

# Encryption and Decryption Steps in modes of operations



Explain Encryption and decryption steps of ECB mode (dont give me images)

## ECB (Electronic Codebook) Mode - Encryption & Decryption Steps

ECB is the simplest block cipher mode. Each block is processed **independently** using the same secret key.

### Encryption Steps (ECB)

Let:

- Plaintext =  $P$
- Block size = fixed (e.g., 128 bits for AES)
- Secret key =  $K$
- Block cipher encryption function =  $E_K(\cdot)$

#### Step 1: Divide Plaintext into Blocks

Split the plaintext into equal-sized blocks:

$$P = P_1, P_2, P_3, \dots, P_n$$

(If last block is shorter, apply padding.)

#### Step 2: Encrypt Each Block Independently

Each block is encrypted separately using the same key:

$$C_i = E_K(P_i)$$

for  $i = 1$  to  $n$ .

#### Step 3: Concatenate Ciphertext Blocks

Final ciphertext:

$$C = C_1 \| C_2 \| C_3 \| \dots \| C_n$$

- ✓ No IV (Initialization Vector) is used
- ✓ No chaining between blocks

## Decryption Steps (ECB)

Let:

- Ciphertext =  $C_1, C_2, \dots, C_n$
- Block cipher decryption function =  $D_K(\cdot)$

### Step 1: Divide Ciphertext into Blocks

Split ciphertext into same block size:

$$C = C_1, C_2, \dots, C_n$$

### Step 2: Decrypt Each Block Independently

$$P_i = D_K(C_i)$$

for each block.

### Step 3: Remove Padding

If padding was applied during encryption, remove it.

## Important Properties

- Identical plaintext blocks → Identical ciphertext blocks
- Parallelizable (encryption & decryption)
- No error propagation between blocks
- Not semantically secure (patterns leak)

## One-Line Summary

ECB encrypts each block **independently with the same key**, without chaining or IV, making it simple but insecure for most practical uses.



Explain Encryption and decryption steps of CBC mode

# CBC (Cipher Block Chaining) Mode - Encryption & Decryption Steps

Let:

- Plaintext blocks:  $P_1, P_2, \dots, P_n$
- Ciphertext blocks:  $C_1, C_2, \dots, C_n$
- Secret key:  $K$
- Block cipher encryption:  $E_K(\cdot)$
- Block cipher decryption:  $D_K(\cdot)$
- Initialization Vector:  $IV$  (random & unpredictable)

## Encryption Steps (CBC)

### Step 1: Divide Plaintext into Blocks

Split plaintext into equal-size blocks (pad last block if needed).

### Step 2: First Block Encryption

First plaintext block is XORed with IV:

$$C_1 = E_K(P_1 \oplus IV)$$

### Step 3: Remaining Blocks Encryption

Each plaintext block is XORed with the **previous ciphertext block**, then encrypted:

$$C_i = E_K(P_i \oplus C_{i-1})$$

for  $i = 2$  to  $n$

### Step 4: Concatenate All Ciphertext Blocks

Final ciphertext:

$$C = IV \| C_1 \| C_2 \| \dots \| C_n$$

- ✓ IV must be sent with ciphertext
- ✓ Same plaintext blocks produce different ciphertext (due to chaining)

## Decryption Steps (CBC)

## Step 1: Extract IV

Read IV from the beginning of ciphertext.

## Step 2: First Block Decryption

$$P_1 = D_K(C_1) \oplus IV$$

## Step 3: Remaining Blocks Decryption

$$P_i = D_K(C_i) \oplus C_{i-1}$$

for  $i = 2$  to  $n$

## Step 4: Remove Padding

Remove padding added during encryption.

## ⚠️ Important Properties

- Each block depends on previous ciphertext block
- Encryption is sequential (not parallelizable)
- Decryption can be parallelized
- Error in one block affects two blocks
- Requires unpredictable IV for security

## One-Line Summary

CBC mode encrypts each plaintext block by XORing it with the previous ciphertext block before encryption, providing chaining-based security against pattern leakage.



Explain Encryption and decryption steps of CFB mode

## 🔒 CFB (Cipher Feedback) Mode – Encryption & Decryption Steps

Let:

- Plaintext blocks:  $P_1, P_2, \dots, P_n$

- Ciphertext blocks:  $C_1, C_2, \dots, C_n$
- Secret key:  $K$
- Block cipher encryption function:  $E_K(\cdot)$
- Initialization Vector:  $IV$

 In CFB mode, **only the encryption function  $E_K$**  is used for both encryption and decryption.

## Encryption Steps (CFB)

### Step 1: Divide Plaintext into Blocks

Split plaintext into block-size segments (padding usually not required if using full-block CFB).

### Step 2: First Block Encryption

1. Encrypt the IV:

$$E_K(IV)$$

2. XOR the result with the first plaintext block:

$$C_1 = P_1 \oplus E_K(IV)$$

### Step 3: Remaining Blocks

For  $i = 2$  to  $n$ :

1. Encrypt previous ciphertext block:

$$E_K(C_{i-1})$$

2. XOR with current plaintext:

$$C_i = P_i \oplus E_K(C_{i-1})$$

### Step 4: Send IV + Ciphertext

$$C = IV \| C_1 \| C_2 \| \dots \| C_n$$

## Decryption Steps (CFB)

 Note: Block cipher **decryption function is NOT used**.

### Step 1: Extract IV

Read IV from the start of ciphertext.

## Step 2: First Block Decryption

1. Encrypt the IV:

$$E_K(IV)$$

2. XOR with first ciphertext block:

$$P_1 = C_1 \oplus E_K(IV)$$

## Step 3: Remaining Blocks

For  $i = 2$  to  $n$ :

1. Encrypt previous ciphertext block:

$$E_K(C_{i-1})$$

2. XOR with current ciphertext:

$$P_i = C_i \oplus E_K(C_{i-1})$$

## ⚠️ Important Properties

- Converts block cipher into a **stream cipher**
- No separate decryption algorithm needed
- Encryption is sequential
- Small error affects only current and next block
- Requires unpredictable IV

## One-Line Summary

CFB mode encrypts the previous ciphertext (or IV) and XORs it with plaintext to produce ciphertext, using only the block cipher's encryption function.



Explain Encryption and decryption steps of OFB mode

## 🔒 OFB (Output Feedback) Mode - Encryption & Decryption Steps

Let:

- Plaintext blocks:  $P_1, P_2, \dots, P_n$
- Ciphertext blocks:  $C_1, C_2, \dots, C_n$
- Secret key:  $K$
- Block cipher encryption function:  $E_K(\cdot)$
- Initialization Vector:  $IV$

 OFB turns a block cipher into a **synchronous stream cipher**.

 Only the encryption function  $E_K$  is used in both encryption and decryption.

## Encryption Steps (OFB)

### Step 1: Divide Plaintext into Blocks

Split plaintext into equal-size blocks (padding generally not required for full-block mode).

### Step 2: Generate First Keystream Block

$$O_1 = E_K(IV)$$

### Step 3: Encrypt First Block

$$C_1 = P_1 \oplus O_1$$

### Step 4: Generate Subsequent Keystream Blocks

For  $i = 2$  to  $n$ :

$$O_i = E_K(O_{i-1})$$

### Step 5: Encrypt Remaining Blocks

$$C_i = P_i \oplus O_i$$

### Step 6: Send IV + Ciphertext

$$C = IV \| C_1 \| C_2 \| \dots \| C_n$$

## Decryption Steps (OFB)

 Decryption is identical to encryption process.

## Step 1: Extract IV

Read IV from ciphertext.

---

## Step 2: Regenerate Same Keystream

$$O_1 = E_K(IV)$$

$$O_i = E_K(O_{i-1})$$

---

## Step 3: Recover Plaintext

$$P_i = C_i \oplus O_i$$

---

## ⚠ Important Properties

- Keystream is independent of plaintext and ciphertext
- Encryption and decryption are identical
- Errors affect only corresponding bit (no propagation)
- IV must be unique (reuse breaks security completely)
- Not self-synchronizing

---

## One-Line Summary

OFB generates a keystream by repeatedly encrypting the IV and XORing it with plaintext or ciphertext, making encryption and decryption identical operations.



Explain Encryption and decryption steps of CTR mode

## 🔒 CTR (Counter) Mode – Encryption & Decryption Steps

Let:

- Plaintext blocks:  $P_1, P_2, \dots, P_n$
- Ciphertext blocks:  $C_1, C_2, \dots, C_n$
- Secret key:  $K$
- Block cipher encryption function:  $E_K(\cdot)$
- Nonce (unique value):  $N$

- Counter value:  $ctr$

 CTR converts a block cipher into a stream cipher.

 Only the encryption function  $E_K$  is used for both encryption and decryption.

## Encryption Steps (CTR)

### Step 1: Divide Plaintext into Blocks

Split plaintext into block-size segments (padding usually not required).

### Step 2: Construct Counter Blocks

For each block  $i$ :

$$Counter_i = N \| ctr_i$$

- $N$  = fixed nonce (must be unique per session/message)
- $ctr_i$  = incrementing counter value (0,1,2,3,...)

### Step 3: Generate Keystream Blocks

$$KS_i = E_K(Counter_i)$$

### Step 4: Encrypt Plaintext

$$C_i = P_i \oplus KS_i$$

### Step 5: Transmit

$$C = N \| C_1 \| C_2 \| \dots \| C_n$$

(Nonce is sent along with ciphertext.)

## Decryption Steps (CTR)

 Decryption is identical to encryption.

### Step 1: Reconstruct Counter Blocks

Using received nonce  $N$ :

$$Counter_i = N \| ctr_i$$

## Step 2: Regenerate Same Keystream

$$KS_i = E_K(Counter_i)$$

## Step 3: Recover Plaintext

$$P_i = C_i \oplus KS_i$$

## ⚠️ Important Properties

- Encryption and decryption are identical
- Fully parallelizable (very fast)
- No padding required
- Error affects only corresponding bits
- ! Nonce must never be reused with same key (reuse breaks security completely)

## One-Line Summary

CTR mode encrypts incrementing counter values to produce a keystream and XORs it with plaintext or ciphertext, making encryption and decryption identical and highly parallelizable.