larger than the block size. Hence, the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time.

## Electronic Code Book *ECB* Mode

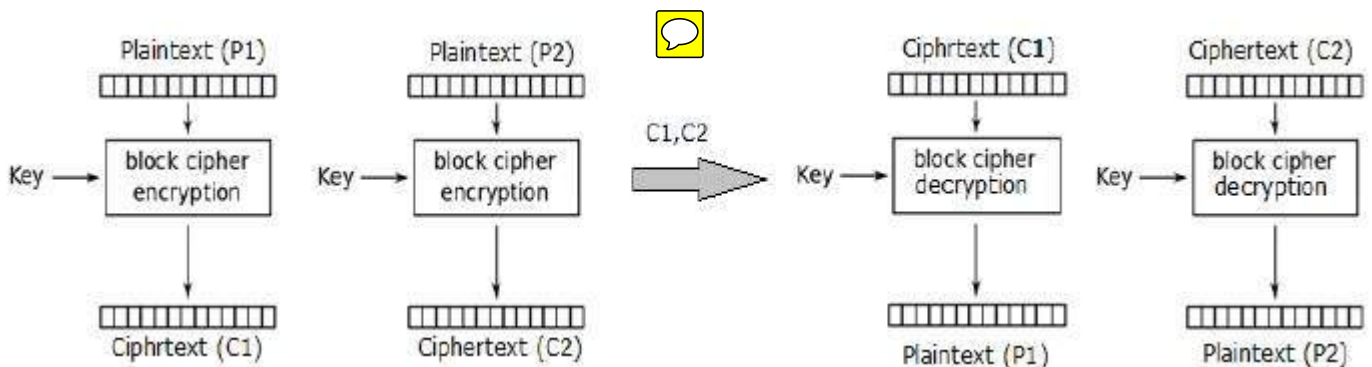
This mode is a most straightforward way of processing a series of sequentially listed message blocks.

### Operation

- The user takes the first block of plaintext and encrypts it with the key to produce the first block of ciphertext.
- He then takes the second block of plaintext and follows the same process with same key and so on so forth.

The ECB mode is **deterministic**, that is, if plaintext block  $P_1, P_2, \dots, P_m$  are encrypted twice under the same key, the output ciphertext blocks will be the same.

In fact, for a given key technically we can create a codebook of ciphertexts for all possible plaintext blocks. Encryption would then entail only looking up for required plaintext and select the corresponding ciphertext. Thus, the operation is analogous to the assignment of code words in a codebook, and hence gets an official name – Electronic Codebook mode of operation *ECB*. It is illustrated as follows –



## Analysis of ECB Mode

In reality, any application data usually have partial information which can be guessed. For example, the range of salary can be guessed. A ciphertext from ECB can allow an attacker to guess the plaintext by trial-and-error if the plaintext message is within predictable.

For example, if a ciphertext from the ECB mode is known to encrypt a salary figure, then a small number of trials will allow an attacker to recover the figure. In general, we do not wish to use a deterministic cipher, and hence the ECB mode should not be used in most applications.

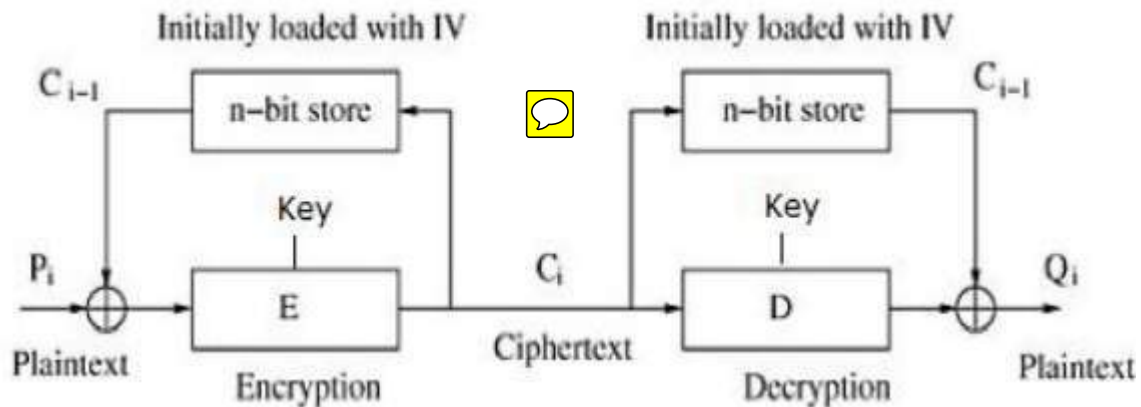
## Cipher Block Chaining *CBC* Mode

CBC mode of operation provides message dependence for generating ciphertext and makes the system non-deterministic.

### Operation

The operation of CBC mode is depicted in the following illustration. The steps are as follows –

- Load the n-bit Initialization Vector  $IV$  in the top register.
- XOR the n-bit plaintext block with data value in top register.
- Encrypt the result of XOR operation with underlying block cipher with key  $K$ .
- Feed ciphertext block into top register and continue the operation till all plaintext blocks are processed.
- For decryption,  $IV$  data is XORed with first ciphertext block decrypted. The first ciphertext block is also fed into to register replacing  $IV$  for decrypting next ciphertext block.



## Analysis of CBC Mode

In CBC mode, the current plaintext block is added to the previous ciphertext block, and then the result is encrypted with the key. Decryption is thus the reverse process, which involves decrypting the current ciphertext and then adding the previous ciphertext block to the result.

Advantage of CBC over ECB is that changing  $IV$  results in different ciphertext for identical message. On the drawback side, the error in transmission gets propagated to few further block during decryption due to chaining effect.

It is worth mentioning that CBC mode forms the basis for a well-known data origin authentication mechanism. Thus, it has an advantage for those applications that require both symmetric encryption and data origin authentication.

## Cipher Feedback CFB Mode

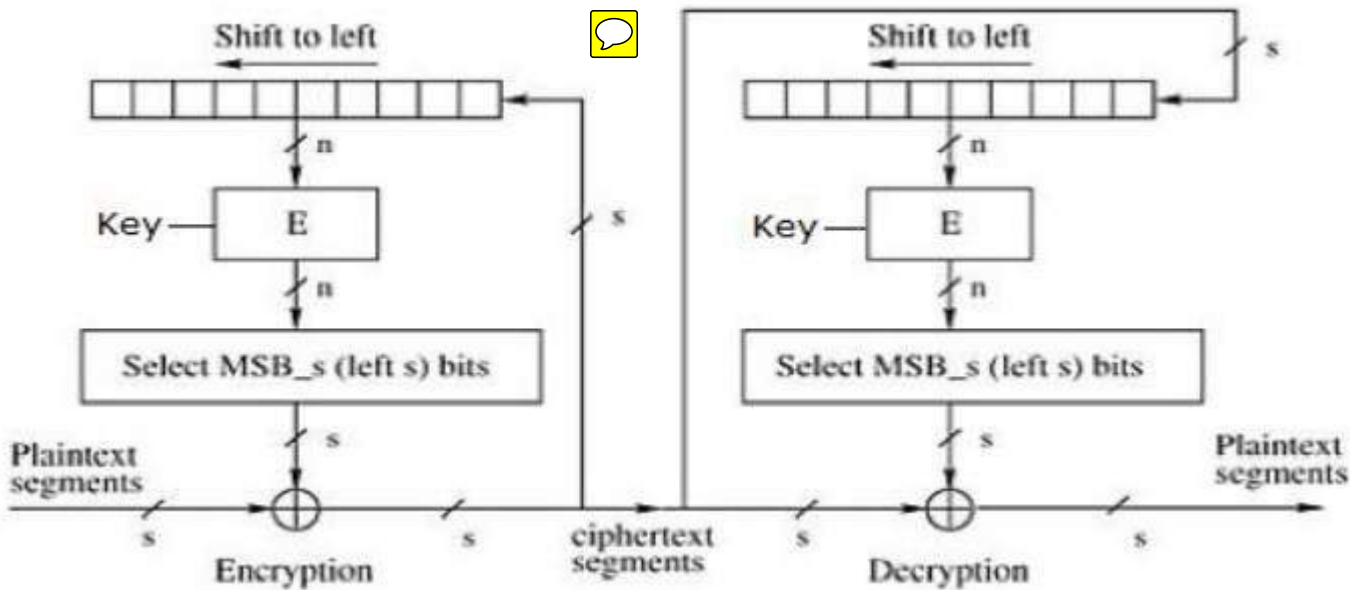
In this mode, each ciphertext block gets 'fed back' into the encryption process in order to encrypt the next plaintext block.

## Operation

The operation of CFB mode is depicted in the following illustration. For example, in the present system, a message block has a size ' $s$ ' bits where  $1 < s < n$ . The CFB mode requires an initialization vector  $IV$  as the initial random n-bit input block. The  $IV$  need not be secret. Steps of operation are –

- Load the  $IV$  in the top register.
- Encrypt the data value in top register with underlying block cipher with key  $K$ .
- Take only ' $s$ ' number of most significant bits *leftbits* of output of encryption process and XOR them with ' $s$ ' bit plaintext message block to generate ciphertext block.
- Feed ciphertext block into top register by shifting already present data to the left and continue the operation till all plaintext blocks are processed.
- Essentially, the previous ciphertext block is encrypted with the key, and then the result is XORed to the current plaintext block.

- Similar steps are followed for decryption. Pre-decided IV is initially loaded at the start of decryption.



## Analysis of CFB Mode

CFB mode differs significantly from ECB mode, the ciphertext corresponding to a given plaintext block depends not just on that plaintext block and the key, but also on the previous ciphertext block. In other words, the ciphertext block is dependent of message.

CFB has a very strange feature. In this mode, user decrypts the ciphertext using only the encryption process of the block cipher. The decryption algorithm of the underlying block cipher is never used.

Apparently, CFB mode is converting a block cipher into a type of stream cipher. The encryption algorithm is used as a key-stream generator to produce key-stream that is placed in the bottom register. This key stream is then XORed with the plaintext as in case of stream cipher.

By converting a block cipher into a stream cipher, CFB mode provides some of the advantageous properties of a stream cipher while retaining the advantageous properties of a block cipher.

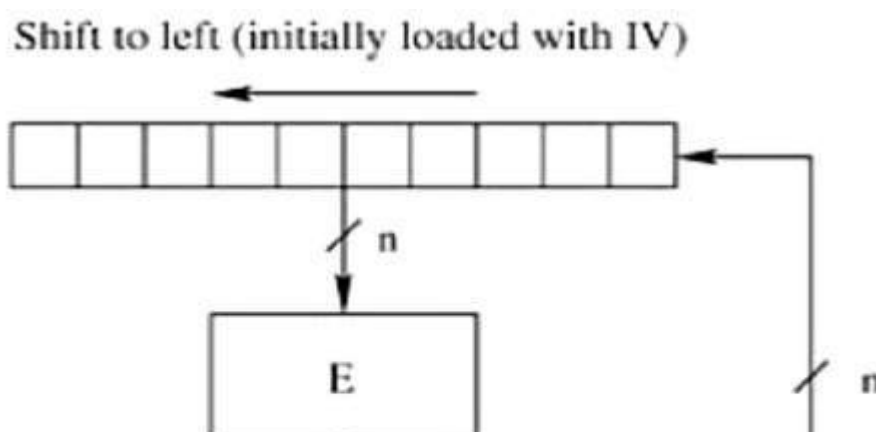
On the flip side, the error of transmission gets propagated due to changing of blocks.

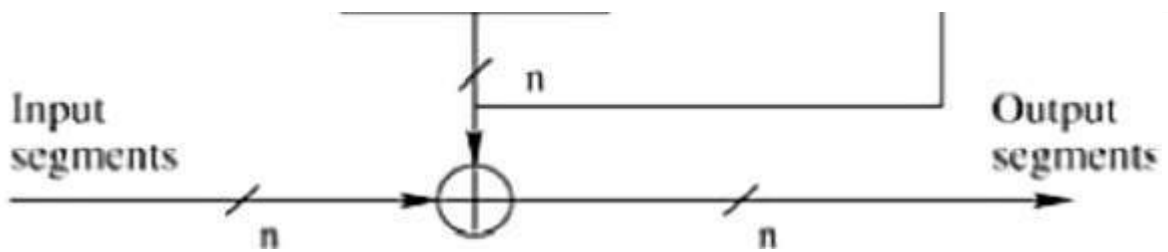
## Output Feedback OFB Mode

It involves feeding the successive output blocks from the underlying block cipher back to it. These feedback blocks provide string of bits to feed the encryption algorithm which act as the key-stream generator as in case of CFB mode.

The key stream generated is XOR-ed with the plaintext blocks. The OFB mode requires an IV as the initial random n-bit input block. The IV need not be secret.

The operation is depicted in the following illustration –





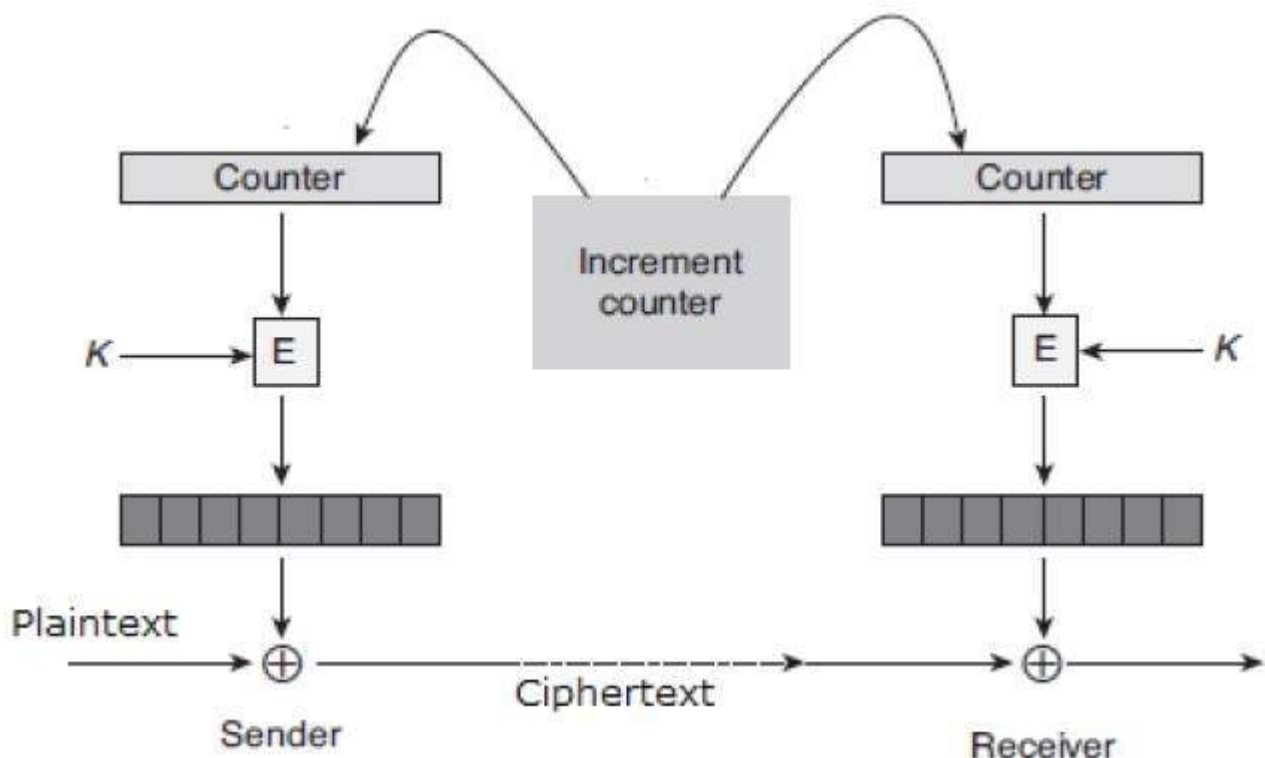
## Counter CTR Mode

It can be considered as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a ciphertext block is exchanged. This shared counter is not necessarily a secret value, but challenge is that both sides must keep the counter synchronized.

## Operation

Both encryption and decryption in CTR mode are depicted in the following illustration. Steps in operation are –

- Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB and CBC mode.
- Encrypt the contents of the counter with the key and place the result in the bottom register.
- Take the first plaintext block P1 and XOR this to the contents of the bottom register. The result of this is C1. Send C1 to the receiver and update the counter. The counter update replaces the ciphertext feedback in CFB mode.
- Continue in this manner until the last plaintext block has been encrypted.
- The decryption is the reverse process. The ciphertext block is XORed with the output of encrypted contents of counter value. After decryption of each ciphertext block counter is updated as in case of encryption.



## Analysis of Counter Mode

It does not have message dependency and hence a ciphertext block does not depend on the previous plaintext blocks.

Like CFB mode, CTR mode does not involve the decryption process of the block cipher. This is because the CTR mode is really using the block cipher to generate a key-stream, which is encrypted using the XOR function. In other words, CTR mode also converts a block cipher to a stream cipher.

The serious disadvantage of CTR mode is that it requires a synchronous counter at sender and receiver. Loss of synchronization leads to incorrect recovery of plaintext.

However, CTR mode has almost all advantages of CFB mode. In addition, it does not propagate error of transmission at all.

Loading [Mathjax]/jax/output/HTML-CSS/jax.js