



CS458/CS558

Introduction to Computer Security



Transposition Ciphers

- ◆ Consider classical **transposition** or **permutation** ciphers
- ◆ Hide the message by **rearranging** the letter order without altering the actual letters used



Rail Fence cipher

- ◆ Write message letters out diagonally over a number of rows
- ◆ Then read the letters row by row
- ◆ Encrypt the message "meet me after the toga party" with a rail fence of depth 2


```

m e m a t r h t g p r y
e t e f e t e o a a t

```


Ciphertext: MEMATRHTGPRYETFETEOAAT

- ◆ **Think:** how to encrypt the above message using rail fence cipher of depth 3?



Rail Fence cipher

- ◆ Write message letters out diagonally over a number of rows
- ◆ Then read the letters row by row
- ◆ Encrypt the message "meet me after the toga party" with a rail fence of depth 3



Rail Fence cipher

- ◆ Write message letters out diagonally over a number of rows
- ◆ Then read the letters row by row
- ◆ Encrypt the message "meet me after the toga party" with a rail fence of depth 3

```


m t a e h o p t
e m f r e g a y
e e t t a r

```

Ciphertext: MTAEHOPTEMFREGAYEETTAR

- ◆ How to decrypt a ciphertext with 3 rows?

ciphertext: CPEERYOURCIMTSUT



Rail Fence cipher: Decryption

- ◆ Example:

ciphertext: CPEERYOURCIMTSUT

|row| = 3

❖ |cipher| = 16

❖ $16/3 = 5, 16 \bmod 3 = 1 \rightarrow$

1st row: 5+1 = 6 letters

2nd row: 5 letters, 3rd row: 5 letters

```

C P E E R Y
O U R C I
M T S U T

```

→ Plaintext: computersecurity

Rail Fence cipher: Decryption

- How to decrypt a ciphertext
 - Let $|row|$ be the number of rows
 - Compute the length of the ciphertext $|cipher|$
 - Compute the number of letters of each row
 - Write down the ciphertext row by row
 - Read the ciphertext diagonally

Row Transposition Ciphers

- A more complex transposition
- Write letters of message out in rows over a specified number of columns
- Then reorder the columns according to some key before reading off the rows

Key: 3 4 2 1 5 6 7

Plaintext: a t t a c k p
o s t p o n e
d u n t i l t
w o a m x y z

Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

Row Transposition Ciphers

- A more complex transposition
- Write letters of message out in rows over a specified number of columns
- Then reorder the columns according to some key before reading off the rows

Key: 3 4 2 1 5 6 7

Plaintext: a t t a c k p
o s t p o n e
d u n t i l t
w o a m x y z

Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

- how to decrypt a ciphertext using the above key?

ciphertext: ATHNRIPTISORPNSOCZ

Row Transposition Ciphers: Decryption

- How to decrypt a ciphertext?

ciphertext: ATHNRIPTISORPNSOCZ

$$|cipher| = 21, |key| = 7 \rightarrow |row| = 3$$

Key: 3 4 2 1 5 6 7

Ciphertext: T R A N S P O
S I T I O N C
I P H E R S Z

Plaintext: transpositionciphers

Product Ciphers

- Ciphers using substitutions or transpositions are not secure because of language characteristics
- Hence consider using several ciphers in succession to make harder
 - Two substitutions make a more complex substitution
 - Two transpositions make a more complex transposition
 - But a substitution followed by a transposition makes a much harder cipher
 - > This is bridge from classical to modern ciphers



Chapter 3

Block Ciphers and the Data Encryption Standard (DES)

Block Ciphers

- ♦ **Block ciphers:** a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
 - ❖ Typically, a block size of 64 or 128 bits is used
 - ❖ Many current ciphers are block ciphers
 - ❖ Broader range of applications
 - ❖ **DES (Data Encryption Standard):** one of the most widely used cryptographic algorithms, especially in financial applications.

Block Cipher Principles

- ♦ A block cipher operates on a **plaintext** block of n bits to produce a **ciphertext** block of n bits.
- ♦ There are 2^n possible different plaintext blocks.

Block Cipher Principles

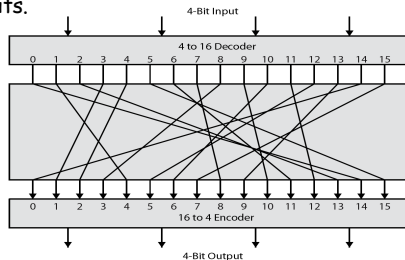
- ♦ A block cipher operates on a **plaintext** block of n bits to produce a **ciphertext** block of n bits.
 - ♦ There are 2^n possible different plaintext blocks
 - ♦ For the encryption to be **reversible** (for decryption to be possible), each must produce a unique ciphertext block
 - ♦ **Reversible:**
- | Plaintext | Ciphertext |
|-----------|------------|
| 00 | 11 |
| 01 | 10 |
| 10 | 00 |
| 11 | 01 |

Block Cipher Principles

- ♦ A block cipher operates on a **plaintext** block of n bits to produce a **ciphertext** block of n bits.
 - ♦ There are 2^n possible different plaintext blocks,
 - ♦ for the encryption to be **reversible** (for decryption to be possible), each must produce a unique ciphertext block
 - ♦ **Irreversible:**
- | Plaintext | Ciphertext |
|-----------|------------|
| 00 | 11 |
| 01 | 10 |
| 10 | 00 |
| 11 | 00 |

Ideal Block Cipher

- ♦ The logic of a general substitution cipher for $n=4$
 - ❖ A 4-bit input produces one of 16 possible input states, which is mapped into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits.



Ideal Block Cipher

- ♦ **Ideal block cipher:** allows for maximum number of possible encryption mappings from the plaintext block.
 - ❖ n bits \rightarrow possible mappings

Ideal Block Cipher

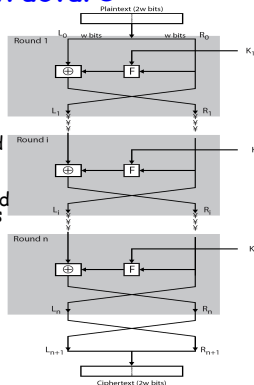
- **Ideal block cipher**: allows for maximum number of possible encryption mappings from the plaintext block.
 - ❖ n bits $\rightarrow 2^n$ possible mappings
 - ❖ Impractical when n is large
 - Each mapping constitutes a key
 - $n=64 \rightarrow$ the key size is $> 63 \cdot 2^{63}$ -- not practical.

The Feistel Cipher

- **The Feistel cipher**: approximate the ideal block cipher by utilizing the concept of a product cipher
 - ❖ Develop a block cipher with a key length of k bits and a block length of n bits, allowing a total of 2^k possible mappings (rather than $2^n!$ Mappings)
 - ❖ Alternates substitution and permutation
- Most symmetric block ciphers are based on a **Feistel Cipher Structure**

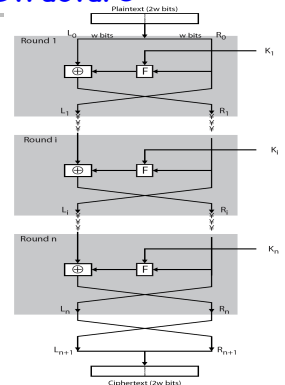
Feistel Cipher Structure

- **Inputs**: a plaintext block of length $2w$ bits and a key K .
- The plaintext block is divided into two halves L_0, R_0
 - ❖ Pass through n rounds of processing and then combined to produce the ciphertext block
 - ❖ Each round i has inputs L_{i-1} and R_{i-1} derived from the previous round, as well as a subkey K_i derived from K .
 - ❖ In general, K_i are different from K and from each other.



Feistel Cipher Structure

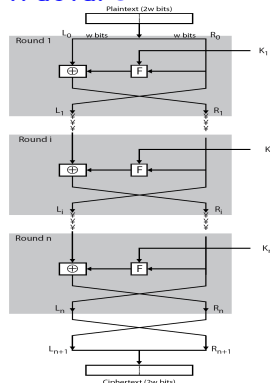
- A **substitution** is applied to the left half
 - ❖ Applying a **round function** F to the right half of the data. F is parameterized by the round subkey K_i
 - ❖ Then take the **exclusive-OR (XOR)** of the output of F and the left half of the data.
- A **permutation** is then performed that exchanges the two halves of the data.



Feistel Cipher Structure

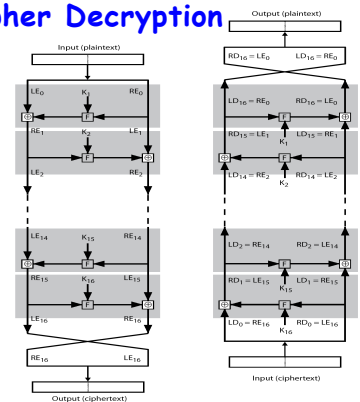
$$L_i = R_{i-1}$$

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



Feistel Cipher Decryption

- Same as the encryption process
- Except that the subkeys K_i are used in reverse order: K_n in the first round, ..., K_1 in the last round.



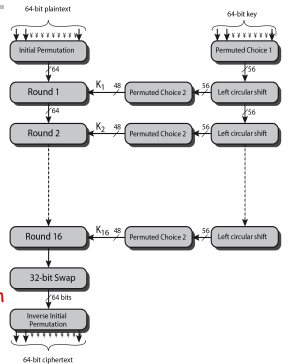
Data Encryption Standard (DES)

Data Encryption Standard (DES)

- Most widely used block cipher in world
- Developed in 1974 by IBM and the U.S. government
- The algorithm transforms 64-bit input in a series of steps into a 64-bit output.
- The **same** steps, with the **same** key, are used to reverse the encryption.
- Use of DES has flourished, especially in **financial applications**

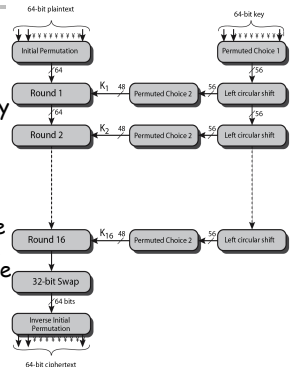
DES Encryption Overview

- The plaintext passes through an **initial permutation (IP)** that rearranges the bits to produce the permuted input
- Then passes through **16 rounds** of the same function, which involves both **permutation** and **substitution** function
- The left and right halves of the output of the last round are **swapped**, which is then passed through a **permutation** that is the **inverse of IP** to produce the 64-bit ciphertext



DES Encryption Overview

- 64-bit key** is passed through a permutation function.
- For each **16 round**, a subkey **K_i** is produced by the combination of a **left circular shift** and a **permutation**.
- Permutation function is the same for each round, but a **different subkey** is produced because of the repeated shifts of key bits



Initial Permutation (IP)

Input:

| | | | | | | | |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| M ₁ | M ₂ | M ₃ | M ₄ | M ₅ | M ₆ | M ₇ | M ₈ |
| M ₉ | M ₁₀ | M ₁₁ | M ₁₂ | M ₁₃ | M ₁₄ | M ₁₅ | M ₁₆ |
| M ₁₇ | M ₁₈ | M ₁₉ | M ₂₀ | M ₂₁ | M ₂₂ | M ₂₃ | M ₂₄ |
| M ₂₅ | M ₂₆ | M ₂₇ | M ₂₈ | M ₂₉ | M ₃₀ | M ₃₁ | M ₃₂ |
| M ₃₃ | M ₃₄ | M ₃₅ | M ₃₆ | M ₃₇ | M ₃₈ | M ₃₉ | M ₄₀ |
| M ₄₁ | M ₄₂ | M ₄₃ | M ₄₄ | M ₄₅ | M ₄₆ | M ₄₇ | M ₄₈ |
| M ₄₉ | M ₅₀ | M ₅₁ | M ₅₂ | M ₅₃ | M ₅₄ | M ₅₅ | M ₅₆ |
| M ₅₇ | M ₅₈ | M ₅₉ | M ₆₀ | M ₆₁ | M ₆₂ | M ₆₃ | M ₆₄ |

| | | | | | | | |
|----|----|----|----|----|----|----|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

Initial Permutation (IP)

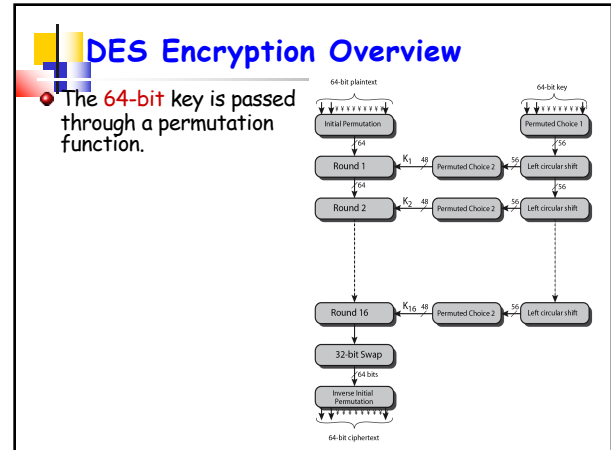
Output:

| | | | | | | | |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|----------------|
| M ₅₈ | M ₅₀ | M ₄₂ | M ₃₄ | M ₂₆ | M ₁₈ | M ₁₀ | M ₂ |
| M ₆₀ | M ₅₂ | M ₄₄ | M ₃₆ | M ₂₈ | M ₂₀ | M ₁₂ | M ₄ |
| M ₆₂ | M ₅₄ | M ₄₆ | M ₃₈ | M ₃₀ | M ₂₂ | M ₁₄ | M ₆ |
| M ₆₄ | M ₅₆ | M ₄₈ | M ₄₀ | M ₃₂ | M ₂₄ | M ₁₆ | M ₈ |
| M ₅₇ | M ₄₉ | M ₄₁ | M ₃₃ | M ₂₅ | M ₁₇ | M ₉ | M ₁ |
| M ₅₉ | M ₅₁ | M ₄₃ | M ₃₅ | M ₂₇ | M ₁₉ | M ₁₁ | M ₃ |
| M ₆₁ | M ₅₃ | M ₄₅ | M ₃₇ | M ₂₉ | M ₂₁ | M ₁₃ | M ₅ |
| M ₆₃ | M ₅₅ | M ₄₇ | M ₃₉ | M ₃₁ | M ₂₃ | M ₁₅ | M ₇ |

Example

Input: 00100....0

| | | | | | | | |
|----|----|----|----|----|----|----|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |



Permuted Choice One (PC-1)

| | | | | | | |
|----|----|----|----|----|----|----|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

Example

Input: 00100001....0

| | | | | | | |
|----|----|----|----|----|----|----|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

DES Encryption Overview

- For each 16 round, a subkey K_i is produced by the combination of a left circular shift and a permutation.
- The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits

Schedule of Left Circular Shifts

- The output of PC-1 is then treated as two 28 bits quantities, labeled C_i and D_i .
- At each round, C_i and D_i are separately subjected to a circular left shift, of 1 or 2 bits.

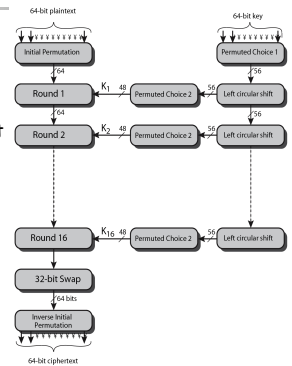
| Round number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|--------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| Bits rotated | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

Permuted Choice Two (PC-2)

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 14 | 17 | 11 | 24 | 1 | 5 | 3 | 28 |
| 15 | 6 | 21 | 10 | 23 | 19 | 12 | 4 |
| 26 | 8 | 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 | 30 | 40 |
| 51 | 45 | 33 | 48 | 44 | 49 | 39 | 56 |
| 34 | 53 | 46 | 42 | 50 | 36 | 29 | 32 |

DES Encryption Overview

- The left and right halves of the output of the last round are **swapped**, which is then passed through a **permutation** that is the **inverse of IP** to produce the 64-bit ciphertext
- ❖ $IP^{-1}(IP(M)) = M$



Inverse Initial Permutation (IP⁻¹)

Input:

| | | | | | | | |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|----------------|
| M ₅₈ | M ₅₀ | M ₄₂ | M ₃₄ | M ₂₆ | M ₁₈ | M ₁₀ | M ₂ |
| M ₆₀ | M ₅₂ | M ₄₄ | M ₃₆ | M ₂₈ | M ₂₀ | M ₁₂ | M ₄ |
| M ₆₂ | M ₅₄ | M ₄₆ | M ₃₈ | M ₃₀ | M ₂₂ | M ₁₄ | M ₆ |
| M ₆₄ | M ₅₆ | M ₄₈ | M ₄₀ | M ₃₂ | M ₂₄ | M ₁₆ | M ₈ |
| M ₅₇ | M ₄₉ | M ₄₁ | M ₃₃ | M ₂₅ | M ₁₇ | M ₉ | M ₁ |
| M ₅₉ | M ₅₁ | M ₄₃ | M ₃₅ | M ₂₇ | M ₁₉ | M ₁₁ | M ₃ |
| M ₆₁ | M ₅₃ | M ₄₅ | M ₃₇ | M ₂₉ | M ₂₁ | M ₁₃ | M ₅ |
| M ₆₃ | M ₅₅ | M ₄₇ | M ₃₉ | M ₃₁ | M ₂₃ | M ₁₅ | M ₇ |

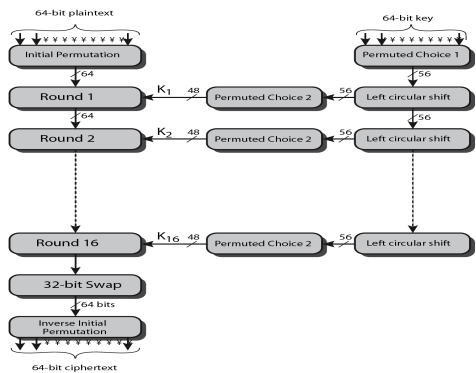
| | | | | | | | |
|----|---|----|----|----|----|----|----|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

Inverse Initial Permutation (IP⁻¹)

Output:

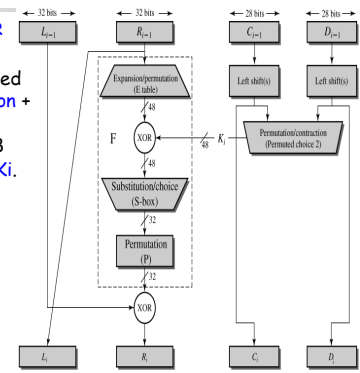
| | | | | | | | |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| M ₁ | M ₂ | M ₃ | M ₄ | M ₅ | M ₆ | M ₇ | M ₈ |
| M ₉ | M ₁₀ | M ₁₁ | M ₁₂ | M ₁₃ | M ₁₄ | M ₁₅ | M ₁₆ |
| M ₁₇ | M ₁₈ | M ₁₉ | M ₂₀ | M ₂₁ | M ₂₂ | M ₂₃ | M ₂₄ |
| M ₂₅ | M ₂₆ | M ₂₇ | M ₂₈ | M ₂₉ | M ₃₀ | M ₃₁ | M ₃₂ |
| M ₃₃ | M ₃₄ | M ₃₅ | M ₃₆ | M ₃₇ | M ₃₈ | M ₃₉ | M ₄₀ |
| M ₄₁ | M ₄₂ | M ₄₃ | M ₄₄ | M ₄₅ | M ₄₆ | M ₄₇ | M ₄₈ |
| M ₄₉ | M ₅₀ | M ₅₁ | M ₅₂ | M ₅₃ | M ₅₄ | M ₅₅ | M ₅₆ |
| M ₅₇ | M ₅₈ | M ₅₉ | M ₆₀ | M ₆₁ | M ₆₂ | M ₆₃ | M ₆₄ |

DES Encryption Overview



Single Round of DES Algorithm

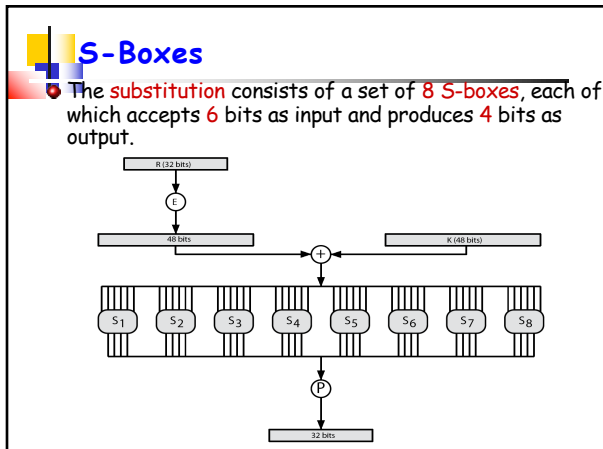
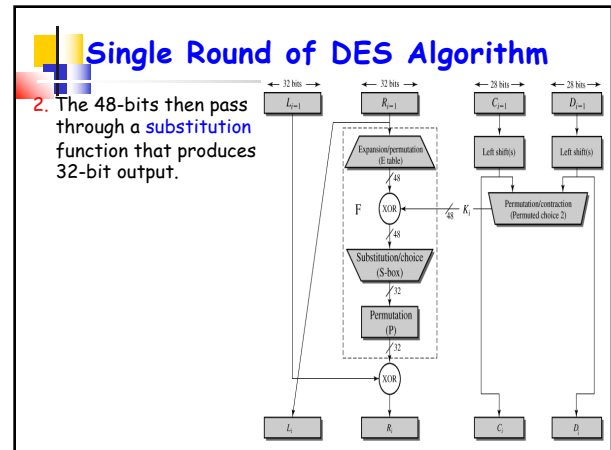
- Uses two 32-bit L & R halves. **Ki: 48 bits**
- 1. The R input is expanded to 48 bits (permutation + duplication of 16 of R bits); the resulting 48 bits are **XORed** with **Ki**.



Expanded Permutation (E)

Input: 0011000....0

| | | | | | |
|----|----|----|----|----|----|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 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 | 30 | 31 | 32 | 1 |



S-Boxes

The substitution consists of a set of 8 S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.

- The first and last bits of the input to S_i form a 2-bit binary number to select one of 4 substitutions defined by the four rows (0, 1, 2, 3) in the table for S_i .
- The middle 4 bits select one of 16 columns (0-15).
- E.g. in S_1 , input 011001

| | | | | | | | | | | | | | | | |
|----|----|----|---|----|----|----|----|----|----|----|----|----|----|---|----|
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

S-Boxes

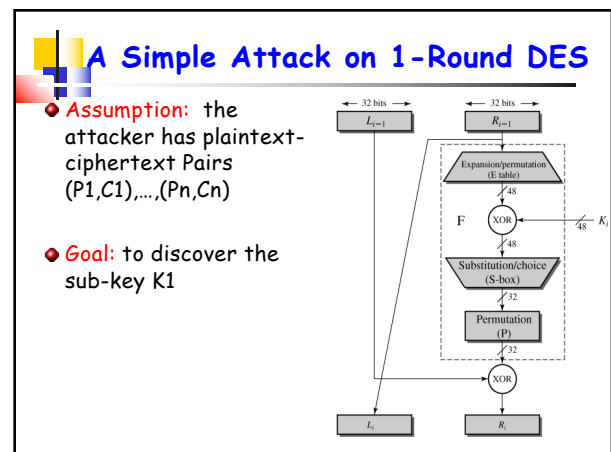
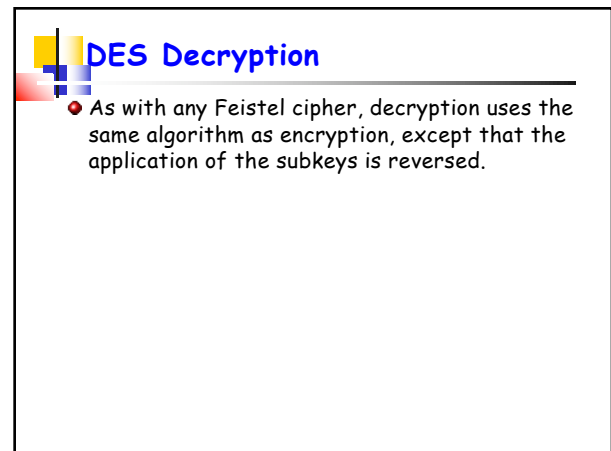
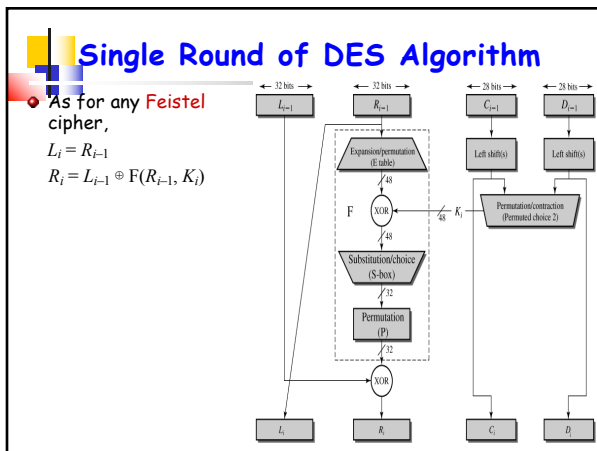
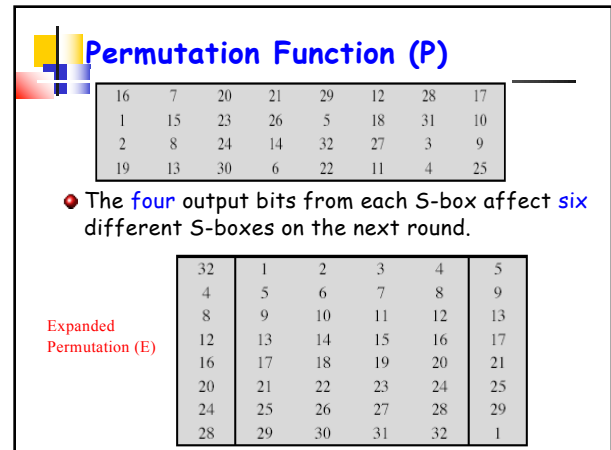
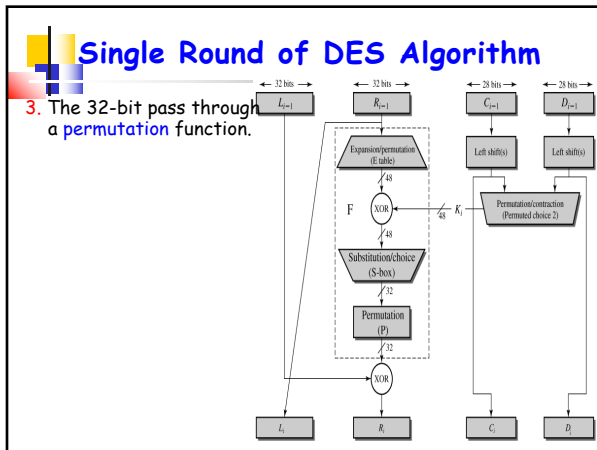
The substitution consists of a set of 8 S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.

- The first and last bits of the input to S_i form a 2-bit binary number to select one of 4 substitutions defined by the four rows (0, 1, 2, 3) in the table for S_i .
- The middle 4 bits select one of 16 columns (0-15).
- E.g. in S_1 , input 011001
 - The row is 01 (row 1)
 - The column is 1100 (column 12)
 - The value is 9 - output is 1001.

| | | | | | | | | | | | | | | | |
|----|----|----|---|----|----|----|----|----|----|----|----|----|----|---|----|
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

S-Box: Design Criteria

- The design of the round function focuses on the design of s-boxes and on the permutation P.
- The design was primarily aimed at thwarting differential cryptanalysis.
 - Any change to the input to an S-box should result in random-looking changes to the output
 - No output bit of any S-box should be too close a linear function of the input bits



A Simple Attack on 1-Round DES

$P1 = (L0, R0)$
 $C1 = (L1, R1)$

$R0 \rightarrow$ output of E table (1)

$R1$ and $L0 \rightarrow$
 output of P table \rightarrow
 output of S-box \rightarrow
 4 possible inputs of each S-box (2)

A Simple Attack on 1-Round DES

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 16 | 7 | 20 | 21 | 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 | 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 | 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 | 22 | 11 | 4 | 25 |

output of P table \rightarrow output of S-box

E.g. the output of P table: 1010000...0

Output of s1:

A Simple Attack on 1-Round DES

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 16 | 7 | 20 | 21 | 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 | 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 | 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 | 22 | 11 | 4 | 25 |

output of P table \rightarrow output of S-box

E.g. the output of P table: 1010000...0

Output of s1: (the 16th bit and 20th bit are 1; others are 0)

A Simple Attack on 1-Round DES

output of S-box \rightarrow
 4 possible inputs of each S-box

E.g. Output of S1: 0000

| | | | | | | | | | | | | | | | |
|----|----|----|---|----|----|----|----|----|----|----|----|----|----|---|----|
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

A Simple Attack on 1-Round DES

output of S-box \rightarrow
 4 possible inputs of each S-box

E.g. Output of S1: 0000

Input of S1:
 011100 (row 0 column 14)
 000001 (row 1 column 0)
 111110 (row 2 column 15)
 111011 (row 3 column 13)

| | | | | | | | | | | | | | | | |
|----|----|----|---|----|----|----|----|----|----|----|----|----|----|---|----|
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

A Simple Attack on 1-Round DES

output of E table &
 4 possible inputs of each S-box
 \rightarrow 4 possible values for
 each $K1_j$, where $K1 = (K11 \dots K18)$

Try other plaintext-ciphertext
 pairs until $K1$ is discovered

Avalanche Effect

- Avalanche Effect:**
 - Desirable property of any encryption algorithm.
 - Small change in either the plaintext or the key should produce a significant change in the ciphertext.
 - Change in 1 bit of the plaintext or the key should produce a change in **many** bits of the ciphertext.
- Making attempts of guessing keys impossible
- DES exhibits strong avalanche

Avalanche Effect

Two plaintexts:

❖ P1:

```
00000000 00000000
00000000 00000000
00000000 00000000
00000000 00000000
```

❖ P2:

```
10000000 00000000 00000000
00000000 00000000 00000000
00000000 00000000
```

❖ Key K:

```
00000001 1001011 0100100
1100010 0011100 0011000
0011100 0110010
```

| (a) Change in Plaintext | |
|-------------------------|----------------------------|
| Round | Number of bits that differ |
| 0 | 1 |
| 1 | 6 |
| 2 | 21 |
| 3 | 35 |
| 4 | 39 |
| 5 | 34 |
| 6 | 32 |
| 7 | 31 |
| 8 | 29 |
| 9 | 42 |
| 10 | 44 |
| 11 | 32 |
| 12 | 30 |
| 13 | 30 |
| 14 | 26 |
| 15 | 29 |
| 16 | 34 |

Avalanche Effect

One plaintext P:

```
01101000 10000101
00101111 01111010
00010011 01110110
11101011 10100100
```

Two keys:

❖ K1

```
1110010 1111011 1101111
0011000 0011101 0000100
0110001 1101110
```

❖ K2

```
0110010 1111011 1101111
0011000 0011101 0000100
0110001 1101110
```

| (a) Change in Plaintext | | (b) Change in Key | |
|-------------------------|----------------------------|-------------------|----------------------------|
| Round | Number of bits that differ | Round | Number of bits that differ |
| 0 | 1 | 0 | 0 |
| 1 | 6 | 1 | 2 |
| 2 | 21 | 2 | 14 |
| 3 | 35 | 3 | 28 |
| 4 | 39 | 4 | 32 |
| 5 | 34 | 5 | 30 |
| 6 | 32 | 6 | 32 |
| 7 | 31 | 7 | 35 |
| 8 | 29 | 8 | 34 |
| 9 | 42 | 9 | 40 |
| 10 | 44 | 10 | 38 |
| 11 | 32 | 11 | 31 |
| 12 | 30 | 12 | 33 |
| 13 | 30 | 13 | 28 |
| 14 | 26 | 14 | 26 |
| 15 | 29 | 15 | 34 |
| 16 | 34 | 16 | 35 |

DES - Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- Brute force search looks hard
 - assume: half of the key space has to be searched, a single machine performing one DES encryption **per microsecond** would take more than a thousand years to break the cipher
- Recent advances have shown is possible
 - In 1977, Diffie and Hellman: technology existed to build a parallel machine with 1 million encryption devices; each performs **one encryption per microsecond** → 10 hours
 - In 1998, Electronic Frontier Foundation (EFF) had broken a DES encryption using a computer built for less than \$250K. The attack took less than 3 days.

S-Box Modifications and Their Effect in DES-like Encryption Systems

http://www.sans.org/reading_room/whitepapers/vpns/s-box-modifications-effect-des-like-encryption-systems_768