

Block Ciphers and DES (Feistel Cipher)

Prepared by Dr. Sumaya Alkubaisi

FNT625
Cybersecurity Concepts

Lecture 4



Lecture

4

Learning Goals

By the end of this lecture, you will be able to:

- Differentiate between stream ciphers and block ciphers.
- Explain Feistel structure, diffusion, and confusion.
- Understand the Data Encryption Standard (DES) algorithm.
- Evaluate DES strengths and weaknesses.
- Link classical block ciphers to modern systems (3DES, AES) and fintech applications.

Stream Ciphers

- A stream cipher encrypts data one bit or one byte at a time, instead of working on large blocks.
Examples: Autokeyed Vigenère cipher, Vernam cipher

One-time pad (ideal case):

If the keystream is truly random and as long as the plaintext, the cipher is unbreakable (theoretically perfect security).

Problem: Both users must already have the same random keystream, which is very hard to share securely if the data is large.

Practical solution:

- Instead of using a huge random keystream, both users share a short secret key.
- A keystream generator (an algorithm) uses this key to create the long cryptographic bit stream.
- The generator must be designed so it's impossible to predict future keystream bits from past ones.

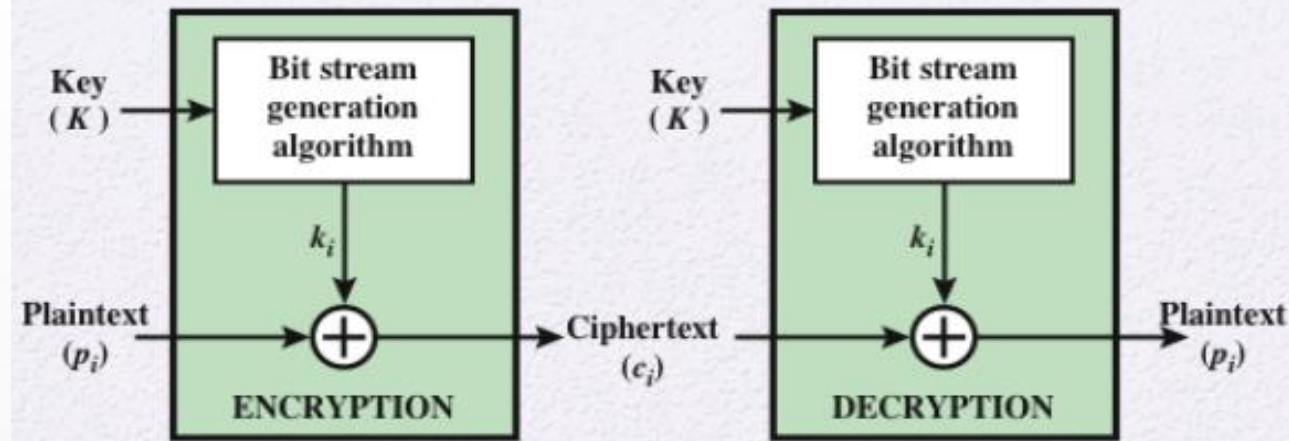
Block Cipher

- In a block cipher, the plaintext is divided into fixed-size chunks (blocks), such as 64 bits or 128 bits.
- Each block of plaintext is fully encrypted to produce a block of ciphertext of the same size.
- Like stream ciphers, both users need to share the same secret key.
- Most encryption systems used in networks today are based on block ciphers.

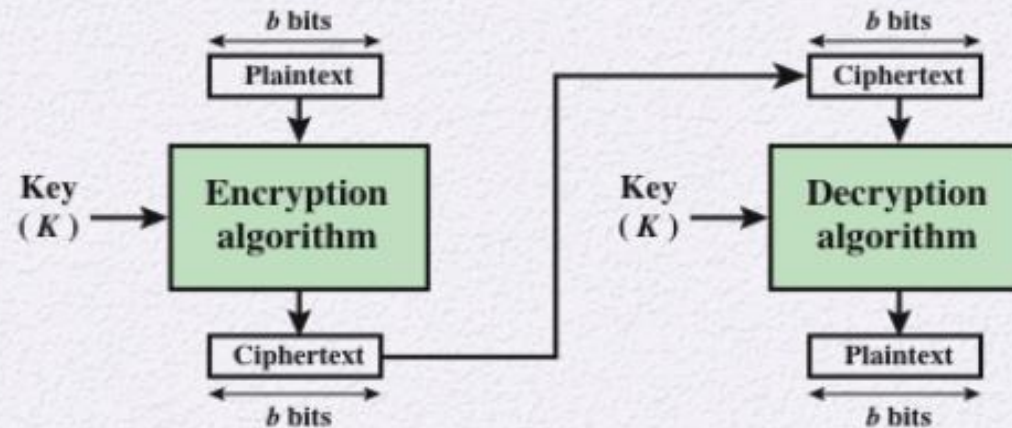
(a) Stream Cipher

- A key (K) is fed into a generator to create a keystream.
- Each plaintext bit/byte is combined with the keystream using XOR $\rightarrow$ produces ciphertext.
- Receiver uses the same key to regenerate the keystream and XOR again $\rightarrow$ gets the original plaintext.

Works bit by bit, good for real-time data (e.g., video, audio).



(a) Stream Cipher Using Algorithmic Bit Stream Generator



(b) Block Cipher

Figure 4.1 Stream Cipher and Block Cipher

(b) Block Cipher

Lecture

- Plaintext is divided into fixed-size blocks (e.g., 64 or 128 bits).
- Each block + key $\rightarrow$ encrypted into a ciphertext block of the same size.
- Receiver applies the decryption algorithm with the same key to get back the plaintext blocks.

Works on chunks/blocks, common in secure network encryption.

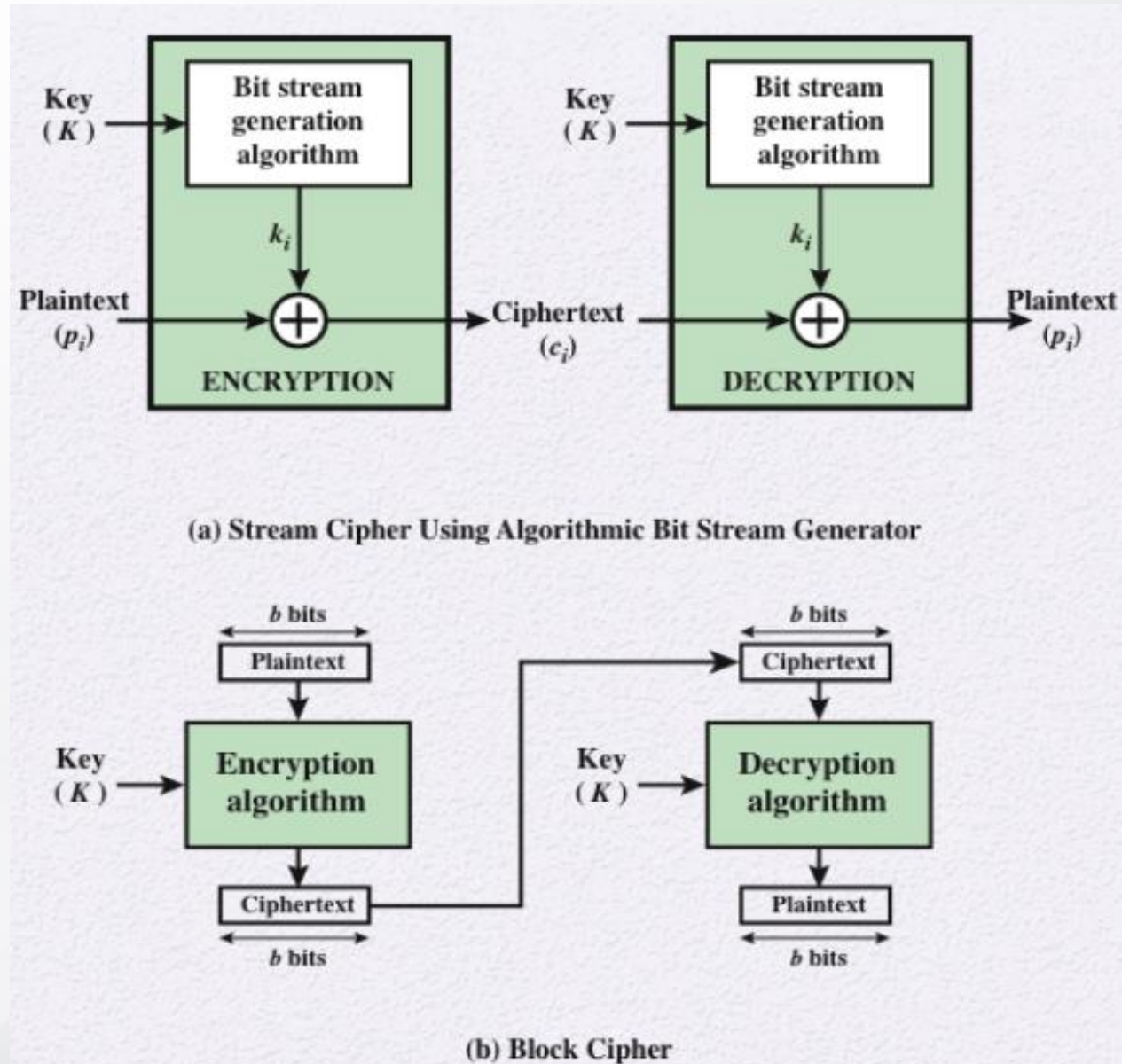


Figure 4.1 Stream Cipher and Block Cipher

Why Block Ciphers?

- Stronger Security through Design Block ciphers are generally preferred over stream ciphers because they are designed to provide stronger security.
- This increased strength comes from a greater ability to achieve the critical security properties of confusion and diffusion
- Standardization for Global Systems: A significant practical advantage of block ciphers is that they are much easier to standardize across vast, global systems.
- The fundamental principles of block ciphers remain the backbone of many modern financial security protocols

Feistel Cipher Structure

- Basis of Modern structure used by many significant symmetric block ciphers currently in use
- Feistel proposed the use of a cipher that alternates substitutions and permutations
- **Substitution** = replace each letter with another
- **Permutation (Transposition)**= rearranging the positions of characters or bits(reordering elements).
- Is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions

Shannon's Principles: Confusion and Diffusion

Modern block cipher design is fundamentally based on two principles outlined by Shannon

Confusion: This principle requires that the relationship between the final ciphertext (the encrypted message) and the key must be completely obscured.

Example: Substitution (replace each letter with another) hides patterns.

Diffusion: This principle demands that the statistics of the plaintext (the original message) must be spread across the entire ciphertext

Example: Transposition (shuffling positions) spreads out patterns.

Shannon's Principles: Confusion and Diffusion

His idea was to combine them (product cipher):

- First apply substitution (confusion).
- Then apply transposition (diffusion).
- Repeat this process multiple times.
- Application: Block ciphers like DES, AES are built exactly on this principle — they alternate substitution steps and permutation steps to make the cipher strong.
- Feistel ciphers achieve both through multiple rounds that mix data and key information.

Diffusion Example

- One-bit change in plaintext → many bits change in ciphertext.

Example:

Plaintext 1001 → Ciphertext A7C3

Change to 1000 → Ciphertext 4F99

Result: nearly all ciphertext bits flipped.

Prevents frequency analysis — attackers can't see patterns.

Lecture

4

Shannon's Principles: Confusion and Diffusion

Confusion Example

- Complex relation between key and ciphertext.
- Small change in key → drastically different ciphertext.

Example:

Key K1 → Ciphertext A7C3

Key K2 (1-bit different) → Ciphertext 8B1E

Result: Makes it impossible to predict ciphertext patterns or deduce the key mathematically.

The Feistel Structure

- a breakthrough in cipher design, was proposed by Horst Feistel at IBM in 1973. This design approach works by dividing the block of plaintext data into two halves.
- It then applies repeated rounds of alternating substitution and permutation operations

How a Feistel Round Works (The Inputs): A single round of a Feistel cipher takes two halves as input: the Left half (L) and the Right half (R). The key steps define the output for the next round

How a Feistel Round Works (The Calculation) : The calculation for the new halves is straightforward:

- The New Left half (L) simply becomes the old Right half (R).
- The New Right half (R) is calculated by taking the old Left half (L) and combining it using the XOR operation $\oplus$ with the result of a specialized round function F.

The function F takes the old Right half (R) and the Round Key (K) as its inputs: **New R = L $\oplus$ F(R, K)**.

- This process is repeated many times for multiple rounds till we get the cipher text

The Feistel Structure

How a Feistel Round Works (The Inputs in Decryption)

A single round of Feistel decryption also takes two halves as input: the Left half (L) and the Right half (R) from the ciphertext.

The key steps define how to get back the previous round's halves.

How a Feistel Round Works (The Calculation in Decryption)

The calculation to recover the old halves is straightforward:

- The Old Right half (R) simply becomes the current Left half (L).
- The Old Left half (L) is calculated by taking the current Right half (R) and combining it with the result of the round function F using XOR $\oplus$.

The round function F takes the current Left half (L) and the Round Key (K) as its inputs:

$$\text{Old L} = \text{Current R} \oplus F(\text{Current L}, K)$$

- This process is repeated in reverse order of the keys until the original plaintext halves are recovered.

Lecture 1

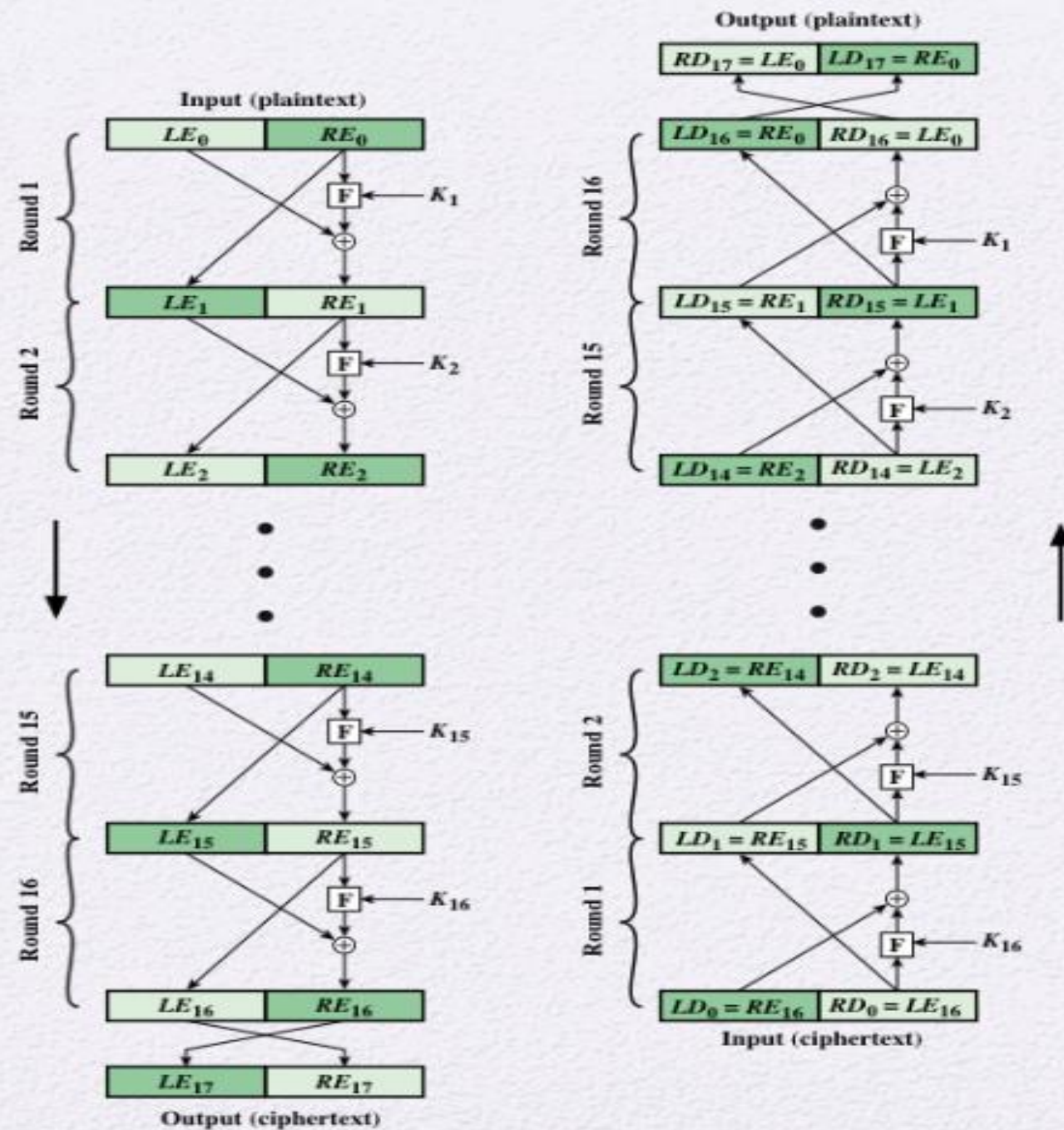


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

The Feistel Structure (Example)

The key idea:

Encryption moves forward: new left = old right, new right = old left $\oplus$ F(old right, key).

Decryption just reverses it: old right = new left, old left = new right $\oplus$ F(new left, key).

Encryption Example (1 Round)

Plaintext (5, 7) $\rightarrow$ halves: L0=5, R0=7

Key: K=3

Function F(x, K): just do (x+K)(x + K)(x+K). (Addition is chosen just for simplicity)

Steps:

1. L1=R0=7
2. R1=L0 $\oplus$ F(R0,K)
 $\rightarrow$ R1=5 $\oplus$ (7+3)
 $\rightarrow$ R1=5 $\oplus$ 10=15

Ciphertext: (L1,R1)=(7,15)

The Feistel Structure (Example)

Decryption Example (1 Round)

Ciphertext halves:

$L1=7, R1=15$

Key: $K=3K$

Steps:

$R0=L1=7$

$L0=R1 \oplus F(L1, K)$

$\rightarrow L0=15 \oplus (7+3)$

$\rightarrow L0=15 \oplus 10=5$

Recovered Plaintext: $(L0, R0)=(5, 7)$

Why Feistel Works

- Encryption and decryption share the same structure.
- Only the key order is reversed during decryption.
- This symmetry makes implementation efficient and error-free.
- Each round increases security, even if the round function alone is not strong.
- Feistel's power lies in iterative transformation — building complexity step by step.

Feistel Cipher Design Features (1)

How Feistel achieves Diffusion & Confusion?

Block size:

Larger blocks = more diffusion and stronger security.

Common: 64 bits (DES), 128 bits (modern ciphers).

Key size:

Larger key = higher resistance to brute-force attacks.

Common today: 128–256 bits.

- Both affect speed and security balance – designers must find a balance between making the cipher strong enough and keeping it fast enough to use in real systems.

Feistel Cipher Design Features (2)

How Feistel achieves Diffusion & Confusion?

Number of rounds:

- More rounds → more confusion and diffusion.
- DES uses 16 rounds.

Round function (F):

Should be nonlinear, complex, and depend on both data and key.

Subkey generation:

- Each round uses a different subkey derived from the main key.
- Must avoid patterns to maintain strength – subkeys must be different, random-looking, and hard to predict.

Feistel in Financial Systems

- Feistel-based encryption (DES) secured ATMs, banking systems, and credit card transactions from the 1970s to 2000s.
- Enabled secure electronic payments and PIN protection.
- Replaced by AES today, but Feistel principles still shape modern cryptography.
- Legacy systems still depend on Feistel-derived methods for backward compatibility.

Data Encryption Standard (DES)

- DES was created and approved as a U.S. standard in 1977 by NIST.
- It became the main encryption method for data security for over 20 years.
- DES works by encrypting data in 64-bit blocks using a 56-bit secret key.
- It was widely used in banks, government, and business systems until 2001, when AES replaced it.

How DES Works

DES uses the Feistel structure (same for encryption and decryption).

The process has three main steps:

Initial Permutation (IP): Rearranges the bits of the input data.

16 Rounds: Data passes through 16 rounds of Feistel operations using different subkeys.

Final Permutation (FP): Rearranges bits again to produce the final ciphertext.

The result is a 64-bit ciphertext.

❖ **IP and FP** doesn't change the bit values, only their positions. It helps spread patterns and prepares the data for the Feistel rounds.

Subkey Generation

- DES doesn't use the same key for every round.
- It creates 16 different subkeys, one for each round.
- Each subkey is 48 bits long and generated from the original 56-bit key using shifts and permutations.
- This makes every round different and increases encryption strength.

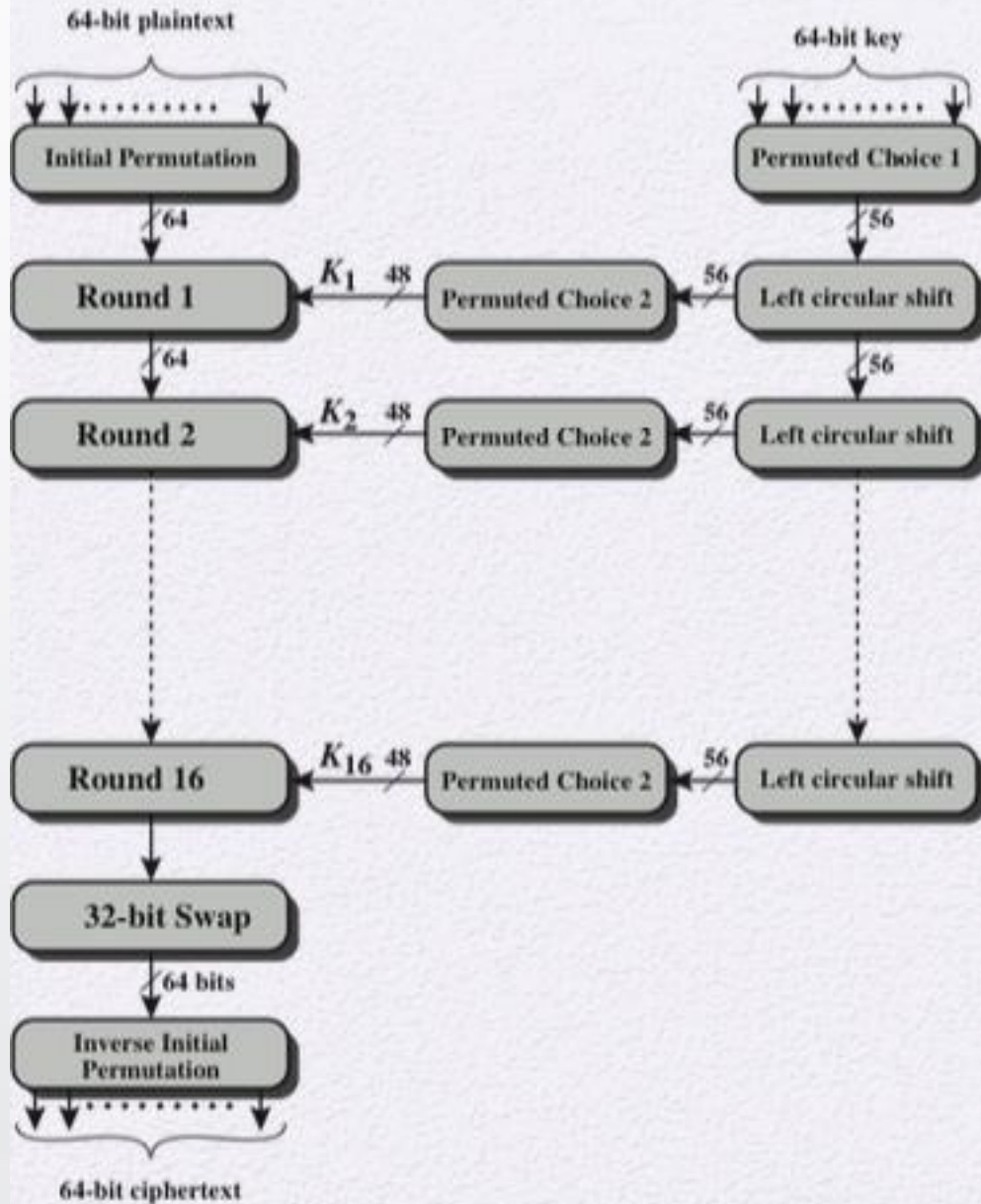


Figure 4.5 General Depiction of DES Encryption Algorithm

Steps for One DES Round (Round N)

1.Split the Block: The 64-bit input from the previous round is split into two equal parts: the **32-bit Left Half (L_N-1)** and the **32-bit Right Half (R_N-1)**.

2.Generate Subkey: The 48-bit subkey (K_N) for this specific round is derived from the main key using the key scheduling process (as shown on the right side of the diagram).

3.Feistel Function (F): The Right Half (R_N-1) is processed by the Feistel function f . This involves three main sub-steps:

Expansion: R_N-1 is expanded from 32 bits to 48 bits.

XOR with Subkey: The expanded 48 bits are XORed with the 48-bit subkey (K_N).

S-Box Substitution: The result is passed through eight S-Boxes (Substitution Boxes) and then a final Permutation, yielding a 32-bit output.

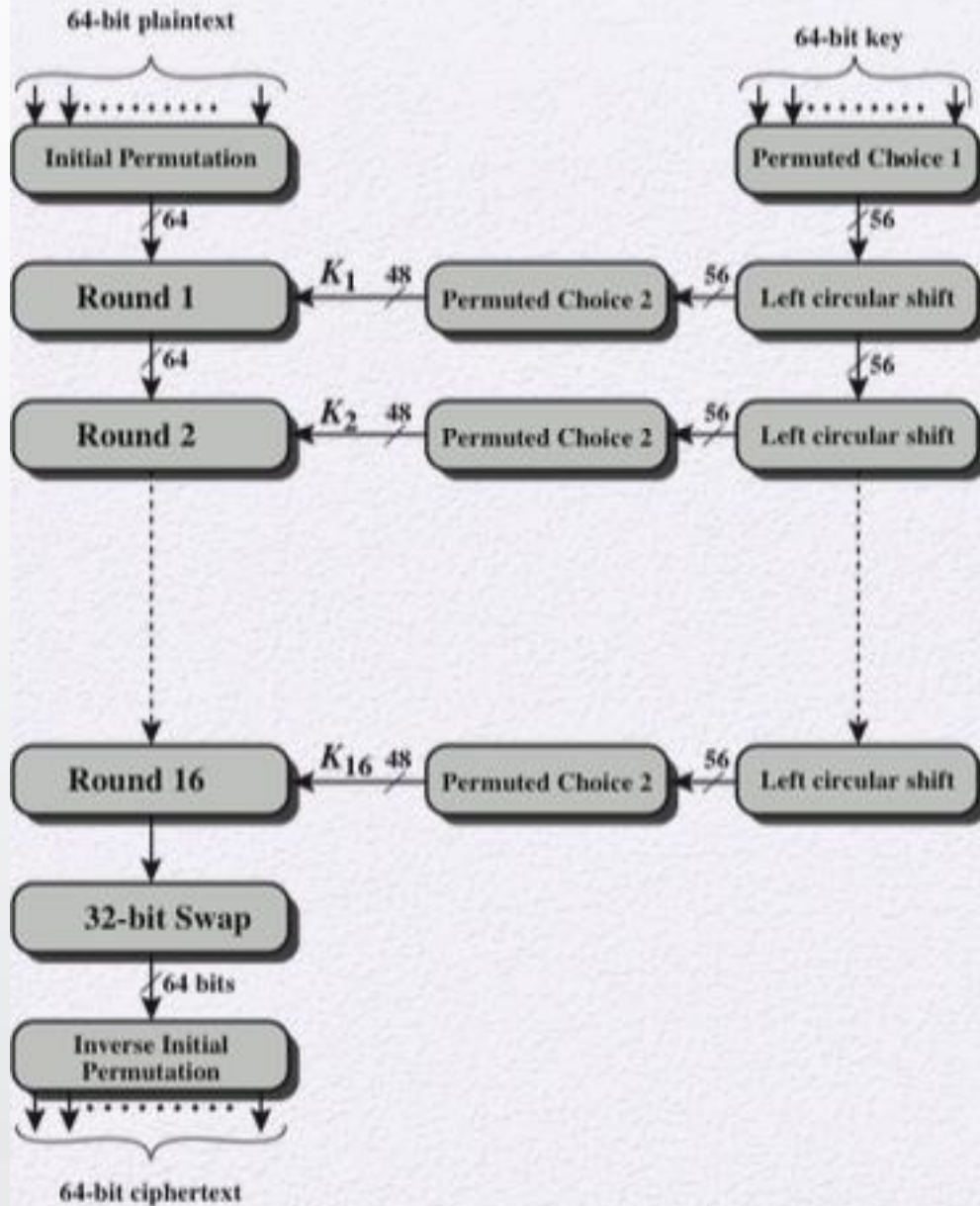


Figure 4.5 General Depiction of DES Encryption Algorithm

Steps for One DES Round (Round N)

4. Calculate New Right Half (R_N): The 32-bit output of the Feistel function is XORed with the old Left Half (L_{N-1}). This result becomes the new Right Half for the next round.

$$R_N = L_{N-1} \oplus f(R_{N-1}, K_N)$$

5. Calculate New Left Half (L_N): The old Right Half (R_{N-1}) simply becomes the new Left Half for the next round.

$$L_N = R_{N-1}$$

Combine: The New Left Half (L_N) and New Right Half (R_N) are concatenated to form the 64-bit input for Round N+1.

The Function F (Heart of DES)

- Function F is used in every round of DES.
- It takes 32 bits of data from the right side and processes it as follows:
- Expansion: Expands the 32 bits to 48 bits.
- Key Mixing: Combines expanded data with the 48-bit subkey using XOR.
- Substitution: Passes the result through S-boxes, replacing groups of bits to create confusion.
- Permutation: Rearranges the output bits to spread them out (diffusion).
- This process hides patterns and makes the ciphertext look random.

The Avalanche Effect

- A good cipher should show the avalanche effect:
 - A tiny change in input (like flipping 1 bit) causes many output bits to change.
 - It's a key property of strong ciphers like DES and AES.
- In DES, changing one bit in the plaintext or key changes about half of the ciphertext bits.
- This property makes the output unpredictable and secure.

Avalanche Effect Example

Plaintext	Changed Bit	Ciphertext	% of Bits Changed
1001	—	A7C3	—
1000	1 bit flip	4F99	~50%

This table shows that even a single-bit change in the input makes the ciphertext completely different — demonstrating strong diffusion.

Strengths of DES

- Strong structure: Based on the secure Feistel design.
- Good confusion and diffusion: Through the S-boxes and permutations.
- Proven design: Survived years of cryptanalysis and testing.
- Foundation for newer ciphers: Inspired algorithms like Blowfish and Triple DES.

Weaknesses of DES

- Small key size (56 bits): Too short for today's computing power.
- Vulnerable to brute-force attacks: Modern computers can try all keys in a short time.
- Advanced attacks discovered: Such as differential cryptanalysis, which can break it with enough data.

Because of these issues, DES is no longer considered secure.

Timing Attacks

- Every encryption operation takes a little time to run on a computer. Sometimes, the time it takes depends on the key or data.
- Attackers can measure how long encryption takes to guess information about the key.
- Small timing differences may reveal clues about internal operations.
- Even if the algorithm itself is strong (like DES), a careless implementation can leak information through timing.

DES in Real Life

- DES was used in many financial and government systems.
- Examples: ATM transactions, credit card processing, and secure file transfers.
- It followed international standards (ISO, ANSI).
- Still used in some older systems for backward compatibility, but no longer recommended.

Triple DES (3DES)

- To make DES stronger, it was modified into Triple DES (3DES).
- It applies DES three times with two or three different keys.
- Effective key length: 112 or 168 bits.
- More secure than DES, but three times slower.
- Used as a temporary replacement before AES became standard.

Key Lesson from DES

- DES was a great design for its time, but technology caught up.
- What was secure in the 1970s became weak by the 2000s.
- Lesson: Cryptography must evolve with computing power.
- Always update encryption methods to stay ahead of attackers.

Block Cipher Design Principles

Number of Rounds

More rounds = stronger cipher.

DES (16 rounds) would be weaker if <15 rounds.

Function F Design

Nonlinearity = resistance to cryptanalysis.

Strict Avalanche Criterion (SAC): 1-bit input change $\rightarrow$ 50% output bits change.

Bit Independence Criterion (BIC): output bits change independently — changing one input bit should not make multiple output bits flip together in a predictable way.

Lecture

4

Block Cipher Design Principles

Key Schedule Algorithm

Subkeys must appear independent.

Should resist attacks that exploit weak subkeys.

Confusion & Diffusion Reminder

Confusion = obscure key–ciphertext link.

Diffusion = spread plaintext info across ciphertext.

Stream vs Block Ciphers

- Stream: bit by bit.
- Block: groups of bits.

Feistel

- Alternating substitution + permutation.
- Basis of DES.

DES

- 64-bit block, 56-bit key.
- Avalanche effect.
- Strong but outdated.

Design Principles

Rounds number, Function F, Key schedule, Strict Avalanche Criterion (SAC)

Bit Independence Criterion (BIC).

References

- Stallings, W. (2017). Cryptography and Network Security, 7th Ed. Pearson.
- Schneier, B. (2015). Applied Cryptography. Wiley.
- Katz, J., & Lindell, Y. (2020). Introduction to Modern Cryptography. CRC Press.
- Nelson, B., Phillips, A., & Steuart, C. (2019). Guide to Computer Forensics and Investigations. Cengage.
- Terzioğlu, T. (2022). Lecture Notes on Block Ciphers and DES.