

Cryptography Notes (Depth Notes)

Feistel, SPN, DES/2DES/3DES, and Finite Fields

(Reference-focused on William Stallings)

Prepared by **Er. Arjun Neupane**

January 11, 2026

Contents

1	Confusion and Diffusion (Important Concepts)	4
1.1	Meaning and purpose (core idea)	4
1.2	Confusion	4
1.2.1	Effect of changing a key bit (avalanche idea)	4
1.3	Diffusion	4
1.3.1	Effect of changing a plaintext bit	4
1.4	“Vagueness” and “Redundancy” interpretation (as in your note)	5
1.5	Stream cipher vs Block cipher (confusion and diffusion)	5
1.6	One-line summary (very exam friendly)	5
2	Feistel Cipher Structure	5
2.1	Why do we need block cipher structures?	5
2.2	Feistel network idea (the main trick)	6
2.3	Feistel diagram (data flow)	7
2.4	What the diagram is showing	7
2.5	One Feistel round (the mathematics behind each round)	8
2.6	Why decryption is possible even if F is NOT invertible (core Feistel property)	9
2.7	Decryption process (prove mathematically with the same structure)	9
2.8	Why the decryption diagram starts with swapped halves	10
2.9	Round-by-round explanation	10
2.10	Algorithm	10
2.11	Most important exam lines (15 marks conclusion)	11
2.12	Why Feistel is reversible (key point for exams)	11
2.13	Design ingredients of the round function F	11
2.14	Feistel vs non-Feistel block cipher	12
3	Substitution-Permutation Network (SPN)	12
3.1	Core idea	12
3.2	SPN diagram (simple)	12
3.3	Feistel vs SPN (exam comparison)	12
4	DES (Data Encryption Standard)	13
4.1	DES overview (Stallings-style summary)	13
4.2	DES high-level structure	13
5	General Depiction of DES Encryption Process (Short Notes)	13
5.1	Block input and initial permutation	13
5.2	Round processing (16 rounds)	13
5.3	Key schedule connection shown in the figure	14
5.4	Final swap and inverse permutation	14
5.5	Why expansion and S-boxes are used	14
6	DES Key Generation Process (Round Key Generator)	15

7	Single Round of DES Algorithm (10 Marks Notes with Diagram Explanation)	17
7.1	Overview of what happens in one DES round	17
7.2	Round equations (core mathematics)	18
7.3	Round function F in detail (step-by-step as shown in the diagram)	18
7.4	How the diagram produces L_i and R_i	19
7.5	Round key K_i generation link shown on the right side	19
7.6	Why a single round still keeps invertibility	20
8	Double DES and Triple DES	20
8.1	Motivation: why not single DES today?	20
8.2	Double DES (2DES)	20
8.3	Meet-in-the-middle attack (why 2DES is not strong enough)	20
8.4	Triple DES (3DES)	20
8.4.1	EDE mode (most common)	21
8.4.2	Keying options	21
9	Finite Fields Background for Cryptography	21
9.1	Why algebra (groups/rings/fields) matters in cryptography	21
9.2	Groups	21
9.3	Rings	22
9.4	Fields	22
10	Modular Arithmetic (Core for Cryptography)	22
10.1	Congruence and modulo	22
10.2	Arithmetic rules (important)	23
10.3	Modular inverse	23
11	Euclidean Algorithm and Extended Euclidean Algorithm	23
11.1	Euclidean Algorithm (GCD)	23
11.2	Extended Euclidean Algorithm (EEA)	23
12	Galois Fields: $GF(p)$ and $GF(2^n)$	24
12.1	$GF(p)$: finite field of prime order	24
12.2	$GF(2^n)$: binary extension fields (used in AES)	25
12.3	Polynomial representation	25
12.4	Addition in $GF(2^n)$ (XOR)	25
12.5	Multiplication in $GF(2^n)$ (polynomial + modular reduction)	25
12.6	Polynomial arithmetic over $GF(2)$: key rules	26
13	Quick Revision Tables	27
13.1	Feistel vs SPN (short table)	27
13.2	DES / 2DES / 3DES (short table)	27
14	Practice Questions (Exam Style)	27
15	References (Stallings-based)	28

1 Confusion and Diffusion (Important Concepts)

1.1 Meaning and purpose (core idea)

In modern cryptography, two fundamental design goals are **confusion** and **diffusion**. These terms were introduced by Claude Shannon and are emphasized in William Stallings to explain how block ciphers achieve security by hiding statistical structure.

Classroom Note

Easy classroom statement:

Confusion hides the relationship between **ciphertext** and **key**.

Diffusion hides the relationship between **ciphertext** and **plaintext**.

1.2 Confusion

Confusion means: even if an attacker knows how the system works, the relationship between the **key** and the **ciphertext** should be highly complex and unclear. This is usually achieved using **substitution** (S-boxes or nonlinear mapping).

1.2.1 Effect of changing a key bit (avalanche idea)

In a good cipher, if we change only **one bit of the secret key**, then **many or most bits of the ciphertext** should change. This makes it very hard for an attacker to guess how the key affects the output.

Exam-Focused Points (Write in Answers)

Write in exam:

In confusion, a small change in **key** causes a large unpredictable change in **ciphertext**. This masks the ciphertext–key relationship.

1.3 Diffusion

Diffusion means: the influence of a single plaintext bit should spread over many ciphertext bits. In other words, the ciphertext should not reveal patterns or redundancy of the plaintext. This is usually achieved using **permutation** and **mixing** operations (like P-box, shifting, linear mixing).

1.3.1 Effect of changing a plaintext bit

In a good cipher, if we change only **one bit of the plaintext**, then **many or most bits of the ciphertext** should change. This helps remove statistical patterns and redundancy from the plaintext.

Exam-Focused Points (Write in Answers)

Write in exam:

In diffusion, a small change in **plaintext** causes many bits of **ciphertext** to change. This masks the ciphertext–plaintext relationship.

1.4 “Vagueness” and “Redundancy” interpretation (as in your note)

Sometimes students explain confusion and diffusion using simple words:

- **Confusion** increases **vagueness/complexity** of how the key affects ciphertext.
- **Diffusion** spreads plaintext information so that plaintext **redundancy/patterns** are not visible in ciphertext.

Classroom Note

Important correction for classroom:

Diffusion does *not* “increase redundancy” in ciphertext.

Rather, diffusion **spreads** the plaintext redundancy so that it becomes **hidden** and does not appear as obvious patterns.

1.5 Stream cipher vs Block cipher (confusion and diffusion)

- **Stream ciphers:** mainly aim for **confusion** (key should unpredictably affect the keystream and ciphertext).
- **Block ciphers:** aim for both **confusion and diffusion** through repeated rounds of substitution and permutation/mixing.

Exam-Focused Points (Write in Answers)

Short 5-mark statement:

Both stream and block ciphers use **confusion**.

Block ciphers (especially SPN/Feistel designs) are designed to strongly provide **diffusion** as well.

1.6 One-line summary (very exam friendly)

Confusion: hides ciphertext-key relationship (mainly substitution/nonlinearity).

Diffusion: hides ciphertext-plaintext relationship by spreading plaintext bits (mainly permutation/mixing).

Avalanche: 1-bit change in key or plaintext should change many ciphertext bits.

2 Feistel Cipher Structure

2.1 Why do we need block cipher structures?

A modern block cipher must achieve two core security goals described in classic cryptography:

- **Confusion:** hide the relationship between the ciphertext and the key.
- **Diffusion:** spread the influence of each plaintext bit over many ciphertext bits.

In William Stallings' explanation, these properties are achieved through **repeated rounds** that combine:

- **substitution** (nonlinear operations, often using S-boxes),
- **permutation/linear mixing** (bit shuffles, XOR mixing),
- and **key mixing** (XOR with round keys).

2.2 Feistel network idea (the main trick)

The Feistel structure is extremely important because it allows **encryption and decryption using the same round structure** (only subkeys are used in reverse order). This was one of the most practical advantages behind DES.

Definition 2.1 (Feistel Round). *Let the input block be split into two halves:*

$$L_{i-1}, R_{i-1}$$

A Feistel round produces:

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

where F is the round function and K_i is the round key.

Classroom Note

Classroom explanation:

“In each round, the right half is processed by a function F and then mixed (XOR) with the left half. After that, we swap halves. Because of the swap and XOR, we can reverse the process easily during decryption.”

2.3 Feistel diagram (data flow)

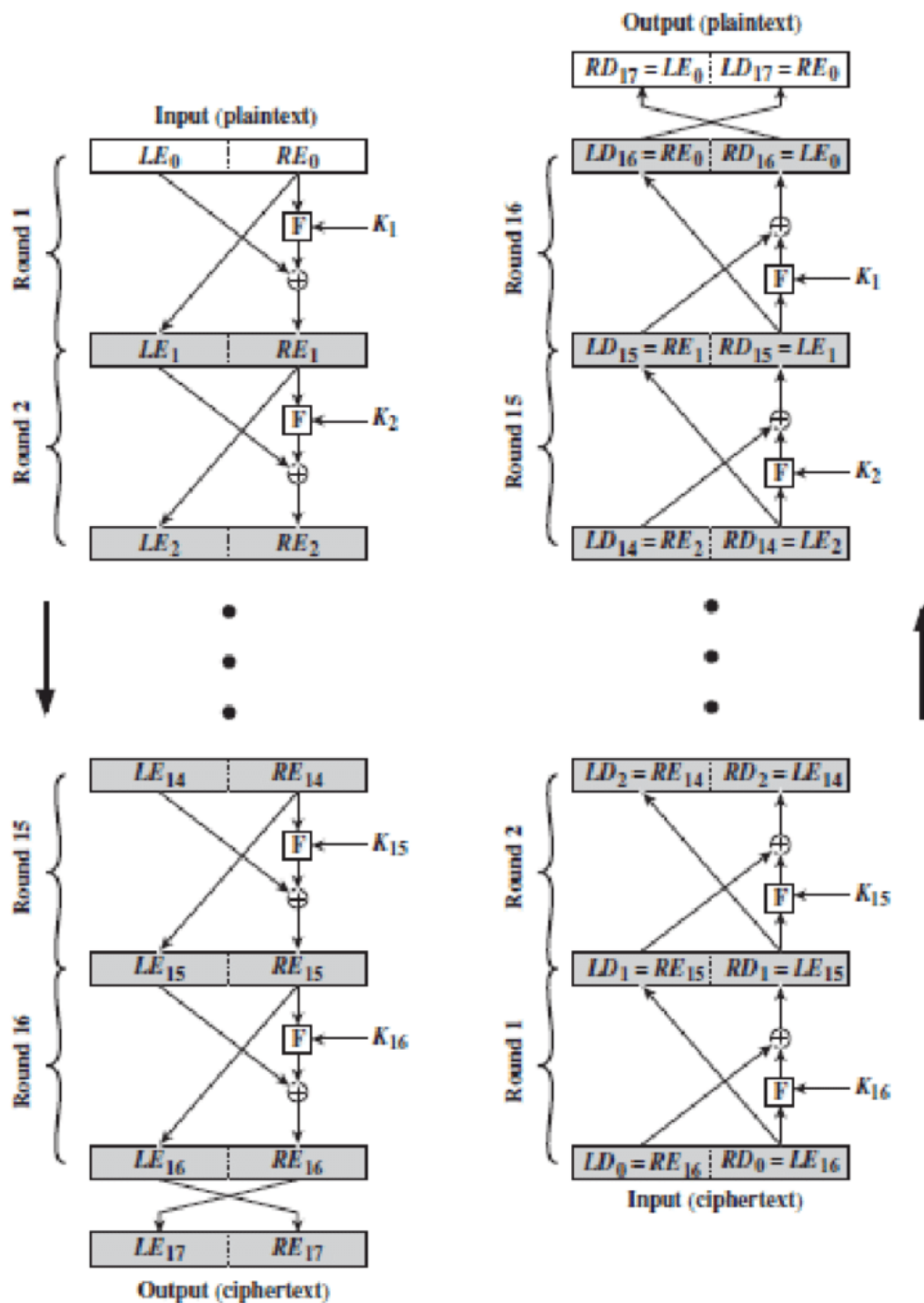


Figure 1: Feistel Cipher Encryption and Decryption

2.4 What the diagram is showing

Figure-1 shows the same Feistel structure used for both:

- **Encryption (left side):** plaintext \rightarrow ciphertext using round keys K_1, K_2, \dots, K_{16}
- **Decryption (right side):** ciphertext \rightarrow plaintext using the **same round function** but keys in reverse order $K_{16}, K_{15}, \dots, K_1$

The block is split into two halves at every stage:

$$\text{Input block} = L_0 \parallel R_0$$

In the figure, encryption halves are labeled as:

$$LE_0, RE_0, LE_1, RE_1, \dots, LE_{16}, RE_{16}$$

and decryption halves are labeled as:

$$LD_0, RD_0, LD_1, RD_1, \dots, LD_{16}, RD_{16}$$

2.5 One Feistel round (the mathematics behind each round)

A Feistel round takes the previous halves (L_{i-1}, R_{i-1}) and produces (L_i, R_i) using a round function F and a round key K_i .

Encryption round equations

For round $i = 1, 2, \dots, 16$:

$$L_i = R_{i-1} \tag{1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i) \tag{2}$$

Where:

- \oplus denotes XOR (bitwise addition mod 2)
- $F(\cdot)$ is the round function (it may include expansion, substitution (S-box), permutation, etc., depending on the cipher)
- K_i is the round subkey for round i

How the figure matches these equations

In the left side (encryption):

- The **crossed lines** show the swap: $L_i \leftarrow R_{i-1}$
- The small \oplus circle shows: $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
- The F box uses the right half and round key K_i

2.6 Why decryption is possible even if F is NOT invertible (core Feistel property)

Feistel design is powerful because the round function F does not need to be invertible. Decryption works because XOR is reversible:

$$A \oplus B \oplus B = A$$

From the encryption equations:

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

If we know R_i and also know $F(R_{i-1}, K_i)$, then we can recover L_{i-1} by XOR again:

$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$$

Also, from $L_i = R_{i-1}$ we directly get:

$$R_{i-1} = L_i$$

2.7 Decryption process (prove mathematically with the same structure)

Assume after encryption we have (L_i, R_i) . We want to recover (L_{i-1}, R_{i-1}) .

From (1):

$$R_{i-1} = L_i$$

From (2):

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

Substitute $R_{i-1} = L_i$:

$$R_i = L_{i-1} \oplus F(L_i, K_i)$$

Now XOR both sides with $F(L_i, K_i)$:

$$R_i \oplus F(L_i, K_i) = L_{i-1}$$

Therefore, the **decryption equations** are:

$$R_{i-1} = L_i \tag{3}$$

$$L_{i-1} = R_i \oplus F(L_i, K_i) \tag{4}$$

Key observation (matches the right side of the figure)

To decrypt the full cipher:

- we apply the same round structure,
- but we feed keys in reverse order: $K_{16}, K_{15}, \dots, K_1$.

That is exactly why on the right side the first decryption round uses K_{16} , then K_{15} , and so on.

2.8 Why the decryption diagram starts with swapped halves

In many Feistel ciphers (including DES), after the last round there is a **final swap** before output. That is why the encryption output is drawn as a swapped pair.

So, if encryption ends at (LE_{16}, RE_{16}) , the ciphertext output is commonly:

$$C = RE_{16} \parallel LE_{16}$$

Now look at the right side (decryption input). It shows:

$$LD_0 = RE_{16}, \quad RD_0 = LE_{16}$$

This is simply the ciphertext halves being assigned as the starting halves for decryption.

2.9 Round-by-round explanation

Encryption (left)

1. Start: (LE_0, RE_0) from plaintext split.

2. Round 1 with K_1 :

$$LE_1 = RE_0, \quad RE_1 = LE_0 \oplus F(RE_0, K_1)$$

3. Round 2 with K_2 :

$$LE_2 = RE_1, \quad RE_2 = LE_1 \oplus F(RE_1, K_2)$$

4. Continue similarly until round 16 with K_{16} .

5. Output (often with final swap): ciphertext $\approx RE_{16} \parallel LE_{16}$ (as shown).

Decryption (right)

1. Input ciphertext halves: $(LD_0, RD_0) = (RE_{16}, LE_{16})$.

2. Decryption round 1 uses K_{16} (reverse order):

$$LD_1 = RD_0, \quad RD_1 = LD_0 \oplus F(RD_0, K_{16})$$

3. Next uses K_{15} , and so on.

4. After 16 decryption rounds, we recover the original halves (LE_0, RE_0) (up to the final swap shown in the diagram).

2.10 Algorithm

Feistel Encryption

1. Input: plaintext block P , round keys K_1, \dots, K_{16} .

2. Split: $P = L_0 \parallel R_0$.

3. For $i = 1$ to 16:

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

4. Output (with final swap): $C = R_{16} \parallel L_{16}$.

Feistel Decryption

1. Input: ciphertext block C , round keys K_{16}, \dots, K_1 .
2. Split: $C = R_{16} \parallel L_{16}$ (so set $L_0 = R_{16}$, $R_0 = L_{16}$ as shown).
3. For $i = 1$ to 16 (keys reversed):

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus F(R_{i-1}, K_{17-i})$$

4. Output (with final swap): recover plaintext $P = R_{16} \parallel L_{16}$ (equivalently $L_0 \parallel R_0$).

2.11 Most important exam lines (15 marks conclusion)

- A Feistel cipher updates halves using: $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$.
- Decryption is possible without inverting F because XOR is reversible.
- Decryption uses the same structure with subkeys in reverse order: K_{16}, \dots, K_1 .
- The swaps drawn in the figure explain why ciphertext halves are fed reversed at the start of decryption.

2.12 Why Feistel is reversible (key point for exams)

Because XOR has the property:

$$A \oplus B \oplus B = A$$

So in decryption, we can recover L_{i-1} as:

$$L_{i-1} = R_i \oplus F(L_i, K_i)$$

and $R_{i-1} = L_i$ (from the swap).

Exam-Focused Points (Write in Answers)

Write these points for 10/15 marks:

- Feistel splits block into halves and uses XOR mixing with F .
- Decryption uses the **same structure** with keys in reverse order.
- XOR reversibility is the mathematical reason.
- Multiple rounds achieve confusion + diffusion.

2.13 Design ingredients of the round function F

Typical ingredients (as described in Stallings for DES-like ciphers):

- Expansion/permutation of bits (make right half larger or rearranged)
- XOR with round key
- Substitution through S-boxes (nonlinearity)
- Permutation/straight P-box (diffusion)

2.14 Feistel vs non-Feistel block cipher

Feistel is a **structure**. Many ciphers use other designs (e.g., SPN/AES). Feistel's key advantage is that invertibility does not require F to be invertible.

3 Substitution-Permutation Network (SPN)

3.1 Core idea

An SPN block cipher builds security by repeating:

- 1) **Substitution layer (S-boxes)** for confusion (nonlinearity)
- 2) **Permutation / linear diffusion layer** to spread bits (diffusion)
- 3) **Round key mixing** (usually XOR)

Definition 3.1 (SPN Round (generic form)). *A typical SPN round can be written as:*

$$\text{State} \leftarrow P(S(\text{State} \oplus K_i))$$

where S is substitution, P is permutation/linear mixing, and K_i is round key.

3.2 SPN diagram (simple)



3.3 Feistel vs SPN (exam comparison)

- **Feistel:** half-block swapping, XOR with F , decryption same structure.
- **SPN:** full-block transformations, substitution + permutation layers; inverse operations used in decryption.

Exam-Focused Points (Write in Answers)

Typical 5 marks: Differentiate Feistel and SPN

- Feistel does not require F to be invertible; SPN requires invertible layers for decryption.
- Feistel processes half-block each round; SPN processes full state each round.
- Both aim for confusion and diffusion by multiple rounds.

4 DES (Data Encryption Standard)

4.1 DES overview (Stallings-style summary)

DES is:

- a block cipher with block size 64 bits
- a Feistel cipher with 16 rounds
- uses an effective key length of 56 bits (64-bit input key includes 8 parity bits)

4.2 DES high-level structure

DES takes plaintext block M :

- 1) Apply **Initial Permutation (IP)**
- 2) Split into L_0 and R_0 (32 bits each)
- 3) Apply 16 Feistel rounds
- 4) Swap halves (after round 16)
- 5) Apply **Final Permutation (FP = IP⁻¹)**

5 General Depiction of DES Encryption Process (Short Notes)

5.1 Block input and initial permutation

DES is a 64-bit block cipher. It takes a 64-bit plaintext block as input and first applies the initial permutation (IP), which is a fixed bit-reordering operation. The output of IP is still 64 bits. After this permutation, the block is treated as two 32-bit halves, commonly denoted by L_0 and R_0 .

5.2 Round processing (16 rounds)

DES performs 16 Feistel rounds. In each round i , a 48-bit round key K_i is used. The round update follows the Feistel equations:

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

Here, F is the DES round function that mixes the right half with the round key using expansion, XOR, substitution (S-boxes), and permutation. After each round, the output remains 64 bits in total (two 32-bit halves).

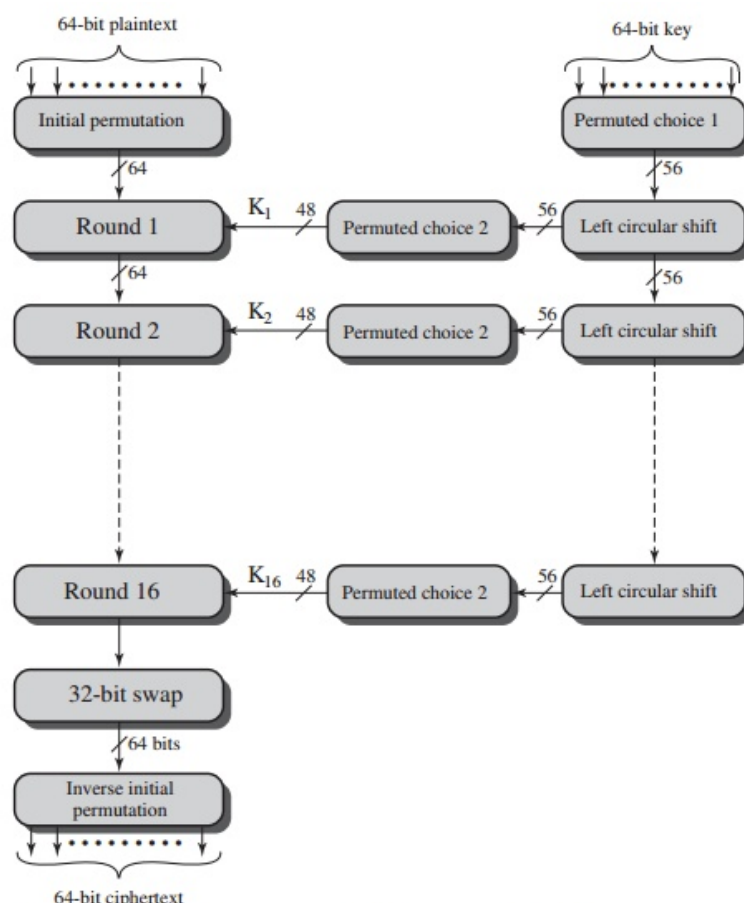


Figure 3.5 General Depiction of DES Encryption Algorithm

Figure 2: Des general structure

5.3 Key schedule connection shown in the figure

The figure also shows how the 48-bit round keys are generated from the 64-bit key. The 64-bit key first passes through permuted choice 1 (PC-1), producing 56 bits (parity bits removed and bits permuted). Then, for each round, the 56-bit value is circularly left shifted (on two 28-bit halves), and permuted choice 2 (PC-2) compresses and permutes it to produce a 48-bit round key K_i . This process repeats to generate K_1 through K_{16} .

5.4 Final swap and inverse permutation

After round 16, DES performs a 32-bit swap of the two halves, so the final pair becomes (R_{16}, L_{16}) . Then the inverse initial permutation (IP^{-1}) is applied to produce the final 64-bit ciphertext block. Decryption uses the same structure but applies the round keys in reverse order from K_{16} down to K_1 .

5.5 Why expansion and S-boxes are used

- Expansion makes 32-bit input match 48-bit key mixing.
- Repeated bits in expansion help diffusion and ensure boundary bits influence multiple S-boxes.

- S-boxes introduce **nonlinearity** (confusion); they are the main defense against linear/differential cryptanalysis (as discussed in modern analysis).

6 DES Key Generation Process (Round Key Generator)

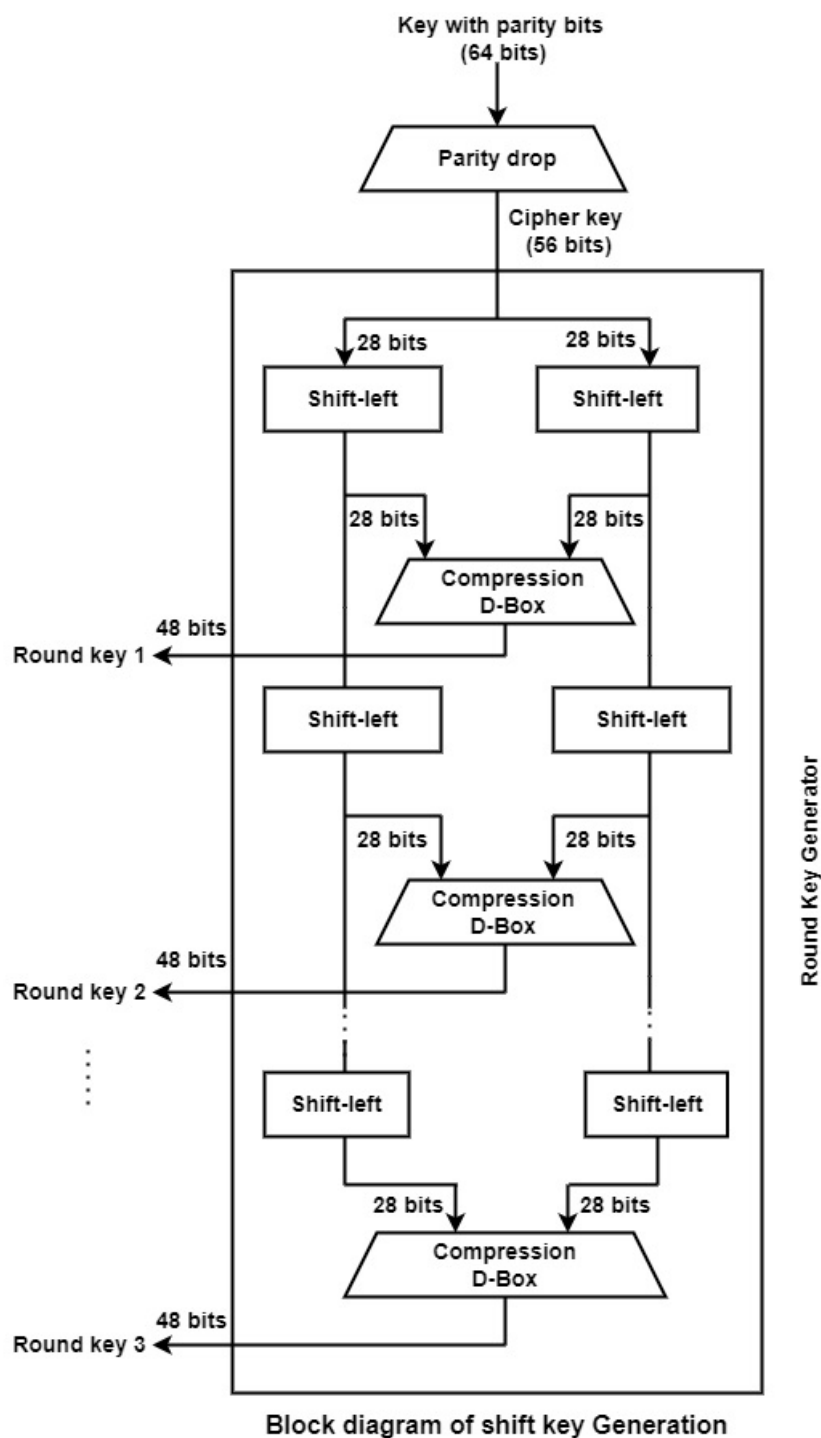


Figure 3: DES KEY generation process

Data Encryption Standard (DES) is a **block cipher** that operates on **64-bit blocks** of data. In encryption, DES takes a **64-bit plaintext block** and transforms it into a **64-bit ciphertext block**. In decryption, DES takes the **64-bit ciphertext block** and converts it back into the original **64-bit plaintext block**. DES is a **symmetric key algorithm**, which means the **same secret key** is used for both encryption and decryption. Although a DES key is given as **64 bits**, only **56 bits** are actually used as the effective cipher key, while the remaining **8 bits are parity bits**. In DES, the overall algorithm structure is fixed and publicly known; the security comes from keeping the **actual key** secret between the sender and the receiver.

DES uses **16 rounds** of processing, and an important feature of DES is that it does not use the same key bits directly in every round. Instead, it generates **sixteen different round keys**, one for each round. Each round key is **48 bits long**. Therefore, the purpose of the DES key generation process (also called the **key schedule** or **round key generator**) is to produce sixteen **48-bit subkeys**:

$$K_1, K_2, K_3, \dots, K_{16}$$

from the single original **56-bit effective key**. These round keys are used inside the DES round function to provide **confusion** and to make the ciphertext strongly dependent on the secret key.

The key generation process begins with the 64-bit key with parity bits. The 64-bit input key contains 8 parity bits placed at positions 8, 16, 24, 32, 40, 48, 56, and 64. These parity bits are discarded because they are not used as real key bits. This first stage is called parity drop. In the parity drop stage, DES applies a fixed permutation (commonly called Permuted Choice 1 or PC-1), which not only removes the parity bits but also reorders the remaining bits according to a predefined table. As a result of this parity drop and permutation, we obtain a 56-bit cipher key:

$$K^+ = PC1(K)$$

This 56-bit key K^+ is the main input to the round key generator, as shown in Figure ??.

After obtaining the 56-bit key, DES divides it into two equal halves of 28 bits each. The left half is denoted by C_0 and the right half is denoted by D_0 :

$$K^+ = C_0 \parallel D_0$$

where C_0 contains the first 28 bits and D_0 contains the last 28 bits. The reason DES splits the key into two halves is because in every round, both halves will be shifted (rotated) and then combined again. This repeated shifting ensures that different sets of key bits appear in different rounds, so the round keys are not identical and not trivially related.

In each round i (from 1 to 16), DES performs a circular left shift on both halves C_{i-1} and D_{i-1} to obtain C_i and D_i . This operation is shown as Shift-left blocks in Figure ??. A circular left shift means the leftmost bit that goes out from the left end is wrapped around and inserted back at the right end, so no key bit is lost. Mathematically, for each round:

$$C_i = LS_i(C_{i-1}), \quad D_i = LS_i(D_{i-1})$$

where LS_i indicates the number of left shifts in round i . DES uses a specific shift schedule. In rounds 1, 2, 9, and 16 the shift is 1 bit, and in all other rounds the shift is 2 bits. Therefore, the shift schedule is:

$$\{1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1\}$$

This shifting schedule is chosen so that the key bits move through different positions across the 16 rounds, producing strong variation among round keys and avoiding simple repetitive patterns.

After shifting, the two 28-bit halves are **combined** again to form a 56-bit intermediate value:

$$C_i \parallel D_i$$

However, DES round keys must be **48 bits**, not 56 bits. So DES applies another fixed permutation and selection step, known as **Permuted Choice 2 (PC-2)** or the **compression D-box**. This compression stage is shown as the **Compression D-Box** blocks in Figure ???. The compression D-box takes the 56 combined bits and **selects 48 of them** (and also permutes them) to produce the round key:

$$K_i = PC2(C_i \parallel D_i)$$

Thus, each round key K_i is exactly **48 bits** and is used only in that particular round of DES.

By repeating the same procedure for all 16 rounds, the round key generator produces sixteen unique keys:

$$K_1, K_2, K_3, \dots, K_{16}$$

These keys are then used in the DES encryption rounds in the order $K_1 \rightarrow K_{16}$. During decryption, DES uses the same round function but applies the subkeys in the reverse order $K_{16} \rightarrow K_1$. This is possible because DES is based on a Feistel structure, and the Feistel design allows decryption to be done by reversing the sequence of round keys.

Historically, DES originated from IBM's research in the late 1960s, where IBM developed a cipher known as LUCIFER. In the 1970s, the design was modified and standardized into DES with several innovations, including the 16-round structure, the key schedule (parity drop, shifts, compression), and carefully designed substitution boxes. Over time, because computing power increased, the effective 56-bit key length became vulnerable to brute-force attacks. Therefore, modern improvements and replacements were introduced, such as using longer keys (for example, algorithms with 128-bit keys) and using multi-pass designs such as Triple DES (3DES), which applies DES multiple times with multiple keys to increase security.

7 Single Round of DES Algorithm (10 Marks Notes with Diagram Explanation)

7.1 Overview of what happens in one DES round

DES is a 16-round Feistel block cipher. In each round, the 64-bit data block is split into two 32-bit halves. The left half and right half before round i are denoted by L_{i-1} and R_{i-1} . The output halves after round i are L_i and R_i . The main operations in one round are swapping, applying the round function F to the right half with the round key K_i , and XOR mixing with the left half. The diagram in Figure 4 shows both the data path (left side) and the key schedule portion that produces K_i (right side).

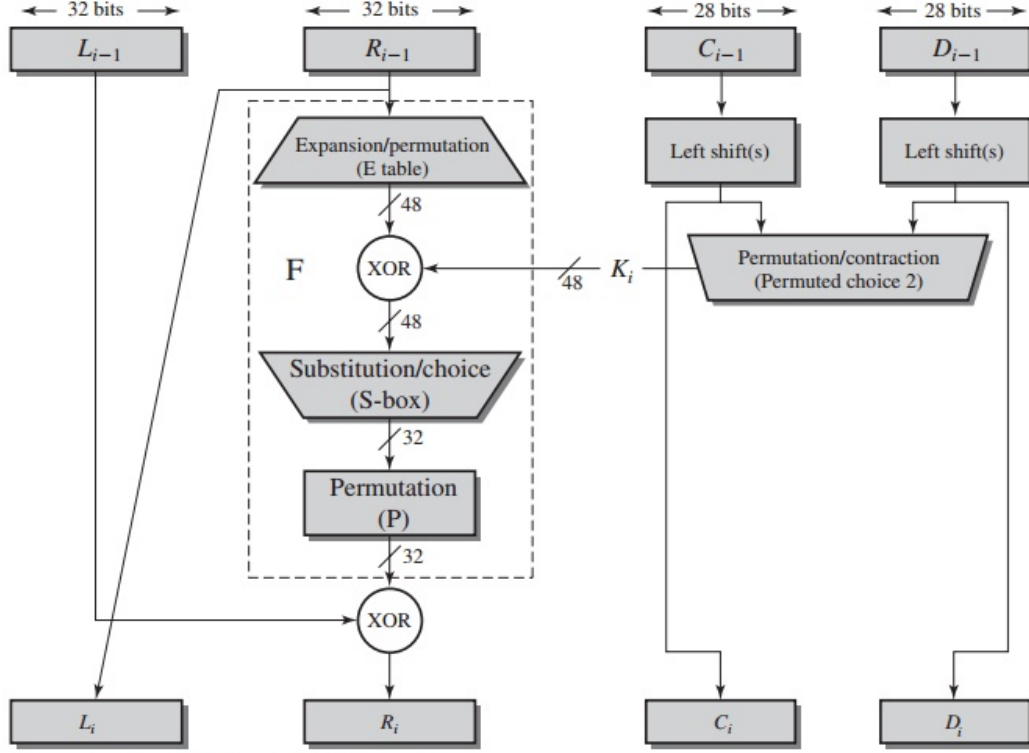


Figure 3.6 Single Round of DES Algorithm

Figure 4: Single round of DES: round function F and round-key generation link.

7.2 Round equations (core mathematics)

A single DES round follows the Feistel equations:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

The first equation means the next left half is simply the previous right half. The second equation means the next right half is formed by XORing the previous left half with the output of the round function F .

7.3 Round function F in detail (step-by-step as shown in the diagram)

The round function F takes two inputs: a 32-bit value R_{i-1} and a 48-bit round key K_i . It produces a 32-bit output. The function contains four main stages: expansion/permutation, XOR with the round key, substitution using S-boxes, and permutation using P-box.

First, R_{i-1} (32 bits) is passed through the expansion/permutation E table. This expansion converts 32 bits into 48 bits by reordering bits and repeating some boundary bits. The main reason for expansion is to match the 48-bit round key length and to ensure diffusion by letting some input bits influence multiple S-box inputs:

$$E(R_{i-1}) : 32 \rightarrow 48$$

Second, the expanded 48-bit result is XORed with the 48-bit round key K_i :

$$B = E(R_{i-1}) \oplus K_i$$

The output B is still 48 bits.

Third, the 48-bit value B is divided into eight 6-bit blocks:

$$B = B_1 \parallel B_2 \parallel \cdots \parallel B_8$$

Each B_j (6 bits) is applied to one S-box S_j . Each S-box maps 6 bits to 4 bits, so after all eight S-box substitutions, the output becomes 32 bits:

$$S(B) : 48 \rightarrow 32$$

In DES, S-boxes provide nonlinearity and are the main source of confusion, meaning they make the relationship between key bits and ciphertext bits complex.

Fourth, the 32-bit output from the S-box layer is permuted using the P permutation (also called the straight permutation). This rearranges bits to spread the influence of each S-box output across positions that will be used in different S-boxes in the next round, supporting diffusion:

$$P(\cdot) : 32 \rightarrow 32$$

Combining these stages, the DES round function is written compactly as:

$$F(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$$

7.4 How the diagram produces L_i and R_i

Figure 4 shows that R_{i-1} enters the dashed box (the F function). The output of F is a 32-bit value that is XORed with L_{i-1} . The XOR circle near the bottom of the dashed box represents:

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

The line from R_{i-1} to L_i represents the Feistel swap:

$$L_i = R_{i-1}$$

Therefore the output of the round is the pair (L_i, R_i) .

7.5 Round key K_i generation link shown on the right side

The diagram also shows the key schedule elements for generating the round key K_i . The 56-bit key state is split into two 28-bit halves, denoted by C_{i-1} and D_{i-1} . Each round performs left circular shift(s) on both halves to obtain C_i and D_i . Then the permuted choice 2 block (permutation/contraction) selects and permutes 48 bits out of the combined 56 bits:

$$K_i = PC2(C_i \parallel D_i)$$

This 48-bit key K_i is fed into the XOR inside the F function.

7.6 Why a single round still keeps invertibility

Even if F is complex and not invertible, the whole round is invertible due to the Feistel design. XOR is reversible and the swap is reversible, so decryption can be done by running the same round structure with keys in reverse order. This is why DES can decrypt using the same internal design.

8 Double DES and Triple DES

8.1 Motivation: why not single DES today?

DES key length is 56 bits. Brute-force search became feasible with advances in computing power. So stronger variants were introduced.

8.2 Double DES (2DES)

Double DES means encrypt twice with two keys:

$$C = E_{K_2}(E_{K_1}(P))$$

Naively, one might think security becomes $2^{56} \times 2^{56} = 2^{112}$, but it does not.

8.3 Meet-in-the-middle attack (why 2DES is not strong enough)

Given P and C :

$$X = E_{K_1}(P), \quad C = E_{K_2}(X)$$

Attack idea:

- Compute $E_{K_1}(P)$ for all 2^{56} keys and store results.
- Compute $D_{K_2}(C)$ for all 2^{56} keys and look for matches.

This reduces complexity to about:

$$\approx 2^{57} \text{ operations (time) and } 2^{56} \text{ storage}$$

So 2DES is not recommended.

Exam-Focused Points (Write in Answers)

Write for 10 marks: 2DES is vulnerable due to meet-in-the-middle, giving about 2^{57} complexity, not 2^{112} .

8.4 Triple DES (3DES)

3DES was designed to strengthen DES while keeping DES hardware compatibility.

8.4.1 EDE mode (most common)

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$$

Why EDE?

- If $K_1 = K_2 = K_3$, then 3DES becomes single DES (backward compatibility).
- Decryption is symmetric:

$$P = D_{K_1}(E_{K_2}(D_{K_3}(C)))$$

8.4.2 Keying options

- **3-key 3DES:** K_1, K_2, K_3 independent (strongest)
- **2-key 3DES:** $K_1 = K_3$, K_2 independent (common in practice historically)

Classroom Note

Classroom explanation:

“3DES is like applying DES three times in a smart way so that old DES systems can still interoperate, but security becomes much stronger than single DES.”

9 Finite Fields Background for Cryptography

9.1 Why algebra (groups/rings/fields) matters in cryptography

Many cryptographic systems depend on arithmetic structures where operations behave predictably:

- RSA uses modular arithmetic over integers.
- AES uses arithmetic in $GF(2^8)$ (a finite field).
- ECC uses groups over finite fields.

So we must understand **groups, rings, and fields**.

9.2 Groups

Definition 9.1 (Group). A set G with a binary operation \circ is a group if:

- Closure:** $a, b \in G \Rightarrow a \circ b \in G$
- Associativity:** $(a \circ b) \circ c = a \circ (b \circ c)$
- Identity:** $\exists e \in G$ s.t. $a \circ e = e \circ a = a$
- Inverse:** $\forall a \in G, \exists a^{-1} \in G$ s.t. $a \circ a^{-1} = e$

If also $a \circ b = b \circ a$, then group is **abelian (commutative)**.

Worked Example

Example: $(\mathbb{Z}, +)$ is an abelian group.
Identity is 0, inverse of a is $-a$.

9.3 Rings

Definition 9.2 (Ring). A set R with two operations $(+, \cdot)$ is a ring if:

- $(R, +)$ is an abelian group.
- Multiplication \cdot is associative.
- Distributive laws hold:

$$a \cdot (b + c) = a \cdot b + a \cdot c, \quad (a + b) \cdot c = a \cdot c + b \cdot c$$

Worked Example

\mathbb{Z} with $(+, \times)$ is a ring.

9.4 Fields

Definition 9.3 (Field). A ring $(F, +, \cdot)$ is a field if:

- $(F, +)$ is an abelian group
- $(F \setminus \{0\}, \cdot)$ is an abelian group

So every nonzero element has a multiplicative inverse.

Worked Example

Examples: \mathbb{Q} , \mathbb{R} , and finite fields like $GF(p)$ where p is prime.

Exam-Focused Points (Write in Answers)**One-line memory:**

Group = 1 operation + inverses; Ring = 2 operations (add group + multiply associative); Field = ring + multiplicative inverses for all nonzero.

10 Modular Arithmetic (Core for Cryptography)**10.1 Congruence and modulo**

We say:

$$a \equiv b \pmod{n}$$

if n divides $(a - b)$.

10.2 Arithmetic rules (important)

If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$ then:

$$a + c \equiv b + d \pmod{n}$$

$$a \cdot c \equiv b \cdot d \pmod{n}$$

10.3 Modular inverse

An integer a has an inverse modulo n if:

$$\exists a^{-1} \text{ such that } a \cdot a^{-1} \equiv 1 \pmod{n}$$

This inverse exists **iff**:

$$\gcd(a, n) = 1$$

11 Euclidean Algorithm and Extended Euclidean Algorithm

11.1 Euclidean Algorithm (GCD)

Theorem 11.1. For integers $a > b > 0$:

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

Worked Example

Find $\gcd(232, 124)$:

$$232 = 124 \cdot 1 + 108$$

$$124 = 108 \cdot 1 + 16$$

$$108 = 16 \cdot 6 + 12$$

$$16 = 12 \cdot 1 + 4$$

$$12 = 4 \cdot 3 + 0$$

So $\gcd(232, 124) = 4$.

11.2 Extended Euclidean Algorithm (EEA)

EEA finds integers x, y such that:

$$\gcd(a, b) = ax + by$$

This is the foundation for finding modular inverse.

Worked Example

Find inverse of 17 modulo 43. We want $17x \equiv 1 \pmod{43}$.

Euclid:

$$43 = 17 \cdot 2 + 9$$

$$17 = 9 \cdot 1 + 8$$

$$9 = 8 \cdot 1 + 1$$

$$8 = 1 \cdot 8 + 0$$

So gcd is 1.

Back-substitute:

$$1 = 9 - 8 \cdot 1$$

But $8 = 17 - 9 \cdot 1$, so:

$$1 = 9 - (17 - 9) = 2 \cdot 9 - 17$$

And $9 = 43 - 17 \cdot 2$:

$$1 = 2(43 - 17 \cdot 2) - 17 = 2 \cdot 43 - 5 \cdot 17$$

Thus:

$$-5 \cdot 17 \equiv 1 \pmod{43}$$

So:

$$17^{-1} \equiv -5 \equiv 38 \pmod{43}$$

Answer: inverse is 38.

Exam-Focused Points (Write in Answers)

Very important for exams:

- Inverse of $a \bmod n$ exists only if $\gcd(a, n) = 1$.
- Use Extended Euclid to express $1 = ax + ny$.
- Then $x \pmod{n}$ is the inverse of a .

12 Galois Fields: $GF(p)$ and $GF(2^n)$ **12.1 $GF(p)$: finite field of prime order**

If p is prime, then:

$$GF(p) = \{0, 1, 2, \dots, p-1\}$$

with addition and multiplication performed modulo p .

Classroom Note

Key reason p must be prime:

If p is prime, every nonzero element has a multiplicative inverse modulo p , so it

forms a field.

Worked Example

In $GF(7)$, find inverse of 3: We need $3x \equiv 1 \pmod{7}$. Try: $3 \cdot 5 = 15 \equiv 1 \pmod{7}$, so $3^{-1} = 5$ in $GF(7)$.

12.2 $GF(2^n)$: binary extension fields (used in AES)

In $GF(2^n)$:

- Elements are polynomials of degree $< n$ with coefficients in $\{0, 1\}$.
- Addition is XOR of coefficients.
- Multiplication is polynomial multiplication modulo an **irreducible polynomial** of degree n .

12.3 Polynomial representation

An element in $GF(2^n)$ looks like:

$$a(x) = a_{n-1}x^{n-1} + \cdots + a_1x + a_0, \quad a_i \in \{0, 1\}$$

Example in $GF(2^8)$:

$$a(x) = x^7 + x^3 + x + 1$$

This corresponds to the byte:

$$10001011_2$$

12.4 Addition in $GF(2^n)$ (XOR)

$$(a_7 \cdots a_0) + (b_7 \cdots b_0) = (a_7 \oplus b_7 \cdots a_0 \oplus b_0)$$

Worked Example

In $GF(2^8)$:

$$10110010_2 + 01100101_2 = 11010111_2$$

(just XOR).

12.5 Multiplication in $GF(2^n)$ (polynomial + modular reduction)

Steps:

- 1) Multiply as polynomials over $GF(2)$ (coefficients mod 2).
- 2) Reduce modulo an irreducible polynomial $m(x)$ of degree n .

12.6 Polynomial arithmetic over GF(2): key rules

Because coefficients are in $\{0, 1\}$:

$$1 + 1 = 0, \quad 1 + 0 = 1, \quad 0 + 0 = 0$$

So subtraction equals addition:

$$a(x) - b(x) = a(x) + b(x)$$

Worked Example

Let us work in $GF(2^4)$ with irreducible polynomial:

$$m(x) = x^4 + x + 1$$

Multiply:

$$a(x) = x^3 + x + 1 \quad (\text{binary } 1011)$$

$$b(x) = x^2 + 1 \quad (\text{binary } 0101)$$

Multiply:

$$a(x)b(x) = (x^3 + x + 1)(x^2 + 1) = x^5 + x^3 + x^3 + x + x^2 + 1$$

Combine like terms (mod 2):

$$x^5 + (x^3 + x^3) + x^2 + x + 1 = x^5 + x^2 + x + 1$$

Now reduce mod $m(x) = x^4 + x + 1$. Since:

$$x^4 \equiv x + 1 \pmod{m(x)}$$

Multiply both sides by x :

$$x^5 \equiv x(x + 1) = x^2 + x$$

So:

$$x^5 + x^2 + x + 1 \equiv (x^2 + x) + x^2 + x + 1 \equiv 1$$

Thus product is 1 in this field.

13 Quick Revision Tables

13.1 Feistel vs SPN (short table)

Feature	Feistel	SPN
Round input	Split: left/right halves	Full state/block
Main mixing	XOR with F and swap	Substitution + permutation layers
Invertibility	F need not be invertible	layers must be invertible for decryption
Decryption	Same structure, keys reversed	Uses inverse S/P layers + keys reversed
Example	DES	AES (SPN-like)

13.2 DES / 2DES / 3DES (short table)

Scheme	Keys	Important note
DES	56-bit effective	16-round Feistel, 64-bit block
Double DES	K_1, K_2	Meet-in-the-middle reduces strength (not 2^{112})
Triple DES (EDE)	K_1, K_2, K_3	Stronger, backward compatible when $K_1 = K_2 = K_3$

14 Practice Questions (Exam Style)

Very common 5 marks

1. Define Feistel cipher structure and write round equations.
2. Differentiate Feistel network and SPN.
3. Explain DES round function F (E, XOR, S-box, P).
4. What is meet-in-the-middle attack? Explain for Double DES.
5. Define group, ring, field with one example each.

Very common 10/15 marks

1. Explain DES structure with diagram and describe F function in detail.
2. Explain DES key schedule: PC-1, shifts, PC-2 and why circular shifts are used.
3. Explain $GF(p)$ and $GF(2^n)$ with examples of addition and multiplication.
4. Use Extended Euclidean Algorithm to find modular inverse and explain why it works.

15 References (Stallings-based)

References

- [1] William Stallings, *Cryptography and Network Security: Principles and Practice*, Pearson (latest available edition in your library/department).
- [2] NIST (historical standard), *Data Encryption Standard (DES)* / FIPS publications (DES details and tables).

