# 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*

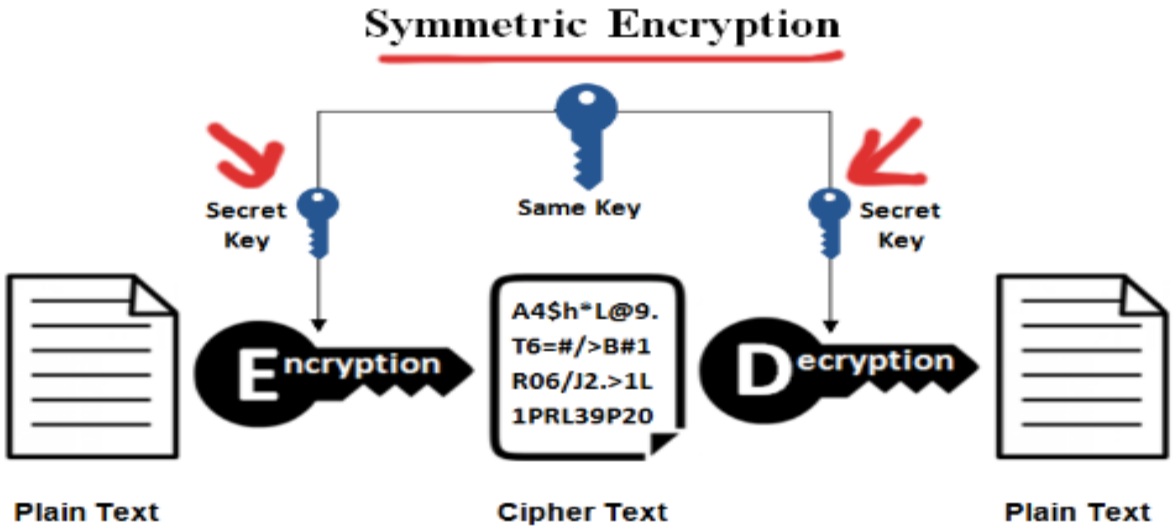
*2.6. Finite Fields: Groups Rings, Fields, Modular Arithmetic, Euclidean Algorithm, Galois Fields (GF(p) & GF(2n)), Polynomial Arithmetic*

*2.7. International Data Encryption Standard (IDEA)*

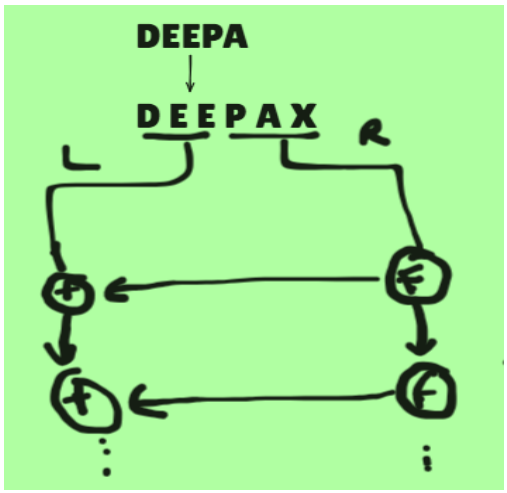
*2.8. Advanced Encryption Standards (AES) Cipher*

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

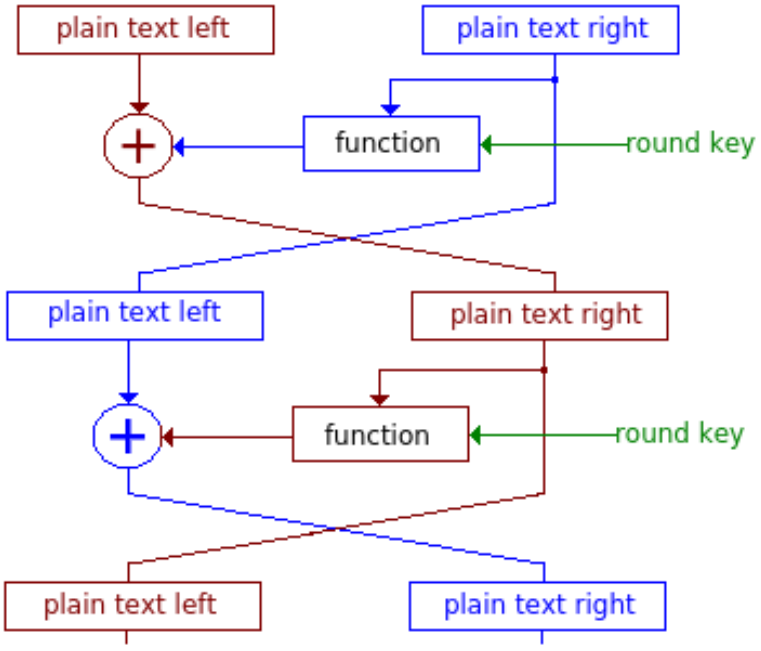
**Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist**




# 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**

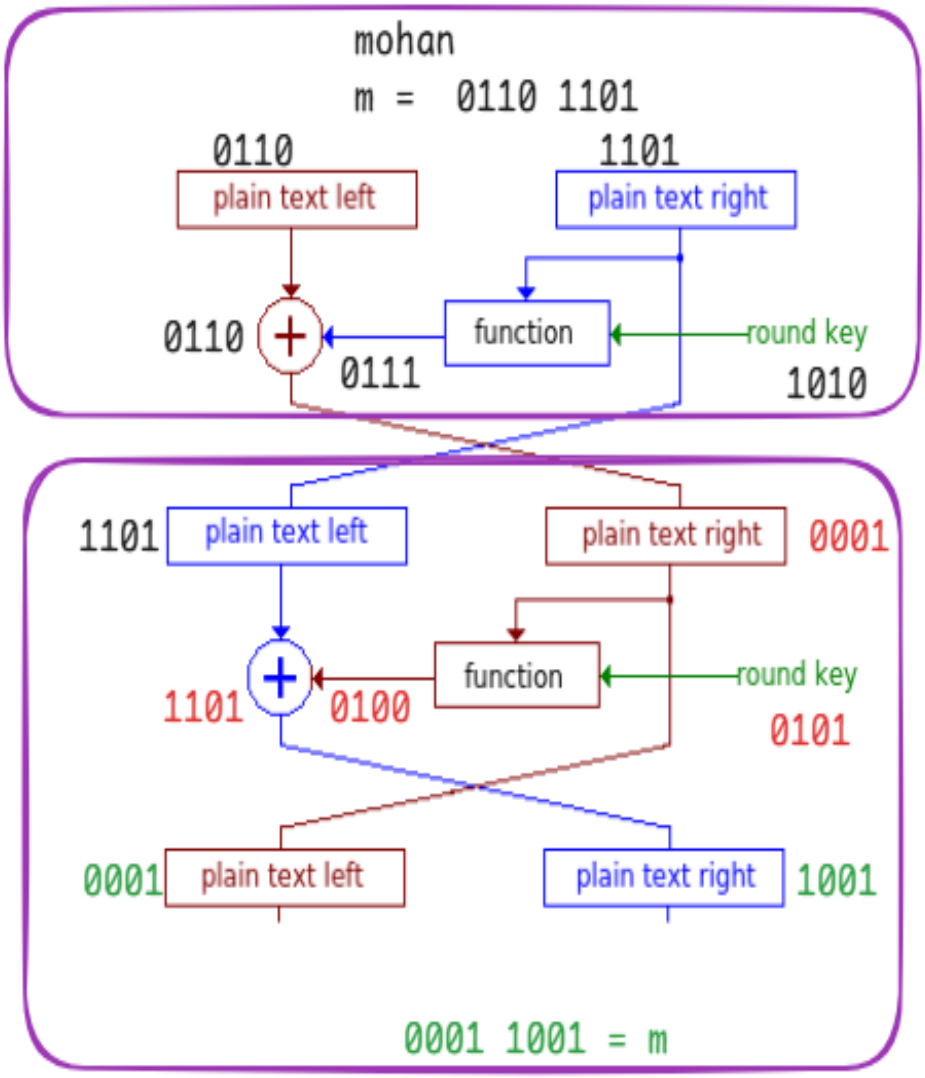
R2 = L1 XOR F

= 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"**.

---

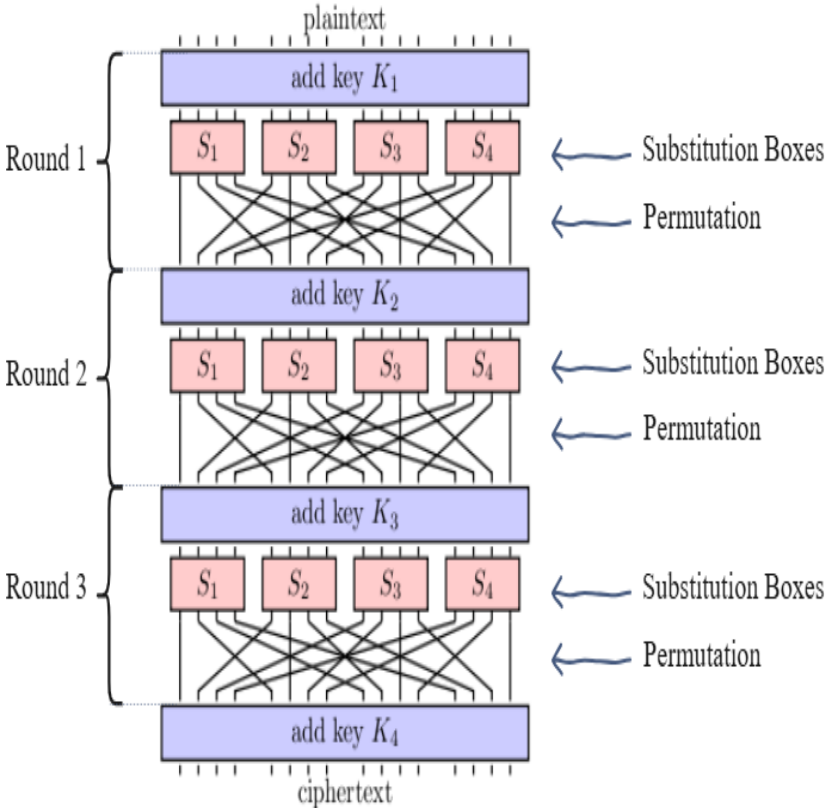
**Why decryption works (important exam point)**

- 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.



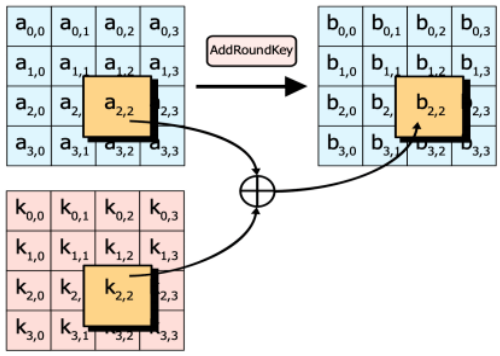

### Operations in an SP network

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**

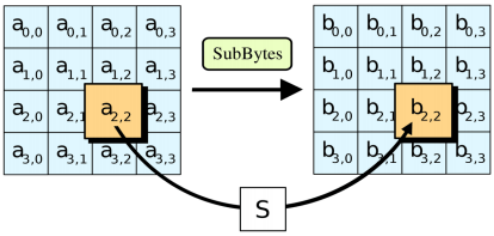
### Addition of the round key

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.



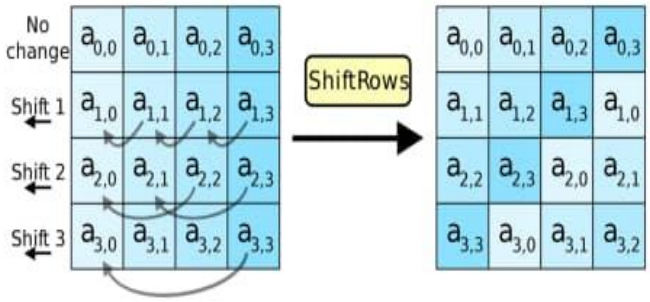

### Substitution of bits

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.



**Permutation of bits**

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:



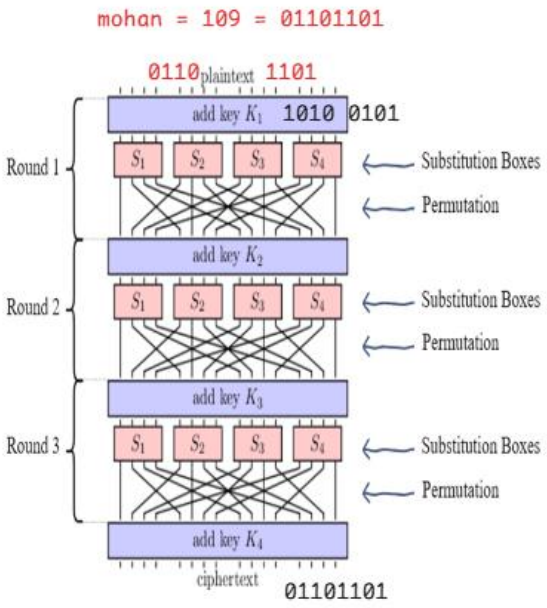

Note: The substitution and permutation boxes are not kept hidden in SP networks. Only the round keys are kept secret for security.

# 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.

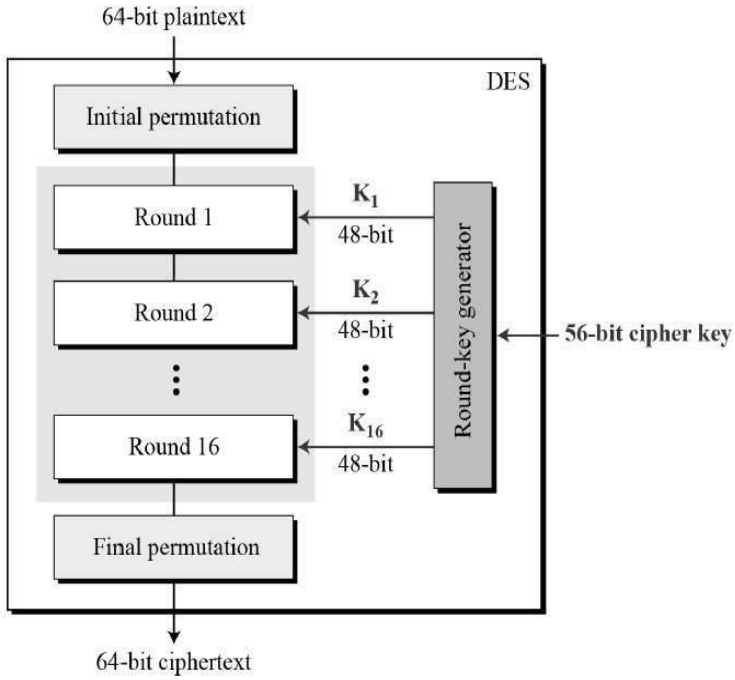
---

**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/



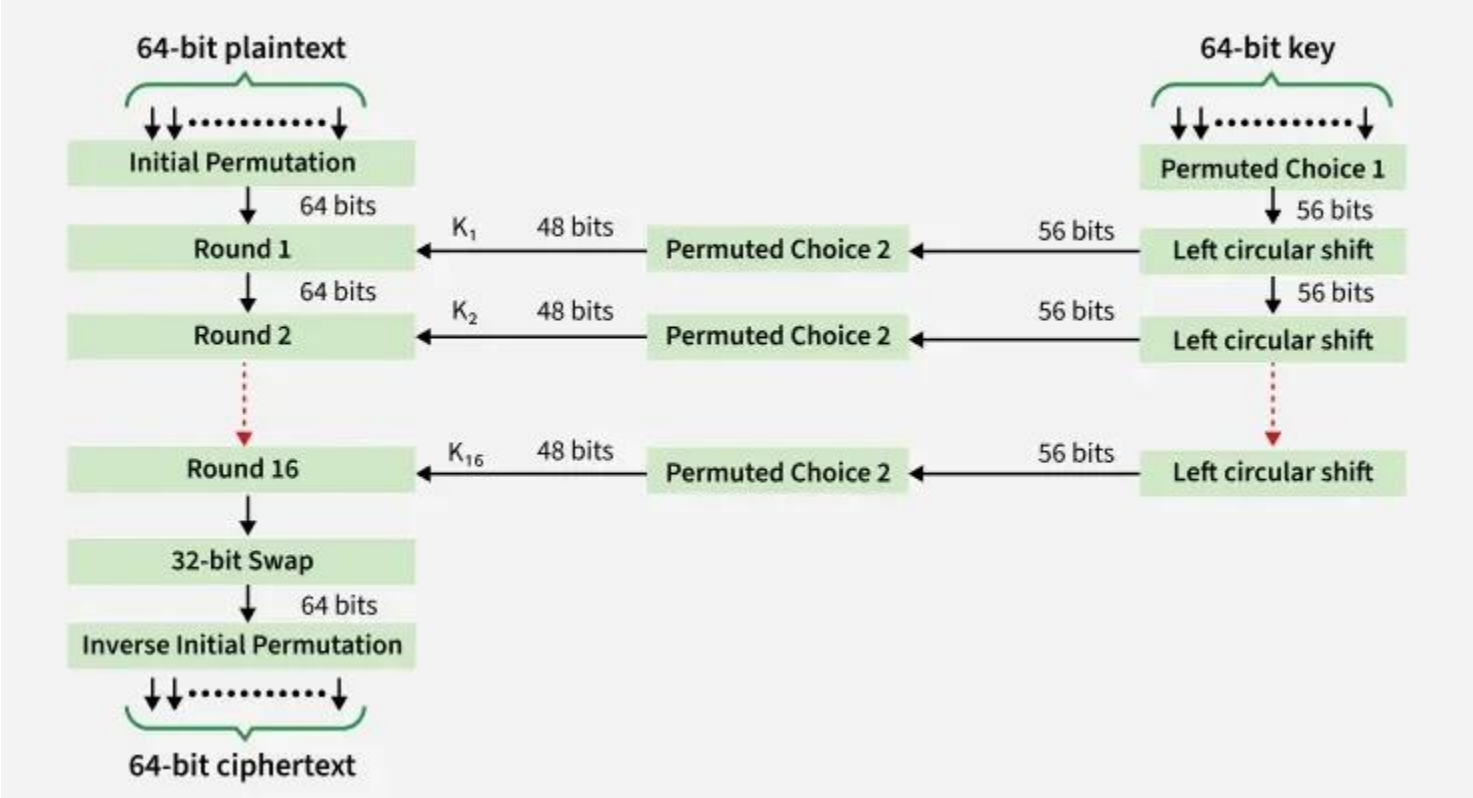

DES is an implementation of a Feistel Cipher. It uses **16 round** Feistel structure. The **block size is 64-bit(PT/CT)**. Though, key length **is 64-bit, DES** has an **effective key length of 56 bits key (64 bits including parity)**, since 8 of the 64 bits of the key are not used by the encryption 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**

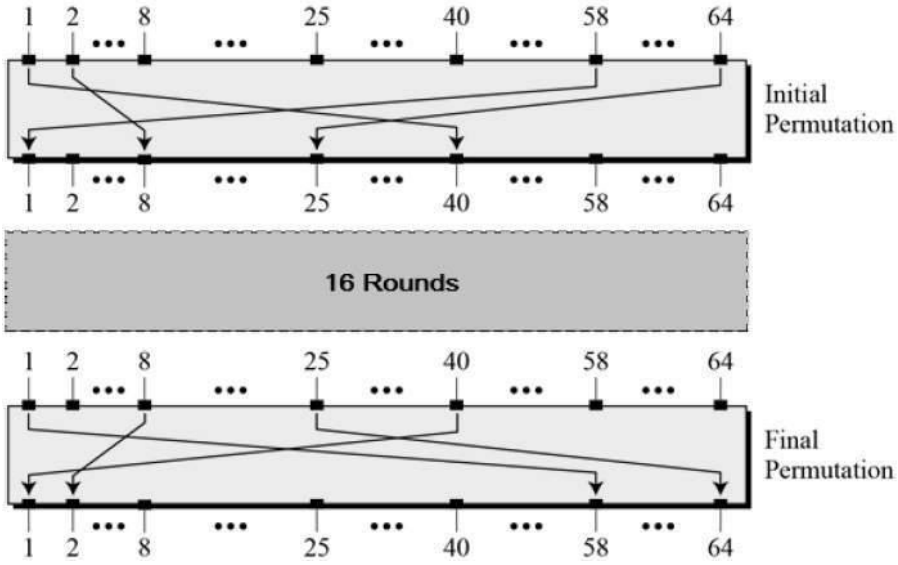
DES Block & Key Sizes

| 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**

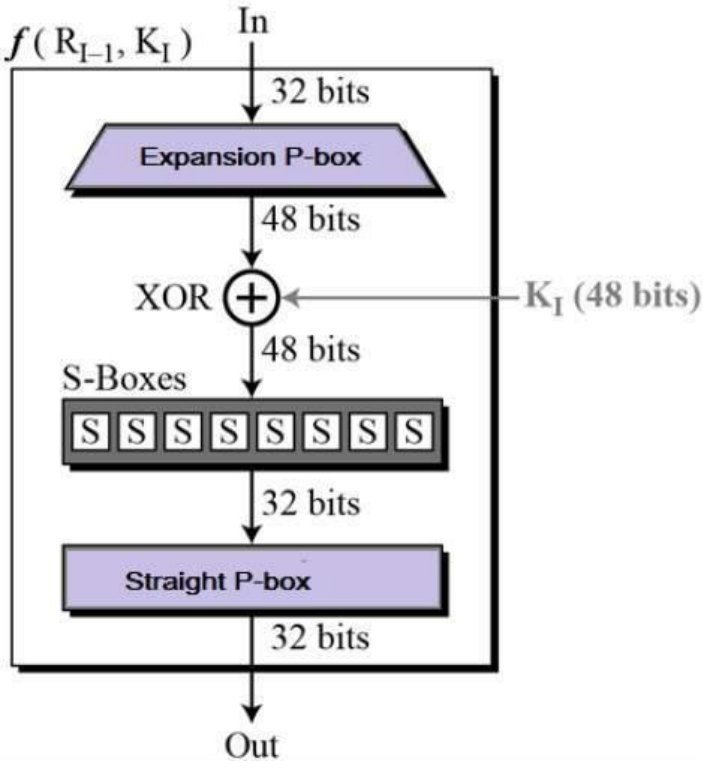
**Ri = Li−1 ⊕ F(Ri−1, Ki)**

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**

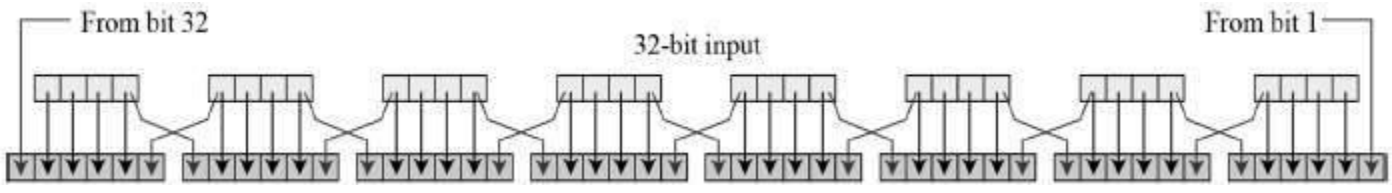
👉 **This is the ONLY non-linear part of DES**

**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 −



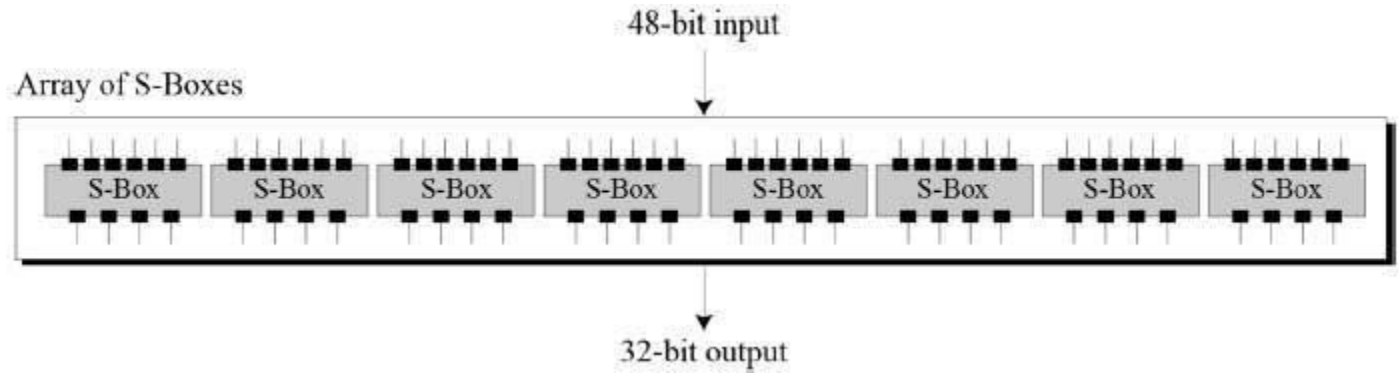

- Expands **32-bit R** to **48 bits**
- Some bits are repeated
- Purpose: Match key size and provide diffusion
- 32 bits → 48 bits

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 |

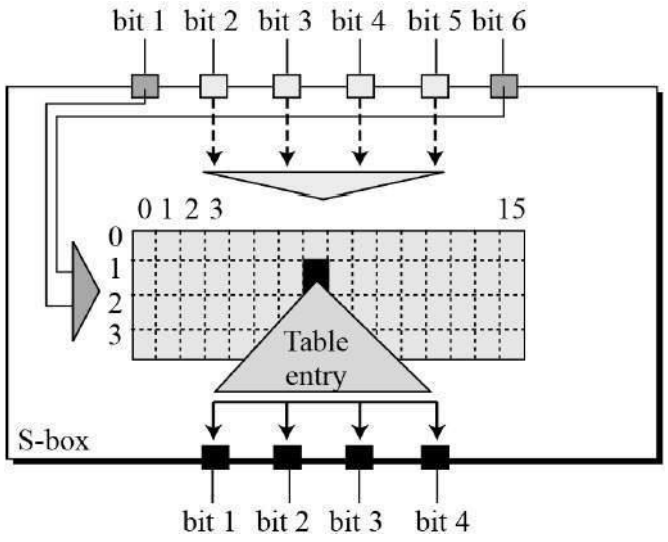
**XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

**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 –

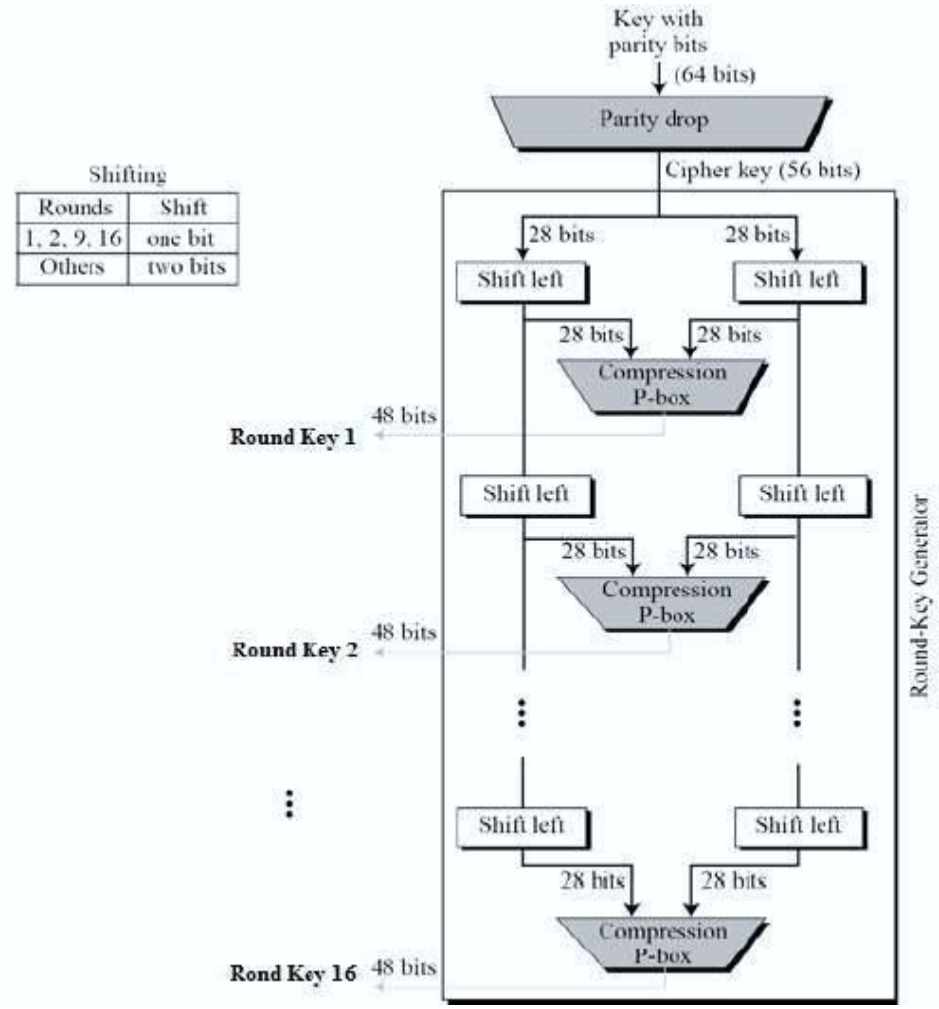
The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

- 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.

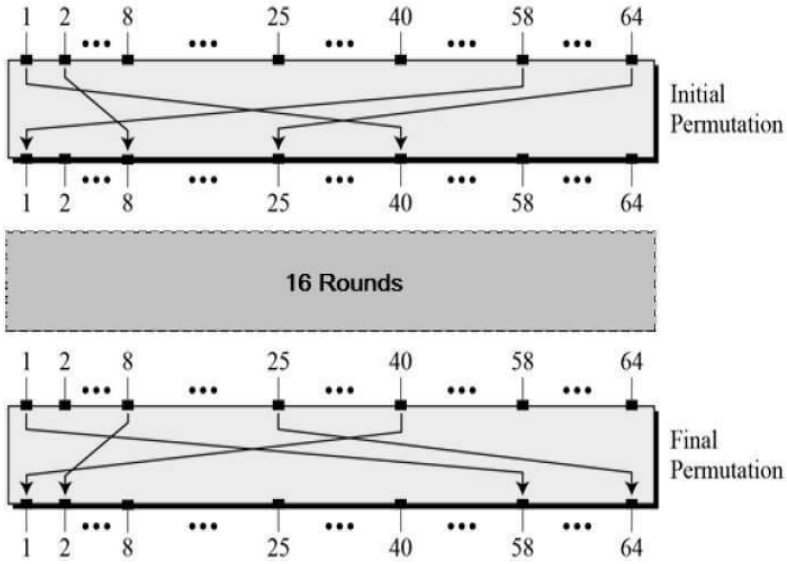
**Steps:**

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

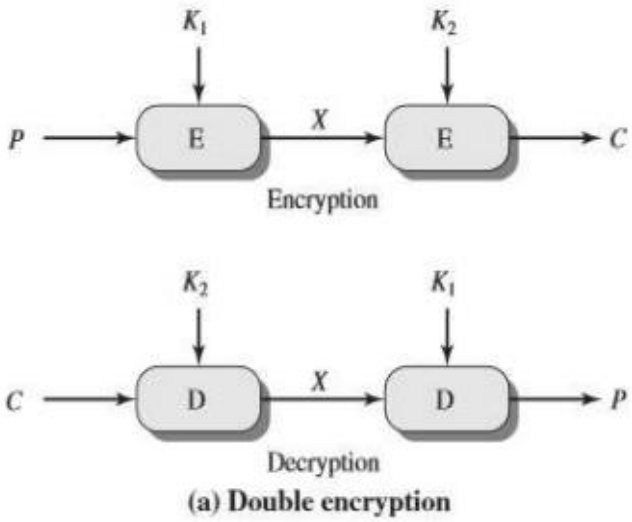
DES problems:

- 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

---

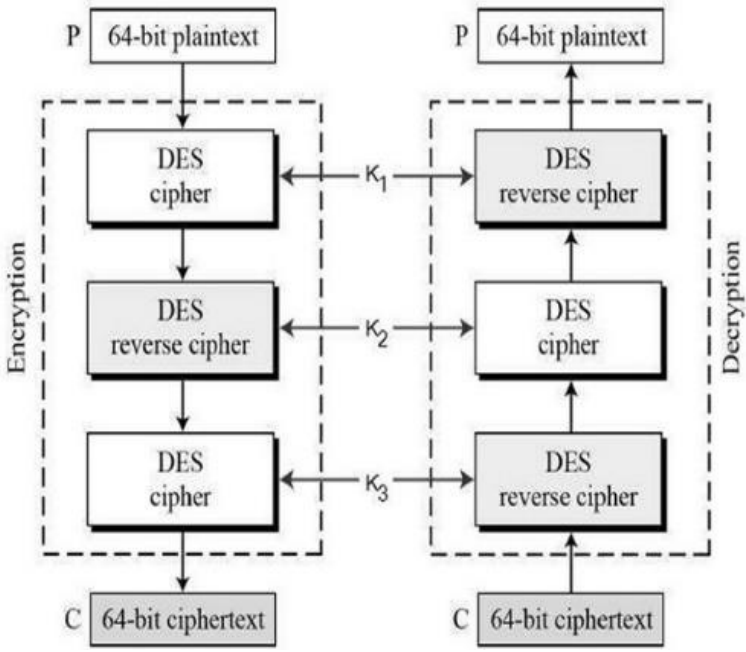
### DOUBLE DES

**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.

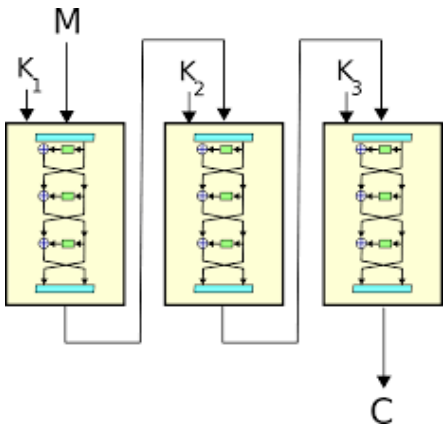
---

## ❌ 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

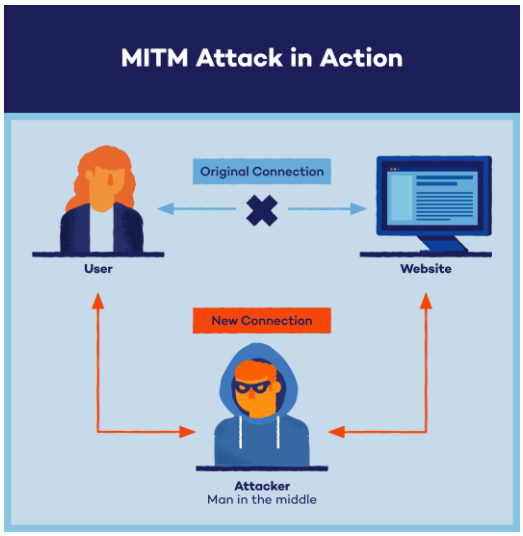
---

| 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.




**HW: Why Avlance Effect?**

# International Data Encryption Standard (IDEA)

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

**Sdsds**

**Sdsds**