Unit 2

Symmetric Ciphers

(10 Hours)

2.4. Feistel Cipher Structure, Substitution Permutation Network (SPN)

2.5. Data Encryption Standards (DES), Double DES, Triple DES

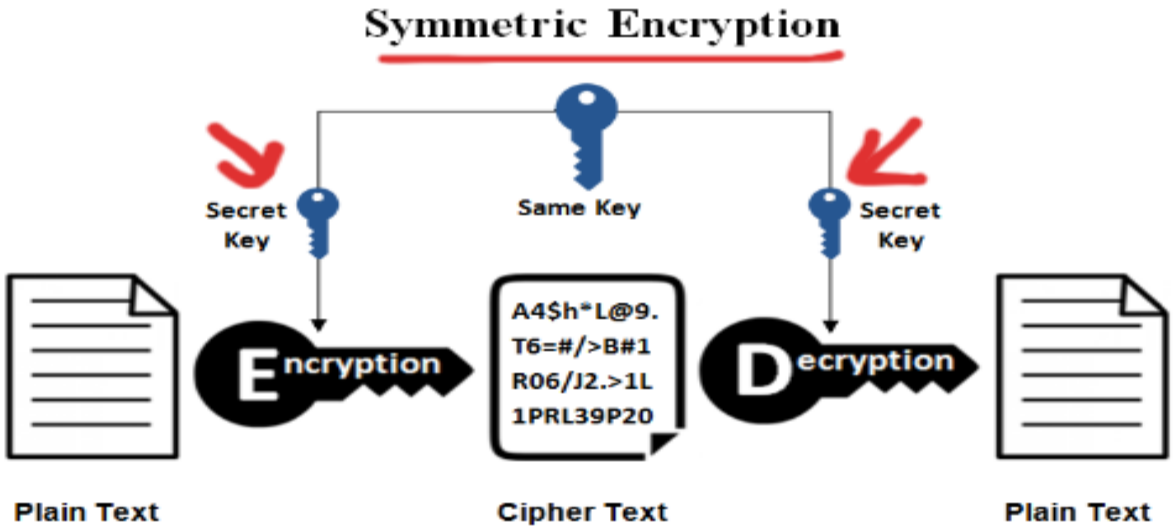
2.6. Finite Fields: Groups Rings, Fields, Modular Arithmetic, Euclidean Algorithm, Galois Fields (GF(p) & GF(2n)), Polynomial Arithmetic

2.7. International Data Encryption Standard (IDEA)

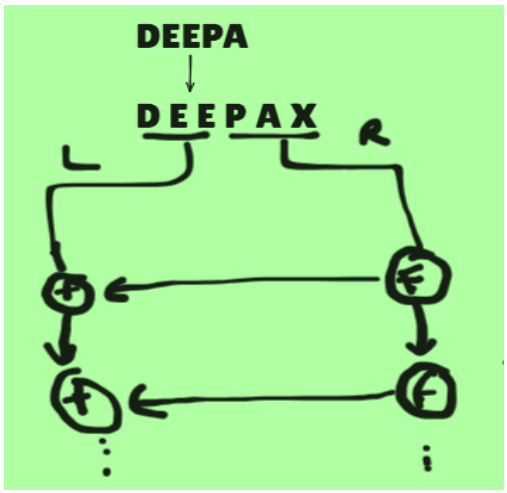
2.8. Advanced Encryption Standards (AES) Cipher

2.9. Modes of Block Cipher Encryptions (Electronic Code Book, Cipher Block Chaining, Cipher Feedback Mode, Output Feedback Mode, Counter Mode)

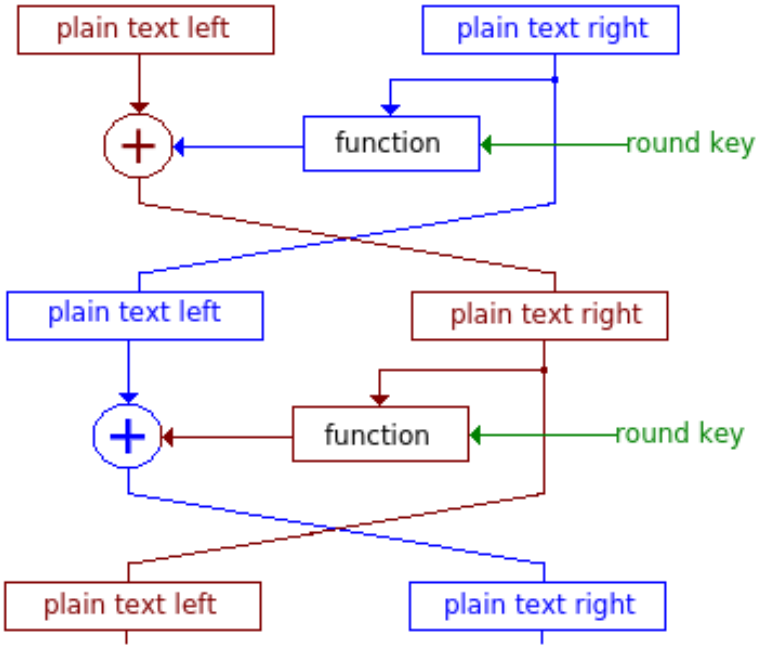
Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist



Feistel Cipher Structure



- Feistel Cipher is not a specific scheme of block cipher. It is a design model from which many different block ciphers are derived. DES is just one example of a Feistel Cipher.
- A cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption.
- Encryption Process
  - The encryption process uses the Feistel structure consisting multiple rounds of processing of the plaintext, each round consisting of a substitution step followed by a permutation step.



- The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half.
- In each round, the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function f that takes two input − the key K and R. The function produces the output f(R,K). Then, we XOR the output of the mathematical function with L.
- In real implementation of the Feistel Cipher, such as DES, instead of using the whole encryption key

during each round, a round-dependent key (a subkey) is derived from the encryption key. This means that each round uses a different key, although all these subkeys are related to the original key.

- The permutation step at the end of each round swaps the modified L and unmodified R. Therefore, the L for the next round would be R of the current round. And R for the next round be the output L of the current round.
- Above substitution and permutation steps form a round. The number of rounds are specified by the algorithm design.
- Once the last round is completed then the two sub blocks, R and L are concatenated in this order to form the ciphertext block.

The difficult part of designing a Feistel Cipher is selection of round function f. In order to be unbreakable scheme, this function needs to have several important properties that are beyond the scope of our discussion.

➢ **Decryption Process**

- The process of decryption in Feistel cipher is almost similar. Instead of starting with a block of plaintext, the ciphertext block is fed into the start of the Feistel structure and then the process thereafter is exactly the same as described in the given illustration.
- The process is said to be almost similar and not exactly same. In the case of decryption, the only difference is that the subkeys used in encryption are used in the reverse order.
- The final swapping of L and R in last step of the Feistel Cipher is essential. If these are not swapped then the resulting ciphertext could not be decrypted using the same algorithm.

➢ **Number of Rounds**

- The number of rounds used in a Feistel Cipher depends on desired security from the system. More number of rounds provide more secure system. But at the same time, more rounds mean the inefficient slow encryption and decryption processes. Number of rounds in the systems thus depend upon efficiencysecurity tradeoff.

Example

Feistel cipher round shown in the image using a **very small, clear binary example**, exactly as you asked:

- Text = **“mohan”**
- We will **take only one character: “m”**
- Convert **“m” to ASCII**
- Split into **Left (L) and Right (R)**

- Apply **XOR, function F, round key**
- Keep it **short and exam-friendly**

What the diagram represents (in words)

The diagram shows **two Feistel rounds**.
Each round does this:

Li = Ri-1

Ri = Li-1 XOR F(Ri-1, Ki)

Where:

- L = left half
- R = right half
- F = round function
- K = round key
- ⊕ = XOR (the "+" symbol in the diagram)

STEP 1 Take plaintext character: "m"

ASCII of **m**:

m = 109 (decimal)

m = 01101101 (binary, 8 bits)

STEP 2 Split into Left and Right halves

Split 8 bits into two 4-bit halves:

Plaintext (m) = 01101101

L0 = 0110

R0 = 1101

This matches **plain text left** and **plain text right** in the diagram.

STEP 3 Choose a round key (K1)

Assume a simple **4-bit round key**:

K1 = 1010

STEP 4 Define the round function F

For simplicity (exam/demo level), define:

F(R, K) = R XOR K

This is common in basic Feistel explanations.

STEP 5 ROUND 1 (as shown in diagram)

🔷 **Compute F(R0, K1)**

R0 = 1101

K1 = 1010

F = 0111 (XOR)

🔷 **Compute new Right half (R1)**

R1 = L0 XOR F(R0, K1)

L0 = 0110

F = 0111

R1 = 0001

🔷 **Compute new Left half (L1)**

L1 = R0 = 1101

✅ **Result after Round 1**

L1 = 1101

R1 = 0001

This exactly matches the **swap + XOR flow** shown in the diagram.

STEP 6 (Optional) ROUND 2 (very briefly)

Assume another key:

K2 = 0101

F(R1, K2)

R1 = 0001

K2 = 0101

F = 0100

New R2

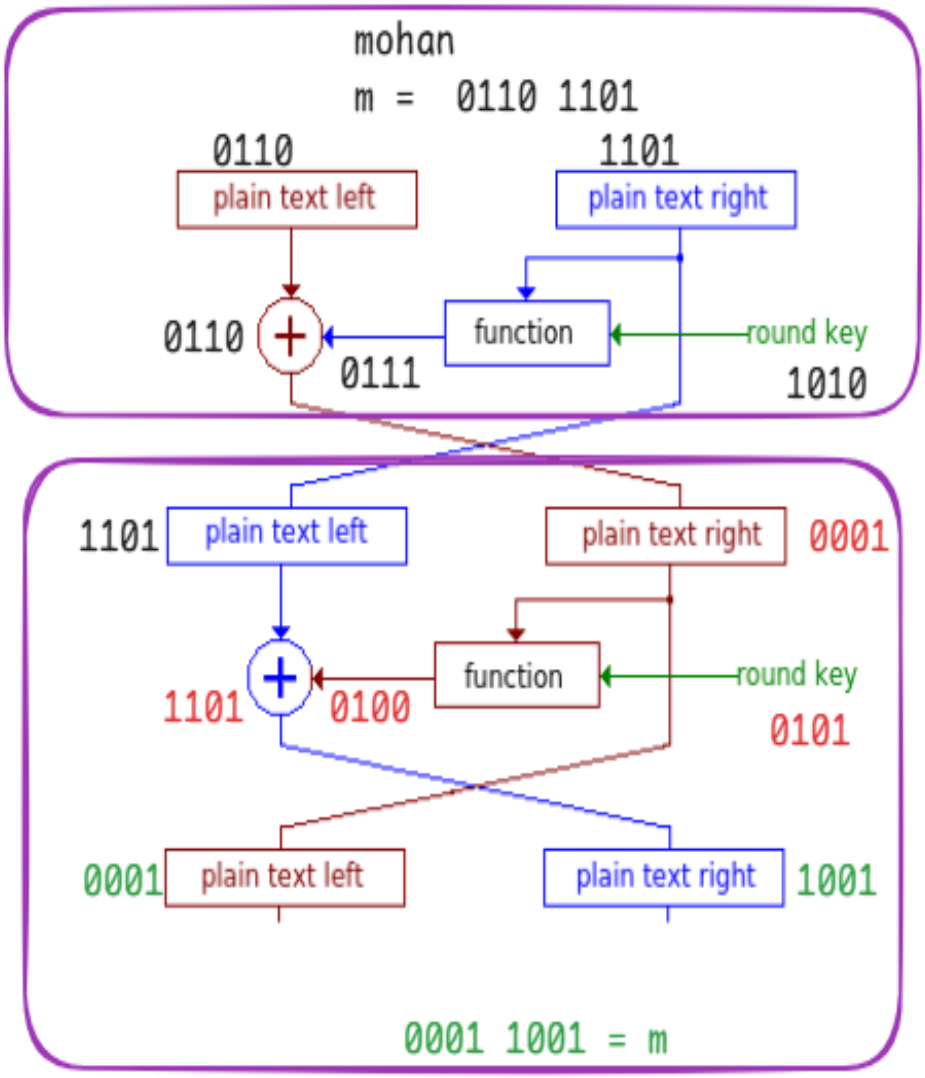
R2 = L1 XOR F

= 1101 XOR 0100

= 1001

New L2

L2 = R1 = 0001



STEP 7 Final output (after 2 rounds)

L2 R2 = 0001 1001

This is the **Feistel output** for character **"m"**.

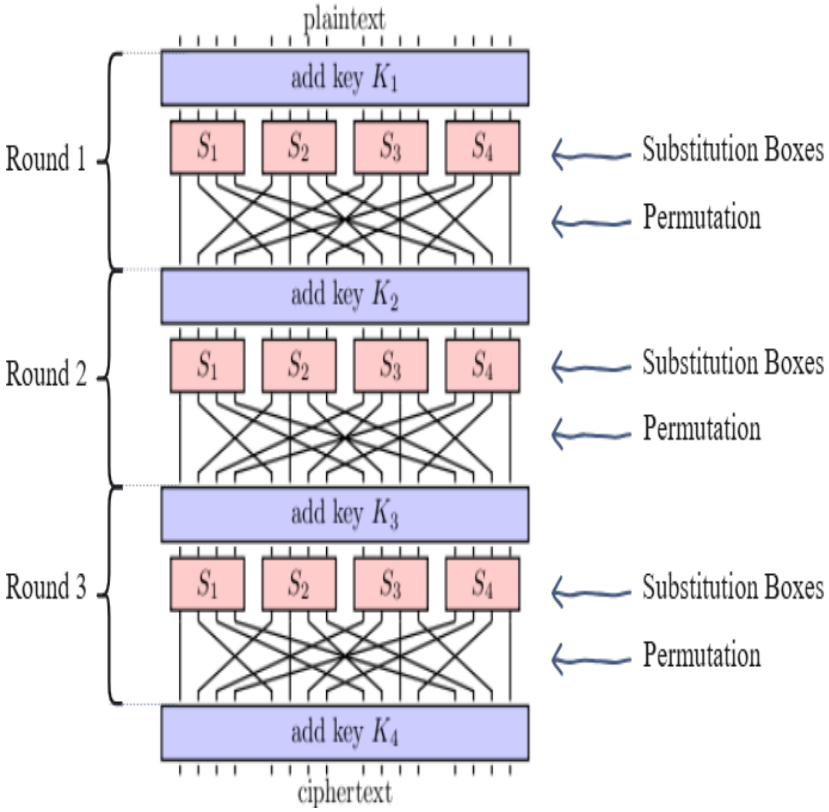
Why decryption works (important exam point)

- Same structure is used
- Keys are applied **in reverse order**
- XOR is reversible

That's why **Feistel networks are invertible even if F is not**

Substitution Permutation Network (SPN)

A **Substitution-Permutation network** or an **SP network** is a class of block ciphers that consist of rounds of a repeated series of mathematical operations. SP networks form the basis of the infamous AES algorithm.



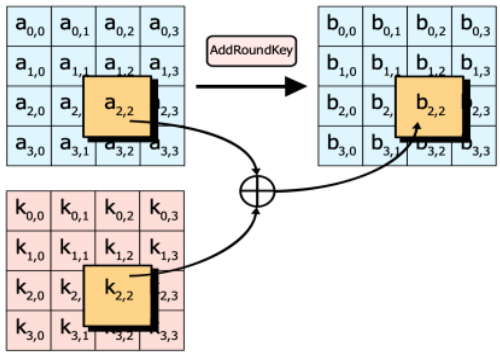
Operations in an SP network

As the figure above shows, the plaintext is passed to an SP network to produce a ciphertext. This is done through rounds, each of which consists of three main operations:

1. **Addition of the round key**
2. **Substitution of bits**
3. **Permutation of bits**

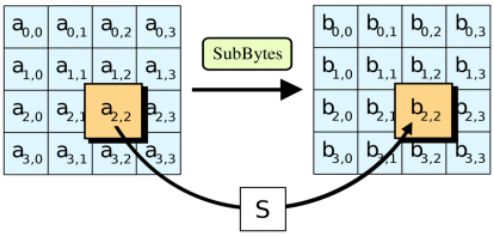
Addition of the round key

Each round in an SP network has a round key. **Round keys** are retrieved from the expansion of one secret key passed to the network at the start. At the start of each round, the text is XORed with the respective round key. This ensures that the ciphertext can only be decrypted by someone who has the round keys.

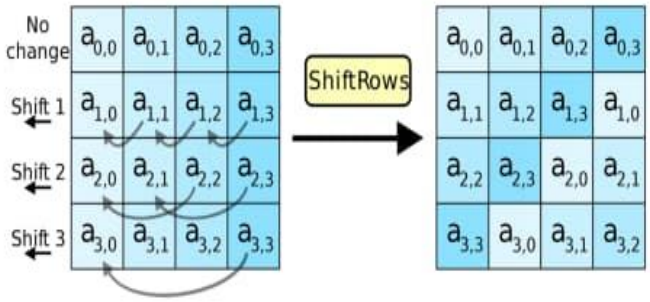


Substitution of bits

Next, the bits of the text are substituted among themselves. Since SP networks are used in block ciphers, the text is arranged as blocks. The block text bytes are substituted based on rules dictated by predefined S-boxes.



Finally, comes the permutation step. In this step, bits in the block text are mixed around. One example of such mixing is in AES, where all rows except the first are shifted by one. This is shown below:



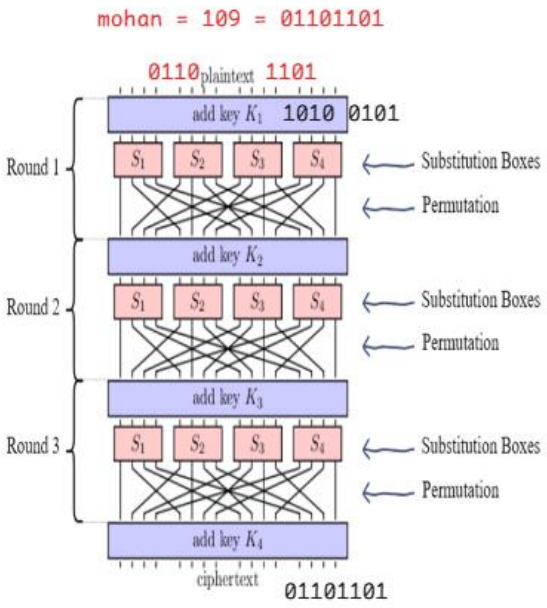
Note: The substitution and permutation boxes are not kept hidden in SP networks. Only the round keys are kept secret for security.

Example

Substitution–Permutation Network (SPN).

It will use:

- Plaintext from **"mohan" → only 'm'**
- 8-bit example (toy SPN)
- 3 rounds
- Simple S-boxes + permutation
- XOR for "add key"



SPN (Substitution–Permutation Network) — SAMPLE WALKTHROUGH

Step 0 Plaintext

Take **only "m"** from *mohan*.

m (ASCII) = 109

Binary = 01101101

This is the **plaintext** at the top of the diagram.

Step 1 Split into 4-bit nibbles

Plaintext = 0110 1101

P1 P2

Step 2 Add Round Key K_1 (XOR)

Assume **Round Key K_1 = 1010 0101**

Plaintext : 0110 1101

Key K_1 : 1010 0101

After XOR : 1100 1000

This matches **"add key K_1"** in the diagram.

Step 3 Substitution (S-Boxes)

Assume **simple S-box** (toy example):

Input → Output

0000 → 1110

0001 → 0100

0010 → 1101

0011 → 0001

0100 → 0010

0101 → 1111

0110 → 1011

0111 → 1000

1000 → 0011

1001 → 1010

1010 → 0110

1011 → 1100

1100 → 0101

1101 → 1001

1110 → 0000

1111 → 0111

Apply S-boxes:

1100 → 0101

1000 → 0011

After substitution:

0101 0011

This is **S_1 S_2 S_3 S_4** in the diagram.

Step 4 Permutation (P-Box)

Assume permutation rule:

Bit positions: 1 2 3 4 5 6 7 8

Permuted as : 2 6 3 1 4 8 5 7

Apply to 01010011:

Before : 0 1 0 1 0 0 1 1

After : 1 0 0 0 1 1 0 1

Result:

10001101

This matches **Permutation layer** in the diagram.

🔁 ROUND 2 (same structure)

Add Key K_2

Assume:

K_2 = 0011 1100

10001101

XOR 00111100

10110001

Substitution

1011 → 1100

0001 → 0100

Result:

1100 0100

Permutation

After P-box:

01010100

🔁 **ROUND 3 (final round)**

Add Key K_3

K_3 = 1111 0000

01010100

XOR 11110000

10100100

Substitution

1010 → 0110

0100 → 0010

Result:

0110 0010

Step 5 Final Add Key K_4 (last layer)

K_4 = 0001 1111

01100010

XOR 00011111

01111101

✅ **FINAL CIPHERTEXT**

01111101 (binary)

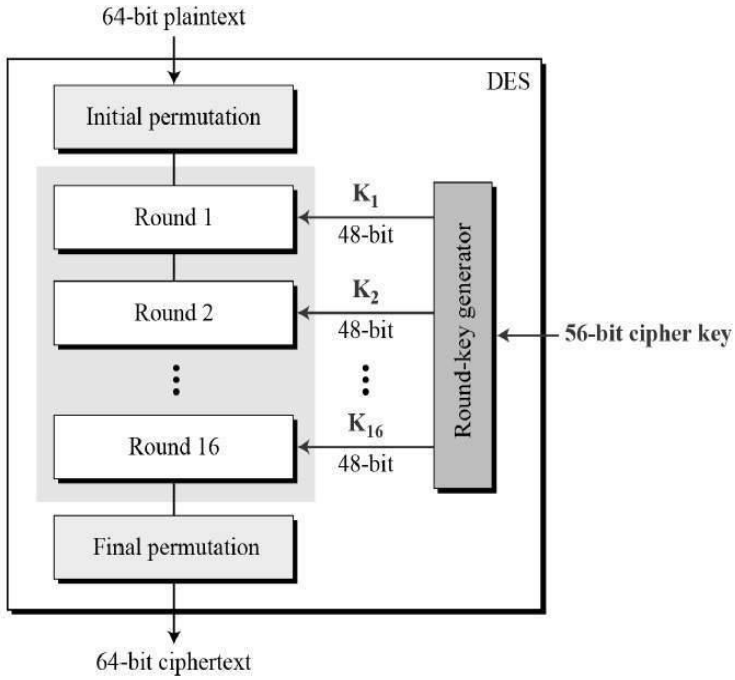
This is the **ciphertext** at the bottom of the SPN diagram.

How this maps EXACTLY to the diagram

Diagram Label	**What we did**
plaintext	01101101
add key K_1	XOR with 10100101
S_1 S_2 S_3 S_4	4-bit S-box substitution
permutation	Bit rearrangement
add key K_2, K_3	XOR in each round
final add key K_4	Last whitening key
ciphertext	01111101

Data Encryption Standards (DES)

The Data Encryption Standard (DES) is a symmetric-key block cipher and developed by **IBM**, standardized and published by the **National Institute of Standards and Technology (NIST) 1977.** https://www.nist.gov/



DES is an implementation of a Feistel Cipher. It uses **16 round** Feistel structure. The **block size is 64-bit(PT/CT)**. Though, key length **is 64-bit, DES** has an **effective key length of 56 bits key (64 bits including parity)**, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration −

Since DES is based on the Feistel Cipher, all that is required to specify DES is −

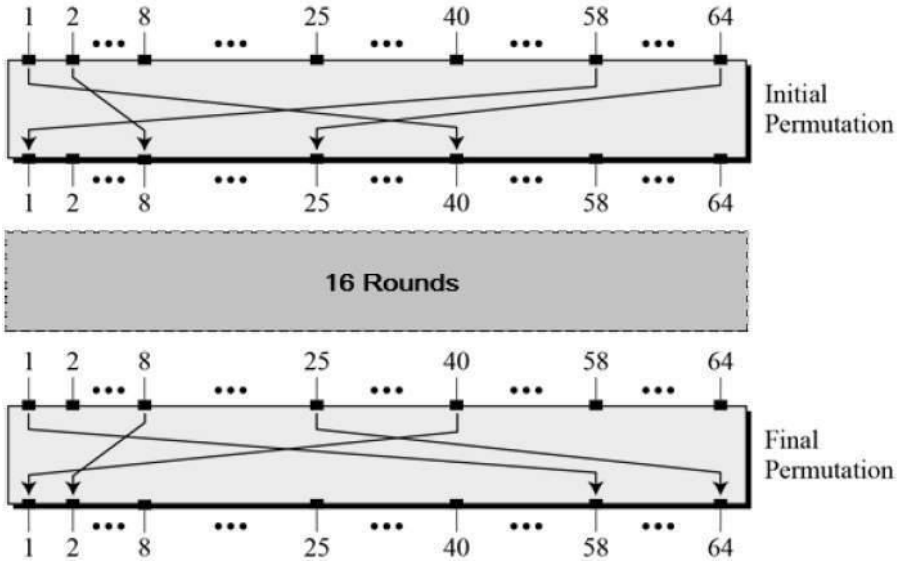
- **Round function**
- **Key schedule**
- **Any additional processing − Initial and final permutation**

DES Block & Key Sizes

Component	Size
Plaintext block	64 bits
Ciphertext block	64 bits
Input key	64 bits
Effective key	56 bits (8 parity bits removed)
Rounds	16

Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows −



- A **fixed bit-reordering** of the 64-bit plaintext
- No cryptographic strength by itself
- Used for hardware efficiency
- After IP:
  Output is split into two halves:
  L0 (32 bits) | R0 (32 bits)

Round Function

The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

Each round i (1 to 16) follows:

Li = Ri−1

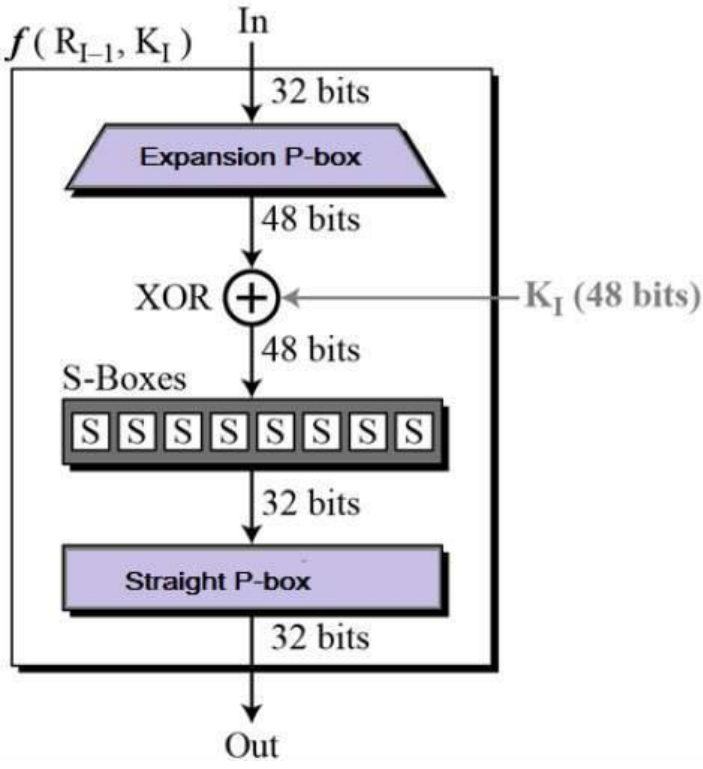
Ri = Li−1 ⊕ F(Ri−1, Ki)

Where:

Ki = 48-bit round key

F = DES round function

⊕ = XOR



DES Round Function F

The function **F(R, K)** consists of **four precise steps**:

Step 1: Expansion (E-Box)

- Expands **32-bit R** to **48 bits**
- Some bits are repeated
- Purpose: Match key size and provide diffusion

32 bits → 48 bits

Step 2: Key Mixing

- XOR expanded R with **48-bit round key Ki**

$E(R) \oplus Ki$

Step 3: Substitution (S-Boxes)

- 48 bits divided into **8 blocks of 6 bits**
- Each block enters **one S-box**
- Each S-box:
  - Input: 6 bits
  - Output: 4 bits

Total output after S-boxes:

8 × 4 bits = 32 bits

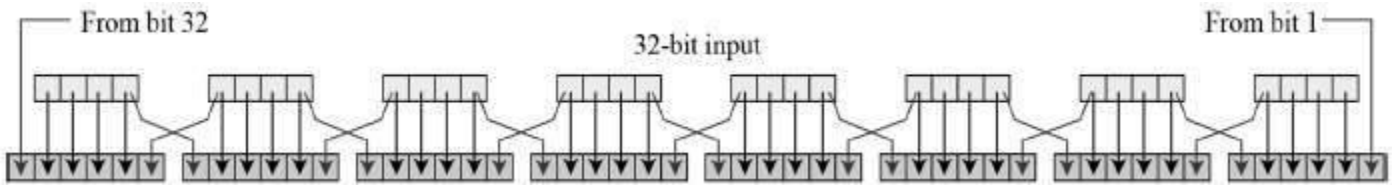
👉 **This is the ONLY non-linear part of DES**

Step 4: Permutation (P-Box)

- Rearranges the 32-bit S-box output
- Spreads bits for diffusion

Expansion Permutation Box / (E-Box)

Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration −



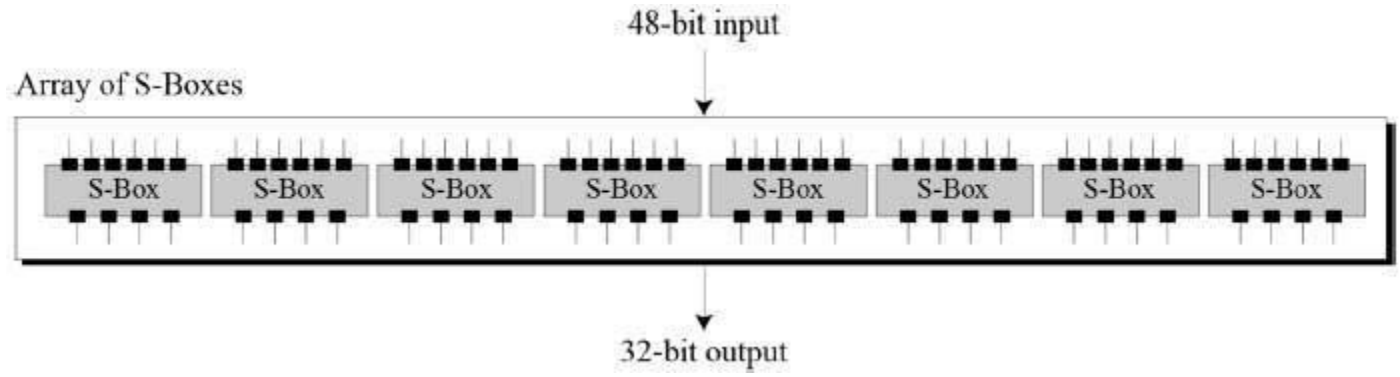
- Expands **32-bit R** to **48 bits**
- Some bits are repeated
- Purpose: Match key size and provide diffusion
- 32 bits → 48 bits

The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown −

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

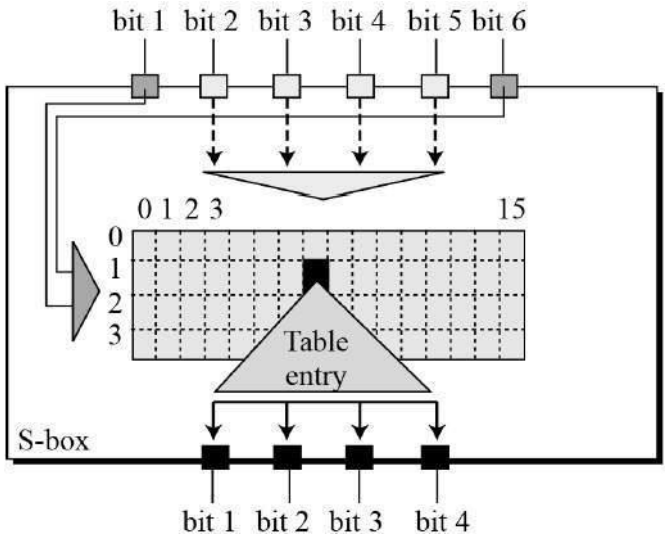
XOR (Whitener). − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

Substitution Boxes. − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −



- 48 bits divided into **8 blocks of 6 bit**
- Each block enters **one S-box**
- Each S-box: Input: 6 bits Output: 4 bits
- Total output after S-boxes: **8 × 4 bits = 32 bits**

- The S-box rule is illustrated below −



- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.

Straight Permutation –

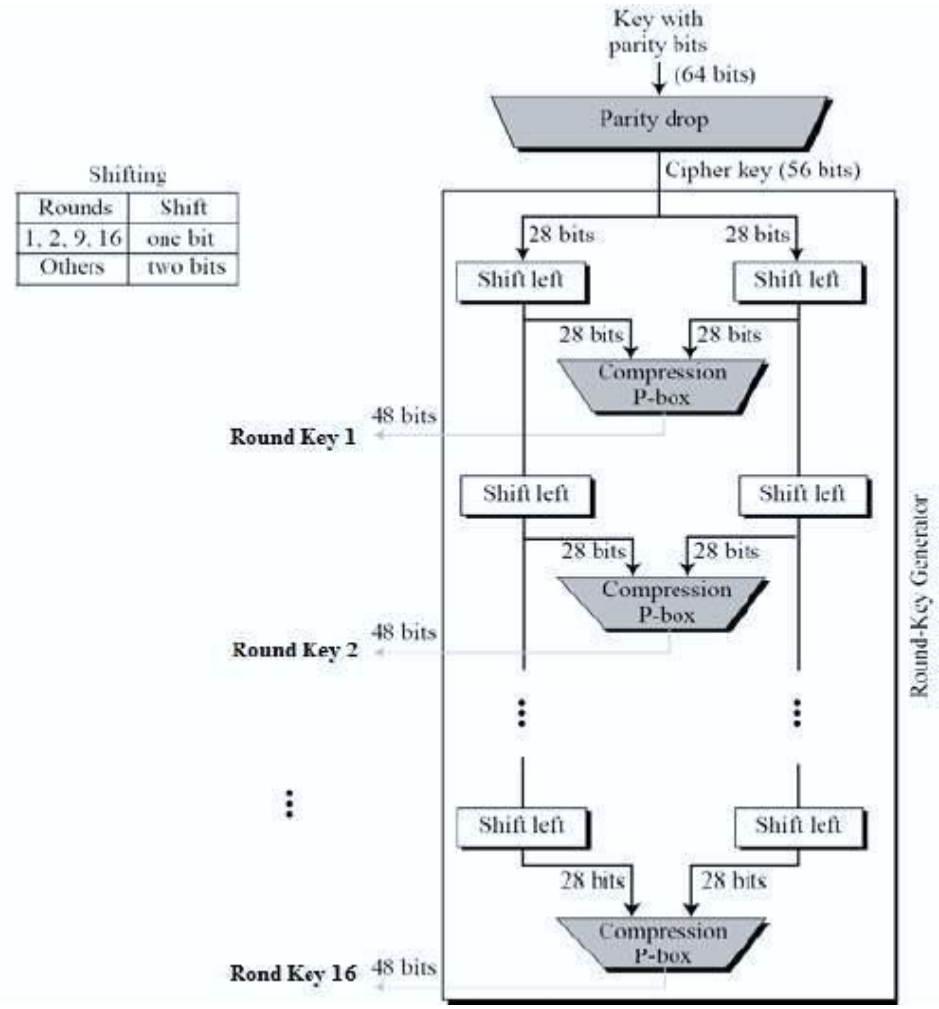
The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

- Rearranges the 32-bit S-box output
- Spreads bits for diffusion

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation/ Key Schedule (Round Key Generation)

- The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration –
- The logic for Parity drop, shifting, and Compression P-box is given in the DES description.



DES uses **16 different 48-bit round keys** derived from the main key.

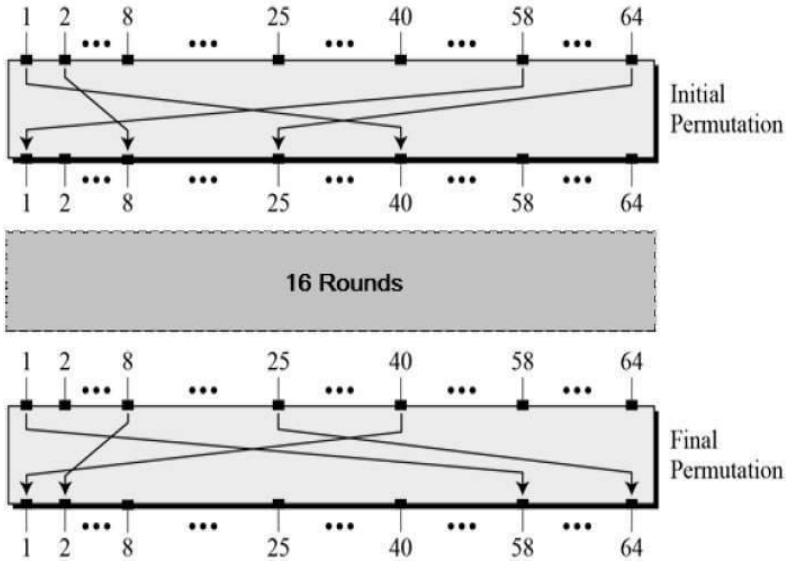
Steps:

1. Remove 8 parity bits → **56-bit key**
2. Split into two halves:
3. C0 (28 bits), D0 (28 bits)
4. Perform **left circular shifts** (1 or 2 bits depending on round)
5. Apply **PC-2 permutation** → 48-bit round key

This produces: K1, K2, K3, …, K16

Final Permutation (IP^{-1})

- Inverse of Initial Permutation
- Applied after round 16
- Produces final **64-bit ciphertext**



DES Decryption

- Same algorithm as encryption
- **Only difference:** round keys applied in **reverse order**

Encryption: K1 → K16

Decryption: K16 → K1

This is possible because DES is a **Feistel cipher**

DES Analysis

- The DES satisfies both the desired properties of block cipher.
- These two properties make cipher very strong.
  - **Avalanche effect** − A small change in plaintext results in the very great change in the ciphertext.
  - **Completeness** − Each bit of ciphertext depends on many bits of plaintext.
- During the last few years, **cryptanalysis have found some weaknesses in DES when key selected are weak keys.** These keys shall be avoided.
- DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.
- Why DES is Insecure Today???

Issue	Explanation
Small key size	56-bit key vulnerable to brute-force
Exhaustive search	Broken in hours using modern hardware
Legacy design	Replaced by AES

Variants of DES

Variant	Description
DES	Original, insecure
Double DES	Two encryptions (still weak)
Triple DES (3DES)	Encrypt-Decrypt-Encrypt, stronger

DES Summary

- ✓ DES is a symmetric block cipher that encrypts 64-bit plaintext blocks using a 56-bit key through 16 rounds of Feistel structure.
- ✓ Each round applies expansion, key mixing, substitution using S-boxes, and permutation.
- ✓ An initial and final permutation surround the Feistel rounds.
- ✓ Although DES was widely used, it is now considered insecure due to its small key size and has been replaced by AES.

DES in One-Line

DES = IP + 16 Feistel Rounds (Expansion → XOR → S-Box → P-Box) + IP^{-1}

WHY DOUBLE DES AND TRIPLE DES?

DES became insecure due to its **small key size (56 bits)**.
To improve security **without redesigning DES**, cryptographers introduced **Double DES** and **Triple DES**.

---

1 Why DES Was Not Enough

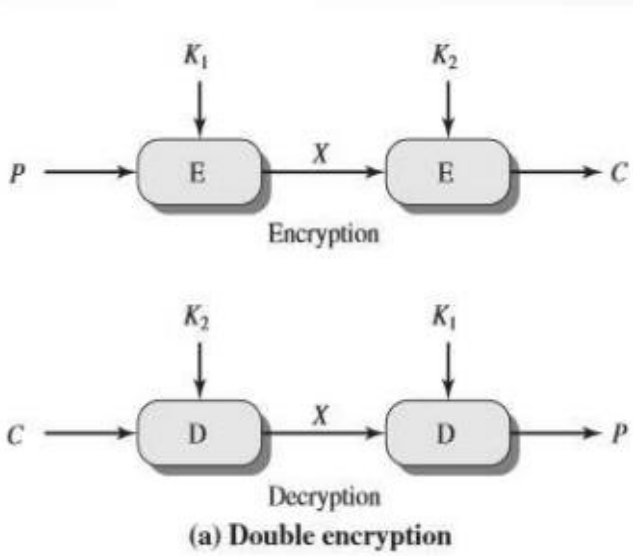
DES problems:

- Key size = **56 bits** Vulnerable to **brute-force attack**
- Modern computers can break DES in hours

So the question arose:

How can we increase security while still using DES hardware and software?

👉 Answer: **Apply DES multiple times**



(a) Double encryption

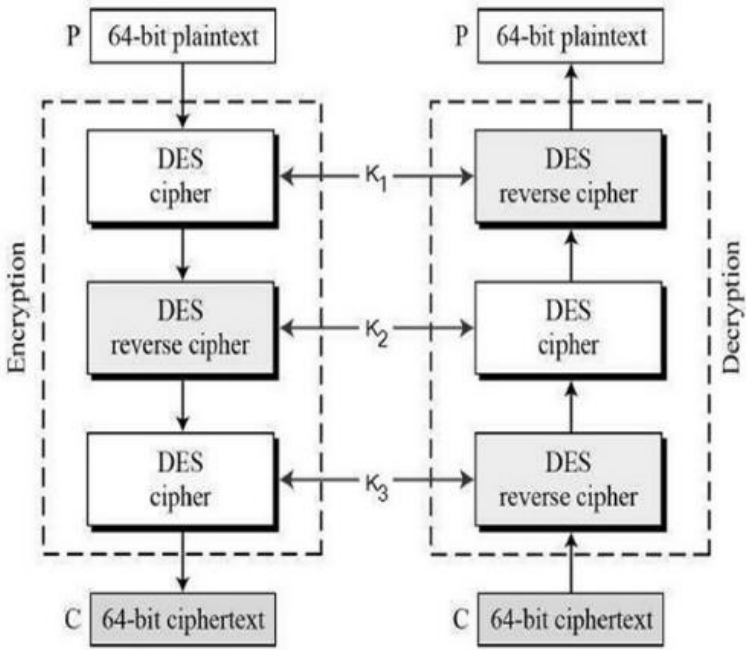
DOUBLE DES

Definition : Double DES encrypts plaintext **twice using two different DES keys**.

Encryption: C = E(K2, E(K1, P))

Decryption: P = D(K1, D(K2, C))

Where: P = plaintext, C = ciphertext, K1, K2 = two independent 56-bit keys



✓ Expected Advantage

- Key size increases from **56 bits → 112 bits**
- Much stronger than single DES (in theory)

❌ Why Double DES Failed

🔒 Meet-in-the-Middle Attack (MITM)

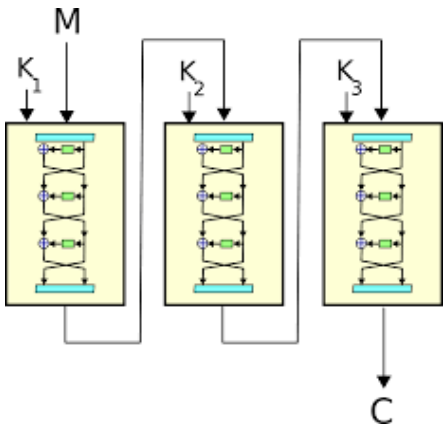
- Attacker encrypts plaintext with all possible K1
- Attacker decrypts ciphertext with all possible K2
- Matches occur in the middle

⛔ Effective security drops to **~57 bits**, almost same as DES.

❌ Conclusion on Double DES

Aspect	Status
Security improvement	❌ Not effective
Vulnerable to MITM(Man-in-the-Middle)	✓ Yes
Practical use	❌ No

🔐 TRIPLE DES (3DES)



✓ **Definition :** Triple DES applies DES **three times** with either **two or three keys**.

Most common version (EDE mode):

Encryption: C = E(K1, D(K2, E(K1, P)))

Decryption: P = D(K1, E(K2, D(K1, C)))

✓ Why EDE (Encrypt–Decrypt–Encrypt)?

- Backward compatibility with DES

- If K1 = K2, then:

3DES = DES

🔑 Types of Triple DES

Version	Keys Used	Effective Key Size
3DES (2-key)	K1, K2	112 bits
3DES (3-key)	K1, K2, K3	168 bits

✓ Advantages of Triple DES

- Stronger than DES and Double DES
- Resistant to Meet-in-the-Middle attacks
- Well-tested and standardized
- Used in banking and financial systems

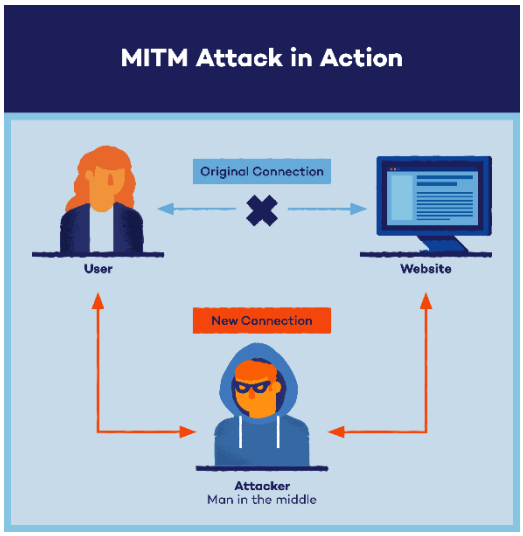
❌ Disadvantages of Triple DES

- **3× slower** than DES
- Slower than AES
- Being phased out in modern systems

Feature	DES	Double DES	Triple DES
No. of DES operations	1	2	3
Keys used	1	2	2 or 3
Effective key size	56 bits	~57 bits	112 / 168 bits
Secure today	❌ No	❌ No	⚠ Limited
MITM resistant	❌	❌	✓

✍ Synopsys/ Summary

- ✓ Double DES and Triple DES were proposed to overcome the small key size limitation of DES.
- ✓ Double DES applies DES encryption twice using two different keys, but it is vulnerable to meet-in-the-middle attacks and does not significantly increase security.
- ✓ Triple DES applies DES three times using either two or three keys and provides much stronger security. It is resistant to meet-in-the-middle attacks and was widely used in banking systems.
- ✓ However, Triple DES is slower than DES and has largely been replaced by AES in modern applications.



Home Work: What is Avlance Effect, diffusion, confusion?

MCQs

Feistel Cipher, SPN & DES

1. In a Feistel cipher, which operation is performed to compute the new right half?

A. Ri = Ri−1 ⊕ Ki
B. Ri = Li−1 ⊕ F(Ri−1, Ki)
C. Ri = F(Li−1, Ki)
D. Ri = Li−1 ⊕ Ri−1

2. Which property allows Feistel ciphers to be decrypted using the same structure as encryption?

A. Invertible S-boxes
B. Key whitening
C. XOR reversibility
D. Block permutation

3. In a Feistel cipher, the round function F must be:

A. Invertible
B. Linear
C. Non-linear only
D. Not necessarily invertible

4. Which cryptographic structure is used by DES?

A. SPN
B. Hybrid network
C. Feistel network
D. Substitution-only network

5. In a Substitution–Permutation Network (SPN), which component provides non-linearity?

A. P-Box
B. XOR
C. S-Box
D. Key schedule

6. Why does SPN require invertible S-boxes?

A. To increase key size
B. To allow decryption
C. To reduce rounds
D. To avoid brute force attacks

7. Which of the following is a major difference between Feistel and SPN?

A. Feistel uses XOR; SPN does not
B. Feistel requires invertible F; SPN does not
C. Feistel splits block into halves; SPN does not
D. SPN uses keys; Feistel does not

8. How many rounds does the DES algorithm use?

A. 8
B. 12
C. 16
D. 32

9. In DES, which operation expands the right half from 32 bits to 48 bits?

A. Initial permutation
B. S-Box
C. Expansion permutation (E-Box)
D. P-Box

10. Which part of DES provides the strongest security against linear cryptanalysis?

A. Initial permutation
B. Final permutation
C. XOR operation
D. S-Boxes

11. Why is DES considered insecure today?

A. Uses SPN structure
B. Uses XOR operations
C. Has only 56-bit effective key
D. Has too many rounds

12. Which cipher is based on SPN architecture instead of Feistel?

A. DES
B. Triple DES

C. AES
D. Blowfish

✅ ANSWER KEY WITH REASONS & INTERPRETATIONS

1. Correct Answer: B

Reason:
In Feistel structure, the new right half is generated by XORing the previous left half with the output of the round function.

Interpretation:
This ensures diffusion and enables reversibility.

2. Correct Answer: C

Reason:
XOR is reversible:

$A \oplus B \oplus B = A$

Interpretation:
This allows the same algorithm to be used for encryption and decryption.

3. Correct Answer: D

Reason:
Feistel networks do NOT require F to be invertible.

Interpretation:
This gives Feistel designs flexibility and simplicity.

4. Correct Answer: C

Reason:
DES is a classic Feistel cipher with 16 rounds.

Interpretation:
DES applies Feistel structure in practice.

5. Correct Answer: C

Reason:
S-Boxes introduce non-linear substitutions.

Interpretation:
They provide **confusion**, essential for security.

---

6. Correct Answer: B

Reason:
SPN uses inverse S-boxes during decryption.

Interpretation:
Without invertibility, decryption would fail.

---

7. Correct Answer: C

Reason:
Feistel splits plaintext into left and right halves; SPN processes the whole block.

Interpretation:
This is the core architectural difference.

---

8. Correct Answer: C

Reason:
DES uses exactly **16 Feistel rounds**.

Interpretation:
Security increases with rounds, but DES key length is insufficient.

---

9. Correct Answer: C

Reason:
The E-Box expands 32 bits to 48 bits before key mixing.

Interpretation:
This allows DES to use 48-bit round keys.

---

10. Correct Answer: D

Reason:
S-Boxes are non-linear and resist cryptanalysis.

Interpretation:
They are the strongest security component of DES.

11. Correct Answer: C

Reason:
56-bit keys are vulnerable to brute-force attacks.

Interpretation:
DES was replaced by AES for this reason.

12. Correct Answer: C

Reason:
AES is based on SPN, not Feistel.

Interpretation:
AES uses substitution, permutation, and finite field arithmetic.

Advanced Encryption Standards (AES) Cipher

- Advanced Encryption Standard (AES) is a highly trusted **encryption algorithm** used to secure data by converting it into an unreadable format without the proper key.
- It is developed by the **National Institute of Standards and Technology (NIST) in 2001**.
- It is is widely used today as it is much stronger than DES and triple DES despite being harder to implement.
- **AES encryption** uses various **key lengths (128, 192, or 256 bits)** to provide strong protection against unauthorized access.
- This **data security** measure is efficient and widely implemented in securing **internet communication**, protecting **sensitive data**, and encrypting files.
- AES, a cornerstone of modern cryptography, is recognized globally for its ability to keep information safe from cyber threats.
    - **AES is a Block Cipher.**
    - **The key size can be 128/192/256 bits.**
    - **Encrypts data in blocks of 128 bits each.**

- That means it takes **128 bits as input and outputs 128 bits of encrypted cipher text.**
- AES relies on the substitution-permutation network principle, which is performed using a series of linked operations that involve replacing and shuffling the input data.

Working of The Cipher

AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

N (Number of Rounds)	Key Size (in bits)
10	128
12	192
14	256

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.

A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

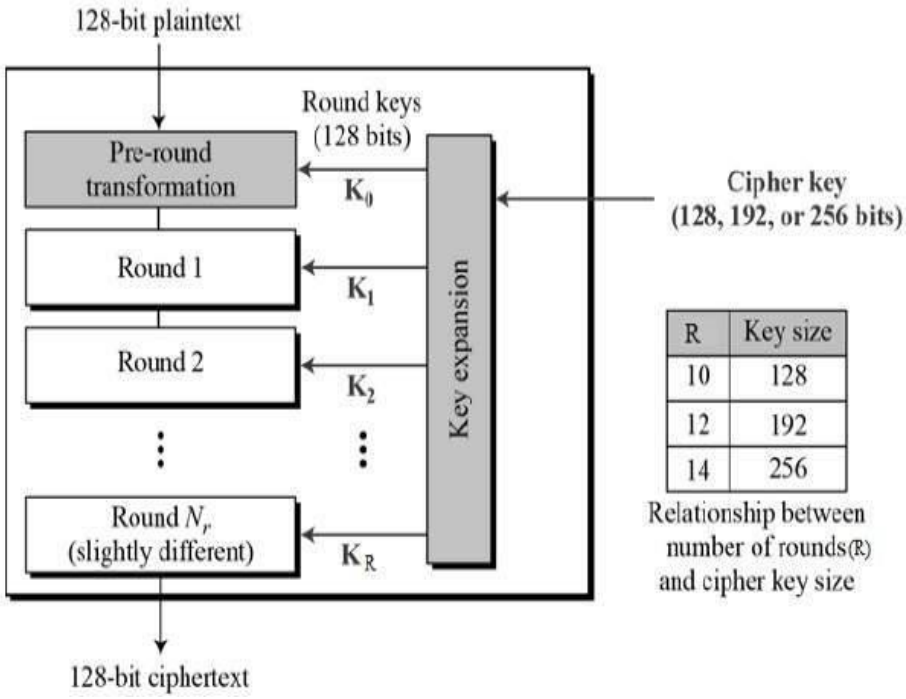
The features of AES are as follows −

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

Operation of AES

- AES is an iterative rather than Feistel cipher.
- It is based on substitution permutation network.
- It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).
- Interestingly, AES performs all its computations on bytes rather than bits.

- Hence, AES treats the 128 bits of a plaintext block as 16 bytes.
- These 16 bytes are arranged in four columns and four rows for processing as a matrix −
- Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys.
- Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.



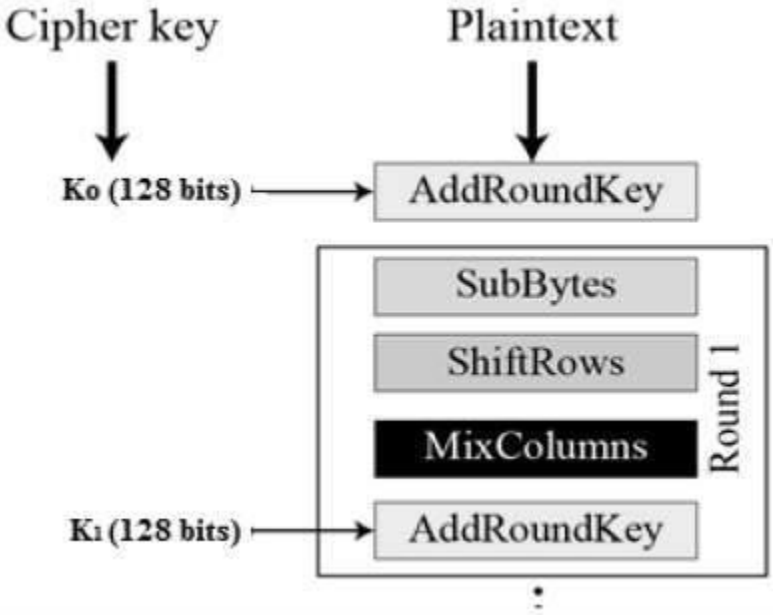
R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

The schematic of AES structure is given in the following illustration −

Encryption Process

- Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below −



Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that fall off are re-inserted on the right side of row. Shift is carried out as follows −

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

AES Analysis

- In present day cryptography, AES is widely adopted and supported in both hardware and software.
- Till date, no practical cryptanalytic attacks against AES has been discovered.
- Additionally, AES has built-in flexibility of key length, which allows a degree of future-proofing against progress in the ability to perform exhaustive key searches.

- However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.

NIST: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf

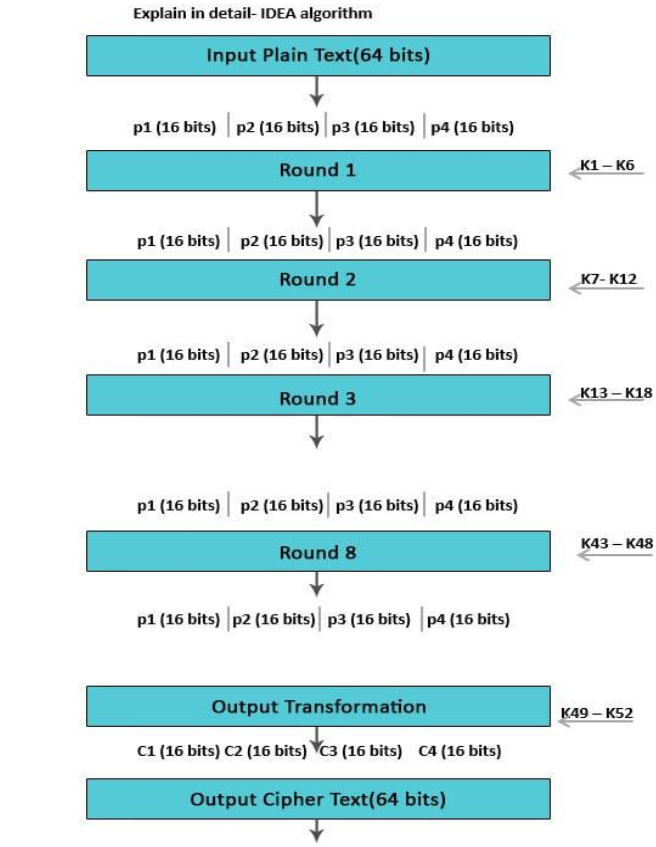
https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf

International Data Encryption Standard (IDEA)

- IDEA stands for International Data Encryption Algorithm.
- IDEA is a block cipher developed by James Massey and Xuejia Lai and initially specified in 1991.
- It has a 128-bit key length and works with 64-bit blocks.
- It consists of a set of eight identical transformations based on bitwise exclusiveor, addition, and multiplication modules.
- It is based on a symmetric cipher and uses a relatively weak key design technique, hence the algorithm's security is considerably less than that of DES.
- *IDEA failed to gain much popularity because of its complicated development*.
- It is an IDEA that, unlike other block cipher algorithms, is being studied by the Swiss corporation Ascom.
- However, they are unique in that they grant permission for free non-commercial usage of their method, which has resulted in IDEA becoming known as the block cipher algorithm used within the famous encryption for the message character.

How IDEA Works?

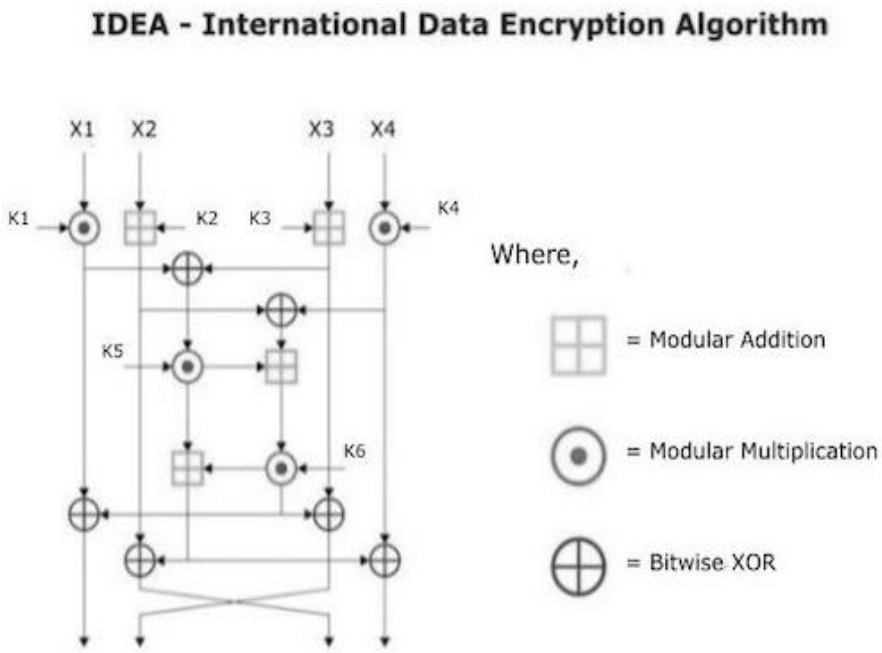
- IDEA is a block cipher that operates on 64-bit plaintext and a 128-bit key. IDEA, like DES, is reversible, which means that the comparable technique can be used for both encryption and decryption. IDEA requires both **diffusion and confusion** for encryption.
- The 64-bit plaintext is broken into four 16-bit sections (P1-P4). These are inputs for the first round. There are eight such rounds. The key contains 128 bits. In each cycle, six sub-keys are generated from the original key, each of which contains 16 bits.
- The first round can use keys K1 through K6, the second round can use keys K7 through K12,



and the final round can use keys K13 through K18. The last step requires an output modification that requires four subkeys (K49 to K52).

- The final output is the result of the output transformation stage. The blocks C1-C4 are joined to generate the final result.

Rounds in IDEA



- There are eight rounds in the IDEA.
- Each round consists of a series of operations on the four data blocks with six keys.
- The first round can include keys K1 to K6, the second round can have keys K7 to K12, and the last round can have keys K13 to K18. The final stage involves an output modification that requires four subkeys (K49 to K52).
- The final output is the result of the output transformation stage.
- The final output is formed by linking the blocks C1-C4. Each round has 14 steps, as follows −

Details of one round in IDEA

Step1 : Multiply * P1 and K1

Step2 : Add * P2and K2

Step3 : Add * P3and K3

Step4 : Multiply * P4 and K4

Step5 : XOR the results of step1 and step3

Step6 : XOR the results of step2 and step4

Step7 : Multiply * the results of step5 with K5

Step8 : Add * the results of step6 and step7

Step9 : Multiply * the results of step8 with K6

Step10 : Add * the results of step7 and step9

Step11 : XOR the results of step1 and step9

Step12 : XOR the results of step3 and step9

Step13 : XOR the results of step2 and step10

Step14 : XOR the results of step4 and step10

- The Add * and Multiply * in the next step in each cycle are not elementary addition and multiplication, but rather addition module 216, i.e., 65536, and multiplication module 216 + 1, i.e., 65537.
- The common addition gives a number of 17 bits. It can only use 16 bit places for the round2 output.
- As a result, it can reduce this number (which is 130753 in decimal) to a 16-bit value. It can take modulo 65536 of this. 130753 modulo 65536 gives 65217, which is 11111111011000001 in binary and is a 16-bit integer that fits well within the system.
- The input blocks are P1-P4, the subkeys are K1-K6, and the output of this step is R1-R4 (rather than C1-C4 because this is not the final cipher text). It is an intermediate output that will be handled in the following steps, as well as during the output transformation stage.

Operations used in IDEA

IDEA's operations include −

- IDEA uses bit-by-bit exclusive-OR, denoted as v
- Add integers modulo 216 (modulo 65536), using unsigned 16-bit integers for input and output.
- This operation is referred to as ⊕.
- Integer multiplication modulo 216+1 (modulo 65537), where inputs and outputs are unsigned 16-bit integers. A block of all zeros represents 216. This operation is referred to as ⊗.

Sub-key Generation for a Round

- In the first round, bit positions 1-96 of the key are used. Bits 97-128 remain not used. They are assigned to round two.
- In the second cycle, bits 97-128 are used first, resulting in a 25-bit circular left shift, followed by new bits from 26-89. Bits 90-128 and 1-25 remain unused.
- In the third round, unused bits 90-128 and 1-25 are initially used again, followed by a circular leftshift of 25 bits and the use of bits 51-82. Bit positions 83-128 and 1-50 remain unused.
- In the fourth round, bits 83-128 and 1-50 are used.
- In the fifth round, a circular left-shift of 25 bits arises, with bit positions 76-128 and 1-43 being used. Bits 44-75 remain unused.

- In the sixth round, the unused bit from the fifth round is used, and a left circular shift of 25 bits occurs, leaving bit positions 37-100 empty.
- In the seventh round, the unused bits from the sixth round, 37-100, are used first, followed by a 25-bit circular left shift to bits 126-128 and 1-29. Bits 30-125 remain unused.
- The unused bit location 30-125 from round seven is being used, and the key is deactivated.

Output Transformation

This is a one-time operation. It takes place at the end of the eighth round. As a result, a 64-bit value is separated into four sub-blocks (say R1 through R4), each with its own subkey.

Decryption

Decryption is similar to encryption, but it reverses the order of the round keys and the subkeys for odd rounds. Thus, the inverse of K49-K52 restores the values of subkeys K1-K4 for the respective set operation in cryptography, while K5, and K6 of each group should be restored by K47 and K48 for decryption in this operation in cryptography.

Characteristics of IDEA

The following characteristics of IDEA related to its cryptographic strength −

- **Block length** − The block length should prevent statistical analysis. However, the complexity of developing an effective encryption algorithm seems to increase exponentially with block size.
- **Key strength** − The key length should be sufficient to avoid exhaustive key searches. With a length of 128 bits, IDEA appears to be secure in this area well into the future.
- **Confusion** − The ciphertext should rely on the plaintext and key in a complex and effective manner. The goal is to make it more difficult to determine how the ciphertext statistics relate to the plaintext statistics. IDEA achieves this purpose by three distinct actions, as discussed subsequently. This contrasts with DES, which is based primarily on the XOR operation and small nonlinear S-boxes.
- **Diffusion** − Each plaintext bit should impact every ciphertext bit, just as each key bit should influence every ciphertext bit. The dispersion of a single plaintext bit over multiple ciphertext bits obscures the plaintext's statistical structure. IDEA is highly effective in this aspect.

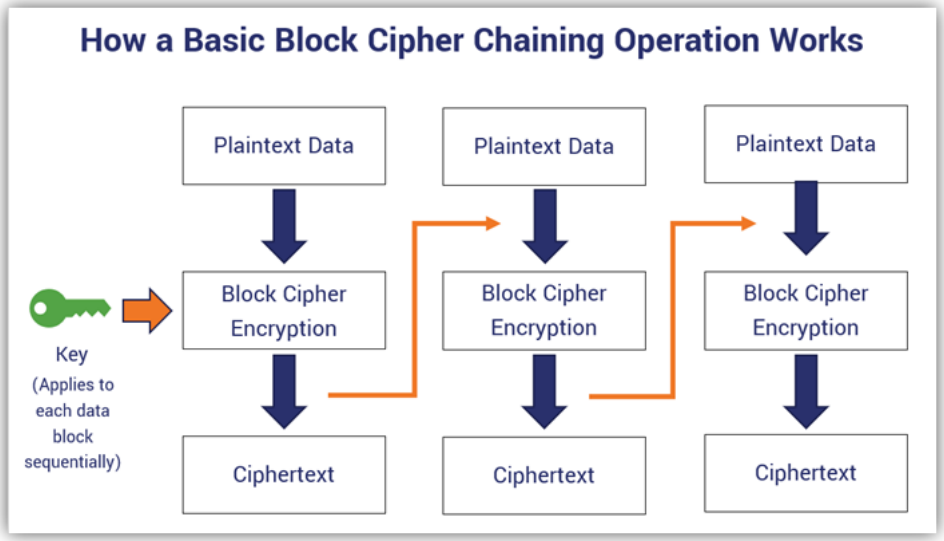
Security of IDEA

- The IDEA encryption method was thought to be very strong against certain types of attacks, like differential cryptanalysis. Until 2007, no one had successfully found weaknesses in its design. Even the best known attack by that time could only break a simplified version of IDEA with 6 rounds, while the full version uses 8.5 rounds.
- Bruce Schneier, a respected cryptographer, praised IDEA in 1996, saying it was the best and most secure encryption method available. However, by 1999, he stopped recommending it because newer, faster methods were developed, some weaknesses were found in IDEA, and there were patent issues.

- In 2011, a method called "meet-in-the-middle" was used to break the full 8.5-round IDEA encryption. Then, in 2012, another attack called "narrow-bicliques" was used to weaken IDEA slightly, but it still remains secure for practical use.

Modes of Block Cipher Encryptions

- We will talk about a block cipher's different modes of operation in this chapter. These are the steps involved in a general block cipher's procedure. It is important to note that the various modes produce various qualities which improve the block cipher's overall security.



What is Block Cipher Modes of Operation?

- An algorithm that uses a block cipher to offer information security, such as confidentiality or authenticity, is known as a block cipher mode of operation in cryptography. A real block cipher can be used to convert secure cryptographic bits (for encryption or decryption) or to create a single block of fixed length. A mode of operation describes how to safely convert amounts of data bigger than a block by constantly utilising a cipher's single-block operation.
- For every encryption operation, most of modes need a distinct binary sequence, commonly referred to as an initialization vector (IV). In addition to being non-repeating, the IV needs to be random in certain modes. The initialization vector is used to guarantee that different ciphertexts are generated even in the event that the same plaintext is encrypted with the same key several times on different occasions. The block size is always fixed all over transformation, even though block ciphers can operate with all kinds of block sizes. If the last data fragment is smaller than the current block size, it must be padded to a full block in order for block cipher modes to function on entire blocks.
- However, certain modes do not require padding because they essentially use a block cipher in place of a stream cipher. In the past, a great deal of research has been done on encryption modes' error propagation characteristics under different conditions of data change. Integrity protection was after that considered to be a completely different

cryptographic objective. Authenticated encryption modes are some modern methods of operation that effectively combine authenticity and secrecy.

Modes of Operation

- Block cipher processes are classified into five categories:
  - **CFB (Cipher Feedback),**
  - **OFB (Output Feedback),**
  - **CTR (Counter),**
  - **CBC (Cipher Block Chaining), and**
  - **ECB (Electronic Code Block).**
- Block ciphers work in ECB and CBC mode, while block ciphers functioning as stream ciphers work in CFB and OFB mode.
- A single value can be transmitted insecurely using ECB, a block of text can be encrypted using CBC, a stream of data can be encrypted using CFB, an encrypted stream of data can be transmitted using OFB, and block-oriented applications can be transmitted using CTR.

- **ECB Mode** − ECB is the simplest block cipher to operate. As each block of input plaintext is directly encrypted and the output is in the form of blocks of encrypted ciphertext.

- **CBC Mode** − Since ECB compromises certain security requirements, such as a direct connection between the cipher text and the plain text that makes it easier for attackers to decrypt the encoded information, CBC is the advanced mode of ECB.

- **CFB Mode** − In this case, the cipher is sent as feedback for the subsequent encryption block.

- **OFB Mode** − With the exception of sending the encrypted output as feedback rather than the actual cipher, which is the XOR output, the OFB operates in a manner very similar to that of the CFB.

- **CTR Mode** − An implementation of a basic counter-based block cipher is the CTR.

History

- "Modes of operation" are a type of encrypted message use procedures that were developed around 1981. They were referred to as CFB, CBC, OFB, and ECB. A second method known as AES was implemented later, in 2001. Then, XTS-AES, a further one, was added in 2010. While there are other methods as well, these are the primary ones that have been accepted by NIST.

- These techniques are helpful in securing private information, but they do not always guarantee that they are not accidentally or intentionally altered. People use a "digital signature" or "message authentication code" to verify if anything has changed. HMAC, CMAC, and GMAC are a few codes for this.

- People realised that it was difficult to create a system for securing information and monitoring changes. Thus, new methods were developed that combine the two tasks. We refer to these as "authenc" or "authenticated encryption." A few examples of this are OCB, IAPM, CCM, GCM, CWC, and EAX.

- These means that security information are governed by different sets of rules. NIST, ISO, IEC, IEEE, ANSI, and IETF are a few major organisations.

Initialization Vector (IV)

- A set of bits called an initialization vector (IV) which is used to mix up encryption. It helps to make sure the outcome is different each time, even if you encrypt the same message multiple times. Unlike a key, the IV does not need to be kept under wraps.

- It is important to use a different IV every time you encrypt something using the same key for a wide range of encryption techniques. We refer to this as a "cryptographic nonce." Certain encryption techniques need the IV to be random.

- Reusing the same IV can lead to problems. Using the same IV, for example, can reveal information about the first portion of the message in some encryption techniques. On the other hand, it may weaken the encryption, making it more easily decrypted.

- Using a portion of the encrypted message as the IV for the subsequent one is one technique to ensure that the IV is unique each time. However, this approach is not safe because someone can decipher the message by guessing the IV.

- "Synthetic initialization vectors" (SIVs) are specialised IVs that can be created in certain methods. They ensure that the IV is properly mixed by using a special mathematical procedure. This keeps the encryption secure even in the event that the randomness is not perfect or is attempted to be controlled.

Other modes and Cryptographic Primitives

- Block ciphers can be used in a variety of ways to protect data. While some of these methods are useful and secure to use, others are completely unsafe. Certain ones are employed especially for encrypting objects, like hard drives.

- An initialization vector (IV) is something that is used in someof these methods. This functions similarly to a unique number that mixes up the encryption process. The IV can require to be kept confidential or unique each time it is used.

- When the same IV and key are used in some methods, it can lead to serious problems and make decryption very easy. Therefore, it is important to use IVs correctly.

- Creating unexpected random numbers and confirming that a message has not been altered are two further uses for block ciphers.

- Block ciphers can be usedto create message authentication codes (MACs), which are used to confirm that a message has not been altered.

- The following are some more modes and cryptographic primitives –

    - Other Modes of Operation

- **Key Feedback Mode**
- **Davies-Meyer Hashing**
- **LRW (Tweakable narrow-block encryption mode)**
- **XEX (Tweakable narrow-block encryption mode)**
- **XTS (Tweakable narrow-block encryption mode)**
- **CMC (Wide-block encryption mode)**
- **EME (Wide-block encryption mode)**

Cryptographic Primitives

- **Deterministic Authenticated Encryption (NIST Key Wrap algorithm)**
- **SIV (RFC 5297) AEAD mode**
- **AES-GCM-SIV**
- **One-way compression function (used to build cryptographic hash functions)**
- **Cryptographically Secure Pseudorandom Number Generators (CSPRNGs)**
- **Message Authentication Codes (MACs) such as CBC-MAC, OMAC, and PMAC**

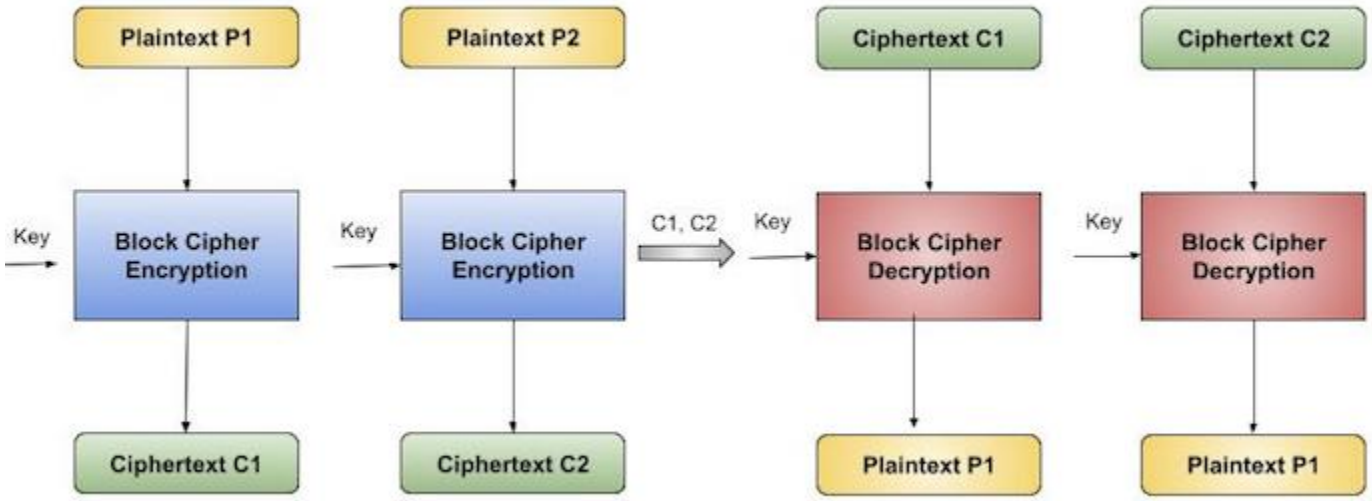
Electronic Code Book (ECB) Mode

- The Electronic Code Book (ECB) is a basic block cipher mode of operation that is mostly used together with symmetric key encryption. It is a simple method for handling a list of message blocks that are listed in sequence.
- There are several blocks in the input plaintext. The encryption key is used to independently and individually encrypt each block (ciphertext). Therefore, it is also possible to decrypt each encrypted block individually. Every kind of block can have a different encryption key supported by ECB.
- Every plaintext block in ECB has a predefined ciphertext value that matches it, and vice versa. Hence, similar ciphertexts can always be encrypted from identical plaintexts using identical keys. This implies that the output ciphertext blocks will always be the same if plaintext blocks P1, P2, and so forth are encrypted several times using the same key.
- In other words, the ciphertext value will always be equal to the plaintext value. This additionally holds true for plaintexts that include somewhat parts that are identical. For example, largely identical ciphertext parts will be included in plaintexts that have identical letter headers and are encrypted using the same key.

Operation

- To create the first block of ciphertext, the user takes the first block of plaintext and encrypts it using the key.
- He then uses the same procedure and key to go through the second block of plaintext, and so on.

- Because the ECB mode is deterministic, the ciphertext blocks that are produced will be identical if plaintext blocks P1, P2,..., and Pm are encrypted twice with the same key.
- In reality, we can technically generate a codebook of ciphertexts for every possible block of plaintext for a given key. Then, all that would be required for encryption would be to search for the necessary plaintext and choose the appropriate ciphertext. Because of this, the process is similar to assigning code words in a codebook, which is why it has an official name:
- Electronic Codebook mode of operation (ECB). Here's a visual representation of it −



Analysis of ECB Mode

- In real life, application data typically contain guessable partial information. For example, one can figure out the salary range. If the plaintext message is contained in a predictable area, an attacker may be able to decipher the ciphertext from ECB by trial and error.
- For example, an attacker can retrieve a salary figure after a limited number of attempts if the salary figure is encrypted using a ciphertext from the ECB mode. Since most applications do not wish to use deterministic ciphers, the ECB mode should not be used in them.

Cipher Block Chaining (CBC) Mode

- The Electronic Code Book (ECB) is a basic block cipher mode of operation that is mostly used together with symmetric key encryption. It is a simple method for handling a list of message blocks that are listed in sequence.

- There are several blocks in the input plaintext. The encryption key is used to independently and individually encrypt each block (ciphertext). Therefore, it is also possible to decrypt each encrypted block individually. Every kind of block can have a different encryption key supported by ECB.

- Every plaintext block in ECB has a predefined ciphertext value that matches it, and vice versa. Hence, similar ciphertexts can always be encrypted from identical plaintexts using identical keys. This implies that the output ciphertext blocks will always be the same if plaintext blocks P1, P2, and so forth are encrypted several times using the same key.

- In other words, the ciphertext value will always be equal to the plaintext value. This additionally holds true for plaintexts that include somewhat parts that are identical. For example, largely identical ciphertext parts will be included in plaintexts that have identical letter headers and are encrypted using the same key.

Operation

- To create the first block of ciphertext, the user takes the first block of plaintext and encrypts it using the key.

- He then uses the same procedure and key to go through the second block of plaintext, and so on.

- Because the ECB mode is deterministic, the ciphertext blocks that are produced will be identical if plaintext blocks P1, P2,..., and Pm are encrypted twice with the same key.

- In reality, we can technically generate a codebook of ciphertexts for every possible block of plaintext for a given key. Then, all that would be required for encryption would be to search for the necessary plaintext and choose the appropriate ciphertext. Because of this, the process is similar to assigning code words in a codebook, which is why it has an official name: Electronic Codebook mode of operation (ECB). Here's a visual representation of it −

Analysis of ECB Mode

- In real life, application data typically contain guessable partial information. For example, one can figure out the salary range. If the plaintext message is contained in a predictable area, an attacker may be able to decipher the ciphertext from ECB by trial and error.

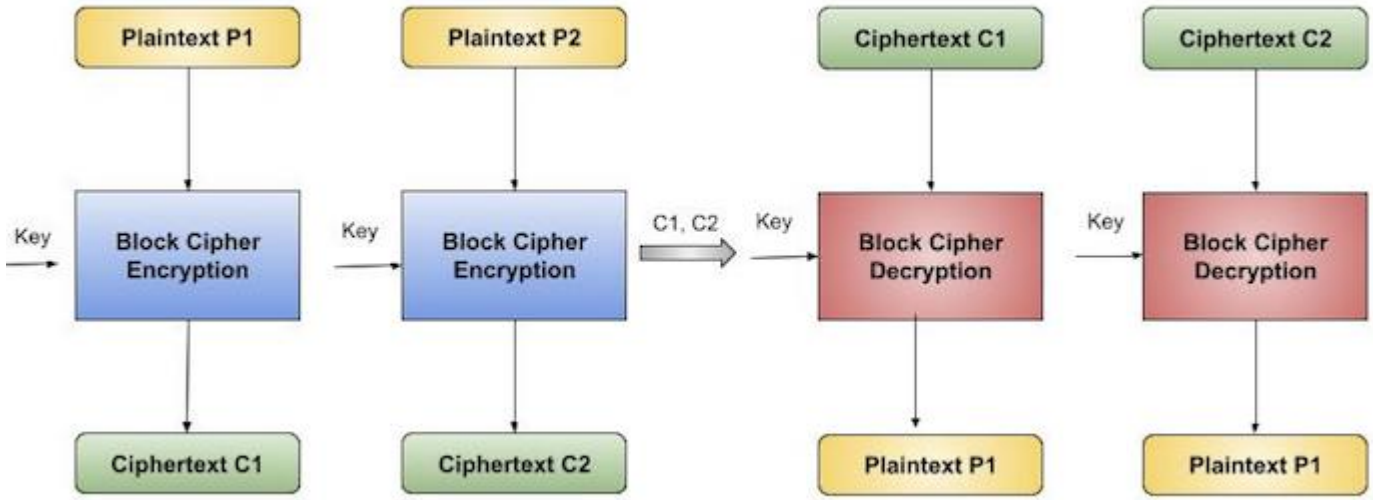
- For example, an attacker can retrieve a salary figure after a limited number of attempts if the salary figure is encrypted using a ciphertext from the ECB mode. Since most applications do not wish to use deterministic ciphers, the ECB mode should not be used in them.

- IBM created the Data Encryption Standard (DES) in the early 1970s, and in 1977 it was recognised as a Federal Information Processing Standard (FIPS). DES can encrypt data in five different ways. Among these is the original DES mode, or ECB.

- FIPS Release 81 now includes three new options: Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Output Feedback (OFB). Later, NIST Special Publication 800-38a was updated to include a fifth mode called Counter Mode. The design concepts of these modes vary, including whether using initialization vectors, whether to use blocks rather than streams, and whether or not encryption faults are likely to spread to subsequent blocks.

Implementation

- This implementation performs ECB encryption and decryption using simple byte-level operations. It is important to keep in mind that, in most cases, ECB mode is not secure and needs to be substituted by more secure modes like CBC or GCM.



Cipher Block Chaining (CBC) Mode

- It is a A block cipher that encrypts a series of bits as a whole or as a block and applies a ciphertext or encrypted key to the full block of cryptography is known as a cipher block chaining (CBC) mode. Initialization vectors (IVs) of particular characters in length are used in cipher block chaining. One of its main features is that it makes use of chaining, a working technique that makes a ciphertext block's decryption dependent on every block that came before it. Therefore, the cryptography of the block that comes right before the ciphertext has all of the validity of all blocks that came before it.

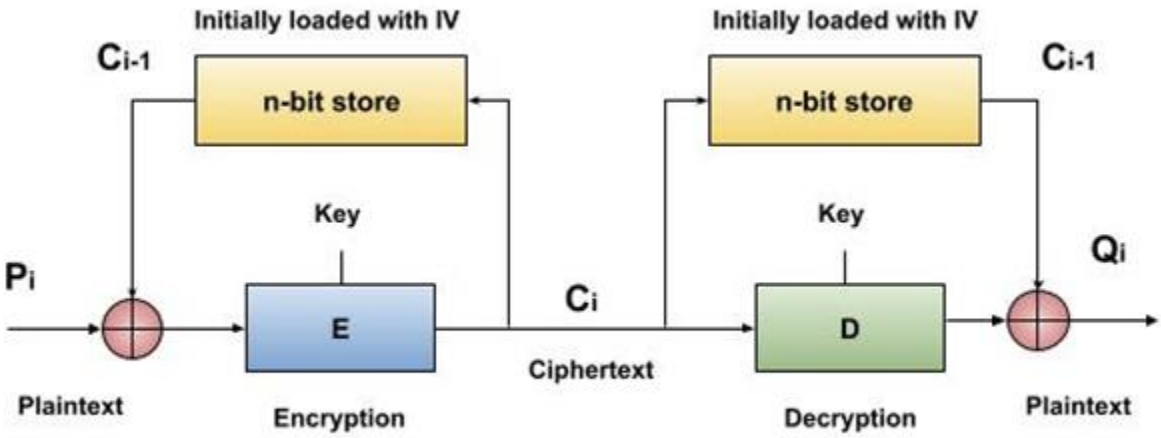
- A single-bit cryptography error in one block of ciphertext affects all subsequent block decryptions. The ciphertext blocks' order can be changed, which corrupts the decryption process. In cipher block chaining, each plaintext block is essentially encrypted using cryptography after being XORed (see XOR) with the block of ciphertext that comes right before it.

- Only when the same plaintext or original text block is encrypted with the same encryption key and the initialization vector (IV) and the ciphertext block order is left unaltered will the identical ciphertext blocks provide the desired outcome. Because the XORing process masks plaintext patterns, it has an advantage over the Electronic Code Book mode.

- When two communications are encrypted using the same encryption key, the initialization vector (IV) should differ. While some applications may find it useful and unique, the initialization vector does not necessarily need to be stored or kept secret.

- Since ECB compromises some security or privacy criteria, cipher block chaining, or CBC, is an improved or more sophisticated version of ECB. After XORing with an initial plaintext block of the cryptography, the preceding cipher block in the CBC is sent as input to the subsequent encryption process. To put it simply, an XOR output of the previous cipher block and the current plaintext or original text block is encrypted to create a cipher block.

- **The process is shown below −**



Operation

The above image shows how the CBC mode works. The following are the steps −

- The top register should be loaded with the n-bit Initialization Vector (IV).
- XOR the data value in the top register with the n-bit plaintext block.
- Use key K to encrypt the output of the XOR operation using the underlying block cipher.
- Continue the process until all plaintext blocks have been processed by feeding the ciphertext block into the top register.
- IV data is XORed with the first decrypted ciphertext block in order to decrypt it. In order to decrypt the next ciphertext block, the previous ciphertext block is added as well into the register that replaces IV.

- In CBC mode, the previous ciphertext block and the current plaintext block are combined, and the outcome is then encrypted using the key. Decryption, then, is the opposite procedure, in which the previous ciphertext block is added to the result after the current ciphertext has been decrypted.

- The benefit of CBC over ECB is that an identical message can use a different ciphertext when the IV is changed. On the negative side, the chaining effect causes the transmission error to propagate to a few further blocks during decryption.

- It is important to note that CBC mode provides the base for a well-known data origin authentication system. Thus, it benefits applications which need both symmetric encryption and data origin authentication.

Bit-Width of CBC Mode

The following table show the bit-width of the interfaces that CBC mode offer −

	plaintext	**ciphertext**	**cipherkey**	**IV**
CBC-DES	64	64	64	64
CBC-AES128	128	128	128	128
CBC-AES192	128	128	192	128
CBC-AES256	128	128	128	128

- A common block cipher mode of operation that makes use of the block cipher algorithm is the Cipher Block Chaining (CBC) mode. It has the ability to process both the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES) in this version.
- The cipherkey length for AES should be 128/192/256 bits, and 64 bits for DES. Another drawback is that, but text in the real world comes in a range of lengths, our working mode only supports units of a fixed size (64 or 128 bits for a single block). Therefore, before encryption or decryption, the final block of text given to this primitive needs to be padded to 128 bits.

Formula for CBC mode

This method can be expressed as follows if it is put it into a formula −

$$Ci = EK(Bi \oplus Ci\text{-}1)$$

where Ci-1 is the cipher that corresponds to Bi-1 and EK is the block encryption technique with key K.

Note: It is assuming that C0 is the initialization vector in the formula above.

In the same way, the CBC can be used for decryption by using −

$$Bi = DK(Ci) \oplus (Ci\text{-}1)$$

Where DK stands for the block decryption technique with key K.

For decryption, the same initialization vector (C0) will be used.

Security Challenge

The primary attributes of this scheme are as follows −

- One additional particular block will be damaged if even a single bit of the conveyed message gets altered or lost. This harm wouldn't affect other blocks.
- In the event that a bit is lost or added to the ciphertext, the bits and block borders will move, producing an incorrect decoding of all ensuing blocks of the cryptography's ciphertext.
- The factor has the ability to append blocks to the end of the deciphered message, thereby enhancing it with either the original or plain text.

Advantages

Cipher Block Chaining (CBC) mode has several advantages −

- When input data is more than one block in size (usually 128 bits or 16 bytes for AES), CBC operates well. Messages of any length can be encrypted and decrypted by breaking them up into blocks and joining them.
- Although CBC itself cannot provide authentication, it can be used in combination with digital signatures or HMAC (Hash-based Message Authentication Code) in order to ensure the integrity and validity of the encrypted data. CBC can offer a complete security solution by combining authentication and encryption.
- When it comes to cryptanalysis, CBC is typically safer than ECB (Electronic Codebook) mode. An attacker finds it more difficult to identify patterns in the encrypted data when using CBC, because each block of ciphertext depends on the one before it. Due to this, CBC is more effective for securing private data from attacks or attempts to decipher it.

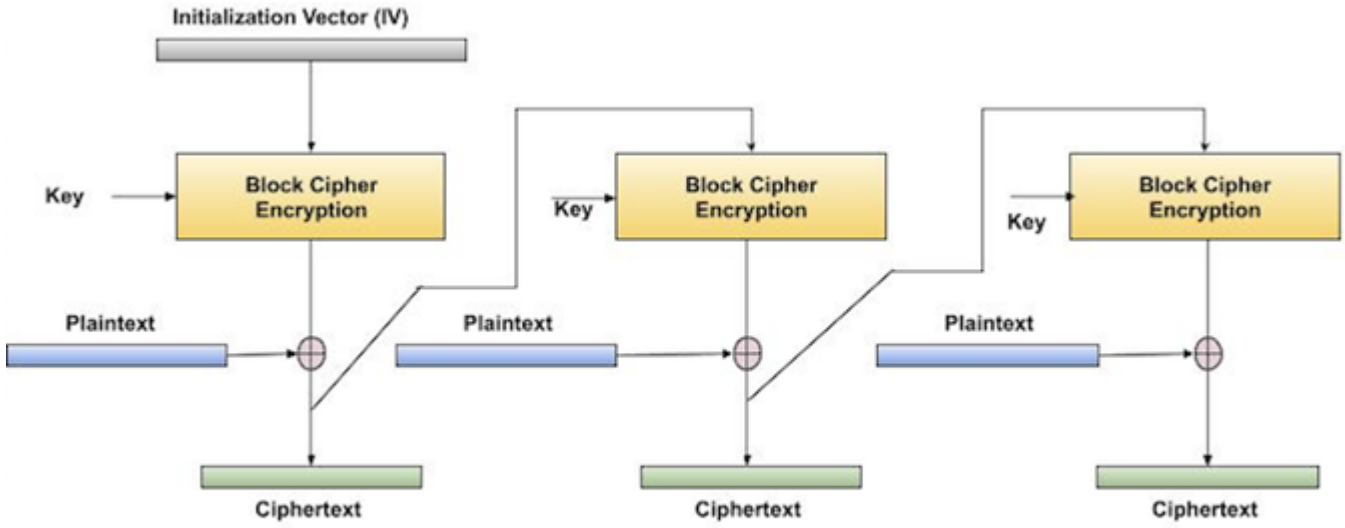
Disadvantages

- The inability of Cipher Block Chaining (CBC) mode for sending out concurrent encryption is one of its drawbacks. Before encryption, each plaintext block in CBC mode is XORed with the previous ciphertext block. This creates a dependency on the previous ciphertext block for the current block's encryption. As each block's encryption depends on the previous block's encryption being finished, the following ones cannot be encrypted concurrently or in parallel.
- In comparison to modes like Electronic Codebook (ECB) or Counter (CTR) mode, which can carry out concurrent encryption of numerous blocks simultaneously, the sequential nature of encryption in CBC mode

can result in slower encryption performance. But because of its security benefits and compatibility for a wide range of encryption scenarios, CBC mode is still widely used despite of this limitation.

Cipher Feedback (CFB) Mode

- The Cipher Feedback (CFB) mode is quite similar to CBC; the main difference is that CFB is a stream mode. It eliminates patterns by using feedback (also known as chaining in stream modes). CFB, like CBC, uses an initialization vector to destroy patterns while propagating errors.
- The CFB mode is a typical block cipher mode of operation that uses the block cipher algorithm. In this version, we support Data Encryption Standard (DES) and Advanced Encryption Standard (AES) processing; the cipherkey length for DES should be 64 bits, and 128/192/256 bits for AES. Another constraint is that our working mode operates on fixed-size units (64 or 128 bits per block), while text in the actual world varies in length.
- As a result, the final block of text given to this primitive must be padded to 128 bits before it can be encrypted or decrypted. While CFB1 and CFB8 modes use the same interface as CFB128 mode, CFB1 and CFB8 modes process plaintext and ciphertext bit-by-bit or byte-by-byte rather than block-by-block.



Cipher Feedback (CFB) Mode Encryption

Operation

The operation of the CFB mode is shown in the above image. In the current system, a message block has a size of 's' bits, where 1 < s < n. The initial random n-bit input block in the CFB mode needs to have an initialization vector (IV). The Initialization Vector does not need to be kept secret. Steps for operation are −

- Load the IV into the top register.

- Encrypt the data value in the top register with the underlying block cipher and key K.
- To construct the ciphertext block, take only 's' number of most important bits (left bits) from the encryption process's output and XOR them with 's' bit plaintext message block.
- Feed the ciphertext block into the top register by moving the existing data to the left, and repeat the operation until all plaintext blocks are processed.
- Basically, the previous ciphertext block is encrypted with the help of the key, and the result is XORed with the current plaintext block.
- Decryption follows similar methods. A predetermined IV is loaded at the start of decryption.

Analysis of CFB Mode

- CFB mode differs considerably from ECB mode in that the ciphertext linked to a given plaintext block is determined not only by that plaintext block and the key, but also by the previous ciphertext block. In other words, the ciphertext block is dependent on the message.
- CFB has a highly unusual property. In this mode, the user decrypts the ciphertext using the block cipher's encryption method. The decryption algorithm for the underlying block cipher is never used.
- Apparently, CFB mode converts a block cipher into a stream cipher. The encryption algorithm functions as a key-stream generator, producing a key-stream that is stored in the bottom register. This key stream is then XORed with the plaintext, as is the case with stream ciphers.
- CFB mode converts a block cipher into a stream cipher, giving it some of the advantages of a stream cipher while preserving the benefits of a block cipher.
- On the other hand, transmission errors proliferate as a result of block changes.

Formula for CFB Mode

- CFB (cipher feedback) is an AES block cipher mode similar to CBC in that it needs the previous block's cipher, Ci-1, to encrypt a block, Bi. Similar to CBC, CFB uses an initialization vector. The main difference is that in CFB, the previous block's ciphertext block is encrypted first before being XOR-ed with the block in focus.
- To better understand this, consider CFB in the form of a formula −

$$Ci = EK(Ci\text{-}1) \oplus Bi$$

where EK represents the block encryption algorithm with key K, and Ci-1 is the cipher for Bi-1.

Note that the calculation above assumes C0 to be the initialization vector.

- Similarly, decryption using the CFB can be represented as follows −

It is important to understand that the decryption algorithm is not used here.

Bit-Width of CFB Mode

The below table show the bit-width of the interfaces that CFB mode offer −

	plaintext	ciphertext	cipherkey	IV
CFB1-DES	64	64	64	64
CFB1-AES128	128	128	128	128
CFB1-AES192	128	128	192	128
CFB1-AES256	128	128	256	128
CFB8-DES	64	64	64	64
CFB8-AES128	128	128	128	128
CFB8-AES192	128	128	192	128
CFB8-AES256	128	128	256	128
CFB128-DES	64	64	64	64
CFB128-AES128	128	128	128	128

CFB128-AES192	128	128	192	128
CFB128-AES256	128	128	256	128

Advantages of CFB Mode

- CFB (Cipher Feedback) mode can sometimes be faster than CBC (Cipher Block Chaining) mode because it does not need an additional decryption technique. This can improve performance, particularly in situations when encryption speed is essential.
- CFB mode uses non-deterministic encryption, which means it cannot see patterns in the plaintext. This provides an additional layer of security by making it more challenging for attackers to determine information about the plaintext from the ciphertext.

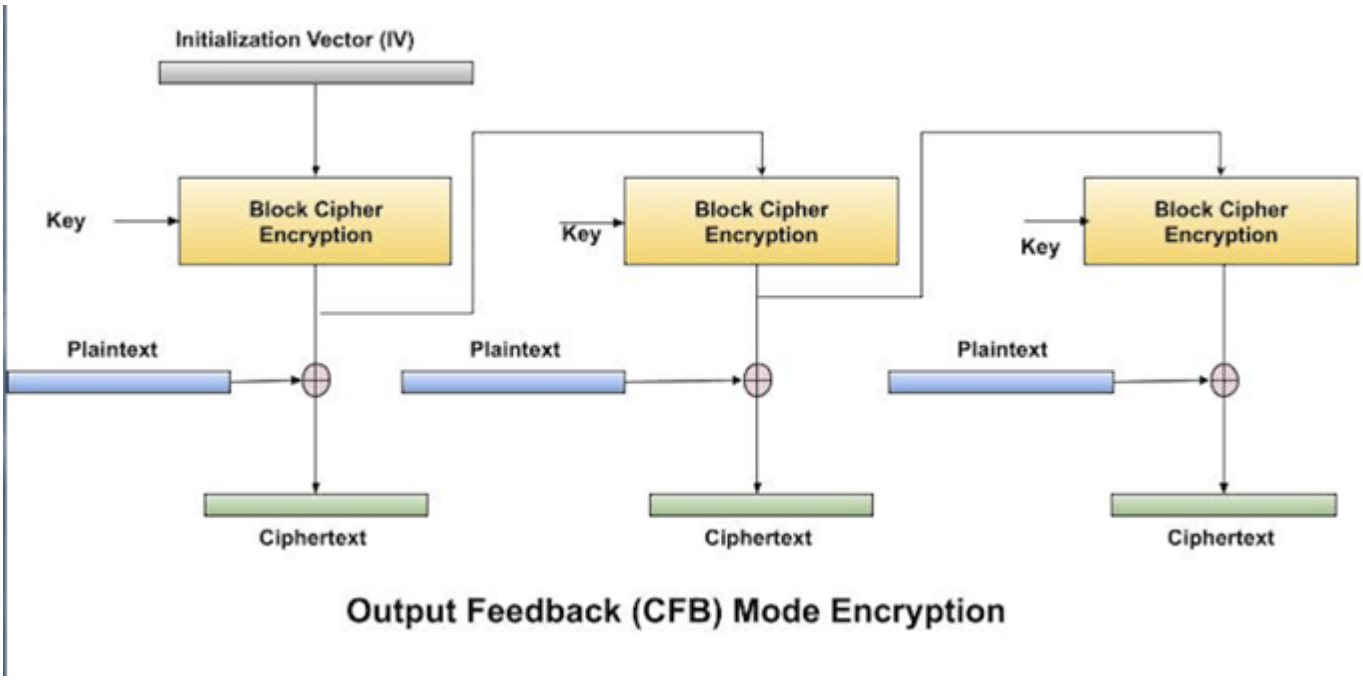
Disadvantages of CFB Mode

- Just like CBC mode, CFB mode cannot handle the loss of encrypted blocks. If even a single block is lost or corrupted throughout transmission, it might cause the decryption process to fail and leaving the entire message inaccessible.
- Like CBC mode, CFB mode does not allow for concurrent encryption of many blocks. Each block's encryption is dependent on the previous ciphertext block, which limits the efficiency of parallel encryption techniques and may affect overall encryption speed.
- While CFB mode can offer advantages in terms of speed and unpredictable encryption, it is more difficult to set up and understand than simpler modes like ECB (Electronic Codebook) mode. This additional level of complexity can lead to potential vulnerabilities if not properly implemented.

Output Feedback (OFB) Mode

- The Output Feedback (OFB) Mode is similar to the Cipher Feedback mode, but it delivers encrypted output instead of the XOR output. This output feedback option sends all block bits rather than just a subset of them. The output feedback mode of block cipher is extremely vulnerable to bit transmission errors. This reduces the dependecy of cipher's need on plaintext.
- It involves inserting the successive output blocks from the underlying block cipher back into it. In CFB mode, feedback blocks send a string of bits to the encryption algorithm, which then generates the key-stream.

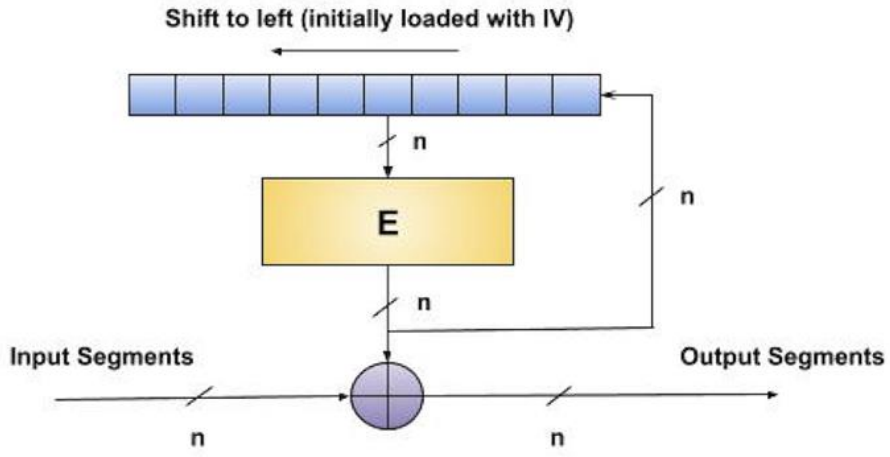
- The OFB mode needs an IV as the first random n-bit input block. The Initialization Vector does not need to be kept secret.
- The operation of OFB mode is shown in the below image −



Output Feedback (CFB) Mode Encryption

Operation

The fundamental difference between OFB and CFB is that the OFB mode uses XOR to combine plaintext and ciphertext blocks with larger copies of the initialization vector.



Output Feedback (OFB) Mode

- This operation can be considered as a one-time pad, and the expanded vectors as pad vectors. The formula below shows how a series of pad vectors is created −

$$Vi = EK(Vi-1)$$

where EK represents the block encryption technique with key K, and Vi and Vi-1 are adjacent vectors.

Note that the calculation above assumes V0 to be the initialization vector.

After creating a series of pad vectors, the following formula can be used to execute encryption in the OFB mode −

$$Ci = Vi \oplus Bi$$

Decryption works in a similar way −

$$Bi = Vi \oplus Ci$$

Note that, like the CFB mode, OFB uses the same encryption algorithm for both encryption and decryption.

Analysis of OFB Mode

- OFB mode gives high security, successful parallel processing, and tolerance for block loss. To avoid any security issues, care must be taken to ensure that the initialization vector is random and unique. Also, the lack of error propagation can call for more precautions to protect data integrity in the case of transmission problems.

Bit-Width of OFB Mode

The following table show the bit-width of the interfaces that OFB mode offer −

	plaintext	**ciphertext**	**cipherkey**	**IV**
OFB-DES	64	64	64	64
OFB-AES128	128	128	128	128
OFB-AES192	128	128	192	128
OFB-AES256	128	128	256	128

Advantages of CFB Mode

- Once the pad vectors (also known as keystream) have been created, the OFB (Output Feedback) mode enables for block encryption and decryption in simultaneously. This can lead to improved performance in situations where processing speed is important, as numerous blocks can be handled concurrently.
- As each block is encrypted independently of the others, OFB mode can withstand the loss of encrypted blocks during transmission. This means that if a block is lost or corrupted, succeeding blocks' decryption will be unaffected, allowing for better error recovery.

Disadvantages of OFB Mode

- Repeatedly encrypting the initialization vector (IV) in OFB mode can generate the same state, possibly revealing a security vulnerability. If the IV is not properly random or changes predictably, the encryption process can reuse the same keystream, allowing an attacker information about the plaintext.
- Unlike other modes, like CBC (Cipher Block Chaining), OFB mode does not transmit errors. While this can be beneficial in some cases since it prevents errors from affecting succeeding blocks, it also means that problems in the ciphertext may not be recognised or fixed, potentially leading to data damage or security vulnerabilities.

Counter (CTR) Mode

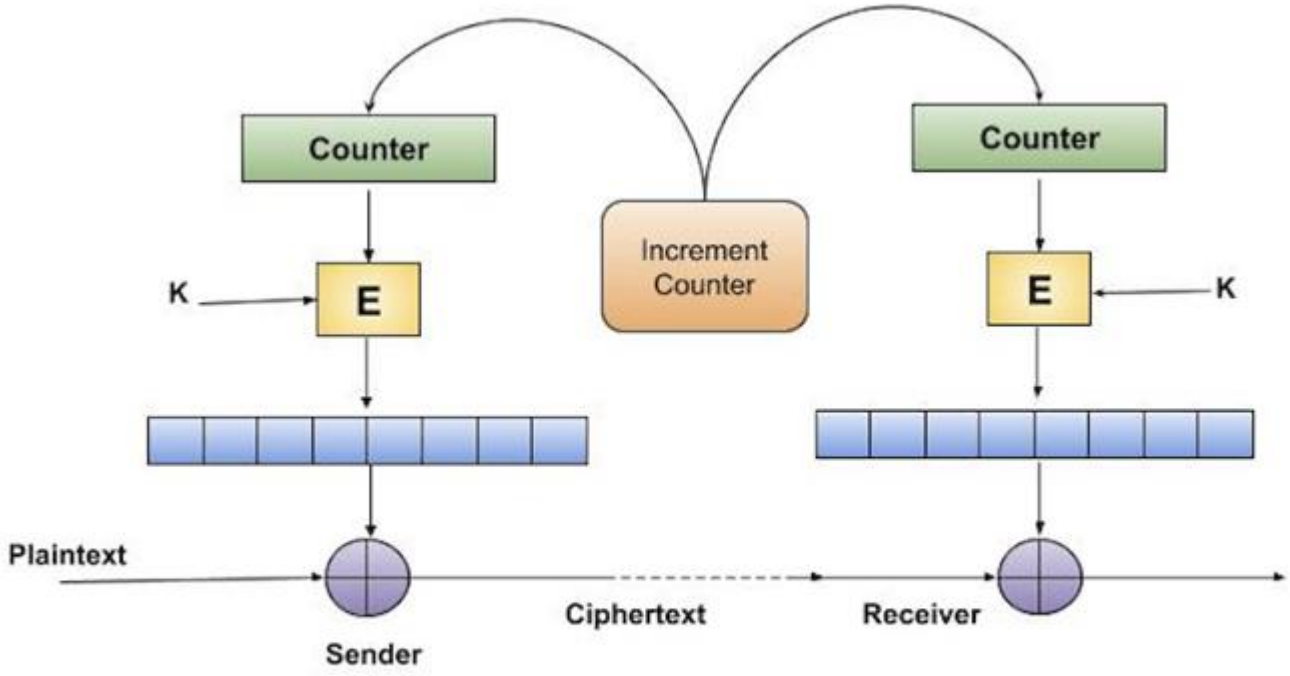
- Counter Mode (CTR) is similar to OFB, with one difference that CTR uses a counter for feedback. This method has the same advantages as OFB (patterns are destroyed and errors are not transmitted), but it also allows for parallel encryption because the feedback can be as simple as an ascending number.
- A simple example is that the first block is XORed with the number 1, the second with the number 2, and so on. This method allows for the simultaneous completion of any number of rounds.
- It can be thought of as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver must have access to a reliable counter that generates a new shared value each time a ciphertext block is transferred. This shared counter is not always a secret value; though, both parties must keep the counter synchronised.

Operation

The following image shows encryption and decryption in CTR mode. Steps in operation are as follows −

- Load the top register with the initial counter value that is the same for both the sender and receiver. It provides the same purpose as the IV in CFB (and CBC) mode.
- Encrypt the data of the counter with the key and save the result in the bottom register.

- Take the first plaintext block (P1) and XOR it with the data of the bottom register. The outcome of this is C1. Send C1 to the receiver, then update the counter. The counter update a substitutes the ciphertext feedback in the CFB mode.
- Continuing in this manner until the final plaintext block is encrypted.
- Decryption is an opposite process. The ciphertext block is XORed with the encrypted data of the counter value. Each ciphertext block counter is updated after decryption, exactly as it was when encrypted.



Couter (CTR) Mode

Analysis of CTR Mode

- It has no message dependency, hence a ciphertext block is not dependent on preceding plaintext blocks.
- Just like CFB mode, CTR mode does not include block cipher decoding. This is because the CTR mode generates a key-stream with the block cipher and then encrypts it with the XOR function. In other words, CTR mode changes a block cipher into a stream cipher.
- The major disadvantage of CTR mode is that it needs the use of synchronous counters at both the transmitter and receiver. Loss of synchronisation causes insufficient plaintext recovery.
- However, CTR mode offers practically all of the same advantages as CFB mode. Also, no transmission errors are propagated.

Formula for CTR Mode

- CTR is similar to OFB in that it XORs a series of pad vectors with plaintext and ciphertext blocks. The primary difference is how these pad vectors are created.
- In the CTR mode, we begin with a random seed, s, and then compute pad vectors using the formula −

$$Vi = EK(s+i-1)$$

where EK is the block encryption technique with key K, Vi is a pad vector, and i is the vector's offset from 1.

- Once the vectors have been constructed, encryption comparable to the OFB mode can be performed using the following formula −

$$Ci = Vi \oplus Bi$$

- Decryption works in a similar way −

$$Bi = Vi \oplus Ci$$

- CTR uses the same encryption algorithm for both encryption and decryption just like CFB and OFB modes.

Bit-Width of CTR Mode

- The Counter (CTR) mode is a typical block cipher mode of operation that uses the block cipher algorithm.In this version, we offer Advanced Encryption Standard (AES) processing; the cipherkey length for AES is 128/192/256 bits.
- Another constraint is that our working mode operates on units of a set size (128 bits per block), but text in the actual world has a variety of lengths. As a result, the final block of text provided to this primitive must be padded to 128 bits before it can be encrypted or decrypted.
- The following table show the bit-width of the interfaces that CTR mode offer −

	plaintext	ciphertext	cipherkey	IV
CTR-AES128	128	128	128	128
CTR-AES192	128	128	192	128
CTR-AES256	128	128	256	128

Advantages of CTR Mode

So below are some advantages of counter (CTR) mode −

- **Hardware efficiency** − Unlike the three chaining modes, CTR mode allows encryption (or decryption) to be performed in parallel on many blocks of plain-text or ciphertext. For chaining modes, the algorithm has to complete the computation on one block before proceeding to the next. This limits the algorithm's maximum throughput to the reciprocal of the time required for a single execution of block encryption or decryption. In CTR mode, throughput is just limited by the amount of parallelism obtained.

- **Software efficiency** − Additionally, while CTR mode supports parallel execution, processors with parallel capabilities like aggressive pipelining, multiple instruction dispatch per clock cycle, a high number of registers, and SIMD instructions can be properly used.

- **Preprocessing** − The underlying encryption technique is executed regardless of whether the plaintext or ciphertext is given. As a result, considering enough memory is available and security is maintained, preprocessing can be utilised to prepare the output of the encryption boxes, which feed into the XOR functions. When the plaintext or ciphertext input is given, the only operation performed is a series of XORs. As an approach significantly increases throughput.

- **Random access** − The ith block of plaintext or ciphertext is possible to handled using random access. With the chaining modes, block Ci cannot be computed before the i - 1 preceding block is computed. There are applications where a ciphertext is kept and just one block needs to be decrypted; in these applications, the random access functionality is useful.

- **Simplicity** − CTR mode is simpler than ECB and CBC modes since it only requires the encryption algorithm to be implemented, not the decryption algorithm. This is especially important when the decryption algorithm differs significantly from the encryption algorithm, as is the case with AES. Also, there is no requirement to create decryption key scheduling.

Disadvantages of CTR Mode

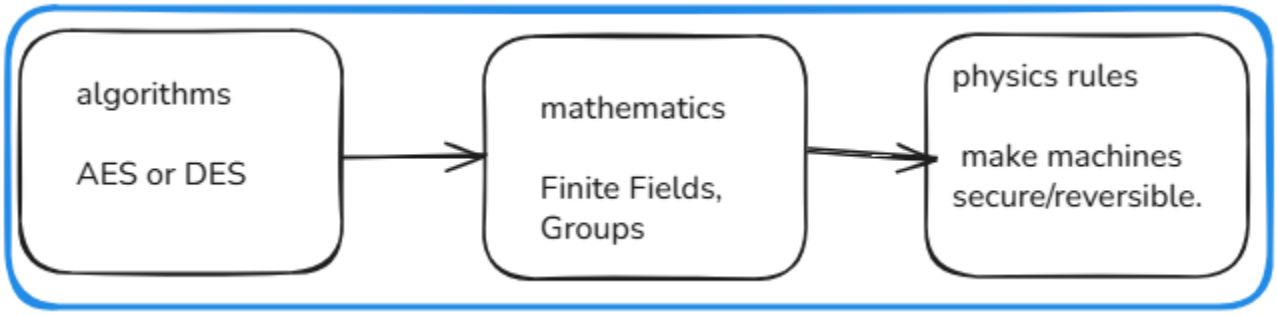
- The main drawback of the CTR is that a synchronised counter must be maintained at both the receiving and sending destinations. Losing track of this counter may result in incorrectly restoring plaintext.

Finite Fields: Groups Rings, Fields, Modular Arithmetic, Euclidean Algorithm, Galois Fields (GF(p) & GF(2n)), Polynomial Arithmetic

Finite Fields: Groups Rings, Fields	**Modular Arithmetic, Euclidean Algorithm**
Galois Fields (GF(p) & GF(2n)),	**Polynomial Arithmetic**

- ✓ The mathematical concepts (Finite Fields, Modular Arithmetic, etc.) are the **engine** that makes the encryption algorithms (AES, DES, IDEA) work.

- ✓ Think of it this way: The **algorithms** (like AES or DES) are the, and the **mathematics** (Finite Fields, Groups) are the *physics rules* that make those machines secure and reversible.



- ✓ Here is the connection between the math and the algorithms:

1. Finite Fields (Galois Fields) : The Backbone of AES

In standard math, numbers go on forever (infinite). In computers, we only have limited space (like 8 bits).

- **The Math:** A **Finite Field**, specifically GF(2^8), is a mathematical "sandbox" where we can add, subtract, multiply, and divide numbers, but the result always fits perfectly inside 8 bits.
- **The Connection: AES** relies entirely on this. When AES mixes bytes (in the "MixColumns" step), it isn't just shuffling them; it is performing mathematical multiplication inside this Galois Field. This guarantees that the mixing is perfect and reversible.

2. Euclidean Algorithm : Creating Secure S-Boxes

- **The Math:** The Euclidean Algorithm is used to find the "Greatest Common Divisor," but its Extended version can find the **Multiplicative Inverse** (the number you multiply by to get 1).
- **The Connection:** To make encryption secure, we need to scramble data in a way that is hard to analyze but easy to reverse if you have the key. The **S-Box** (Substitution Box) in **AES** is built by calculating the multiplicative inverse of bytes using the Euclidean Algorithm. This makes the cipher resistant to mathematical attacks.

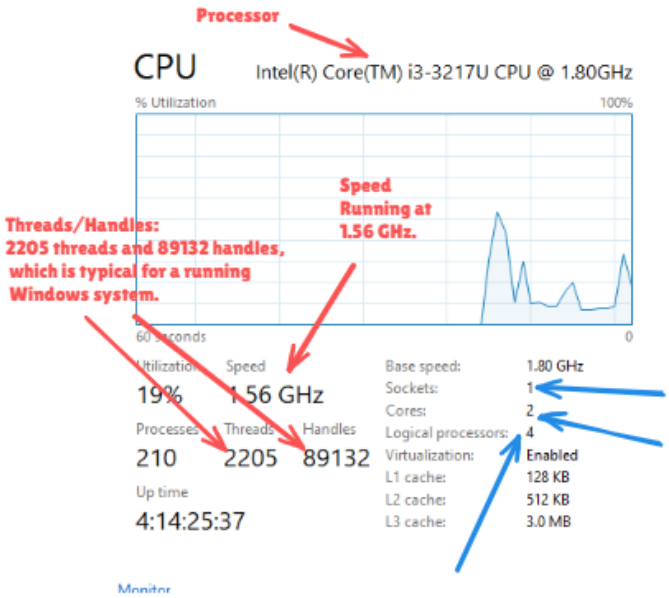
3. Groups and Rings : The Strength of IDEA

- **The Math:** In algebra, **Groups** and **Rings** define sets of rules (like "how addition works" vs. "how multiplication works").
- **The Connection:** The **IDEA** cipher gets its security by forcing an attacker to solve equations that mix three different mathematical groups:
    1. **XOR** (Group 1)
    2. **Addition** Modulo 2^{16} (Group 2)
    3. Multiplication Modulo 2^{16}+1 (Group 3)

Because these mathematical rules don't play nice together, the encryption is incredibly hard to break.

4. Polynomial Arithmetic : Speed on Hardware

- **The Math: Polynomial Arithmetic** treats a string of binary bits (like 1011) as an algebraic equation (like x^3 + x + 1).
- **The Connection:** This allows complex encryption steps (like those in **AES**) to be calculated incredibly fast on computer chips. Instead of doing slow division, the processor can just "shift" bits left or right, which is mathematically equivalent to polynomial operations.



5. XOR (Field Characteristic 2) : Feistel, DES, & Modes

- **The Math:** In the simplest Finite Field GF(2), addition is exactly the same as the **XOR** operation. It has a "magic" property where if you XOR the same thing twice, it cancels out $(A \oplus B \oplus B = A)$.
- **The Connection:**
    - **Feistel Structure (used in DES):** Relies on this XOR property so that the exact same hardware circuit can be used for both Encryption and Decryption.
    - **Block Cipher Modes (CBC, CTR):** These modes use XOR to chain blocks together. They are essentially adding the keystream to the plaintext using Finite Field addition.

Finite Fields

Sdsds

Sdsds