Data Encryption Standard

Dr. E.SURESH BABU

Assistant Professor

Computer Science and Engineering Department

National Institute of Technology, Warangal

Warangal

Outline

- ❖ Introduction to Block Cipher
- ❖ DES (Data Encryption Standard)
- ❖ DES Encryption Algorithm
- ❖ Key Transformation in DEA
- ❖ One Round of Processing in DEA
- ❖ DES Example

Introduction to Block Cipher

DES (DATA ENCRYPTION STANDARD)

History

- The most widely used **Encryption Scheme** is based on the **Data Encryption Standard (DES)**
  - ✓ Adopted by **National Institute of Standards and Technology (NIST)** in **1977**
  - ✓ Formerly known as **National Bureau of Standards(NBS)**

History...

- The **Data Encryption Standard (DES),** known as the **Data Encryption Algorithm (DEA)** by ANSI and ISO,
  - ✓ It has been a **worldwide standard** for **20 years.**
- **DES** is based on a **Lucifer cipher** developed earlier by **IBM**
  - ✓ **Lucifer cipher** is mainly used for **Lloyd’s of London** for **cash transfer.**

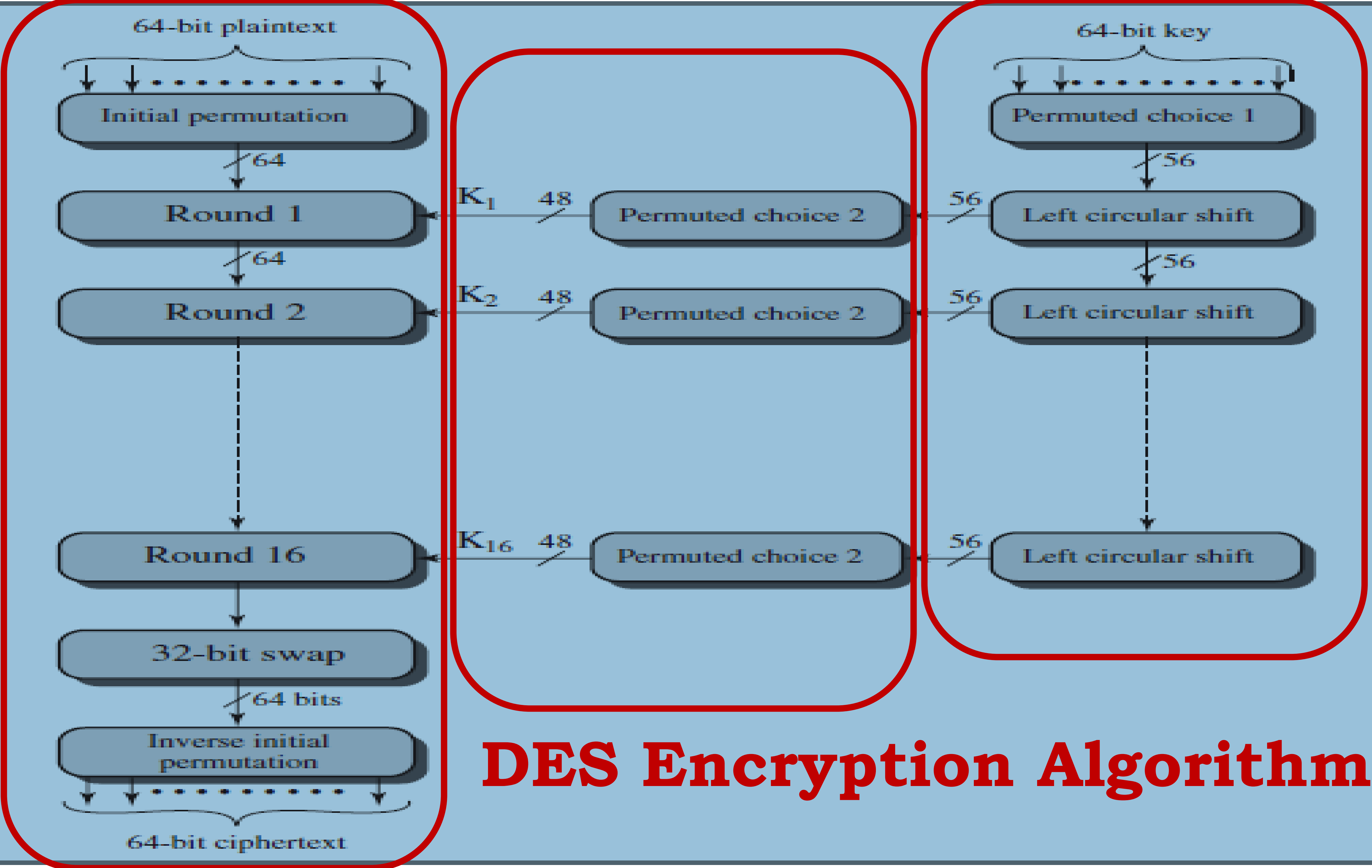
Introduction

- ❖ **DES** is a **Block Cipher**;
  - ✓ It encrypts **plaintext** in **64-bit blocks** ; The plaintext must be **64 bits in length**
  - ✓ The **key** is **56 bits** in length.
- ❖ **DES** is a **symmetric algorithm**;
  - ✓ The **same algorithm** and **key** are used for both **encryption and decryption**

Observation

- **DES** uses the **Feistel cipher structure** with **16 rounds of processing.**

DES Encryption Algorithm

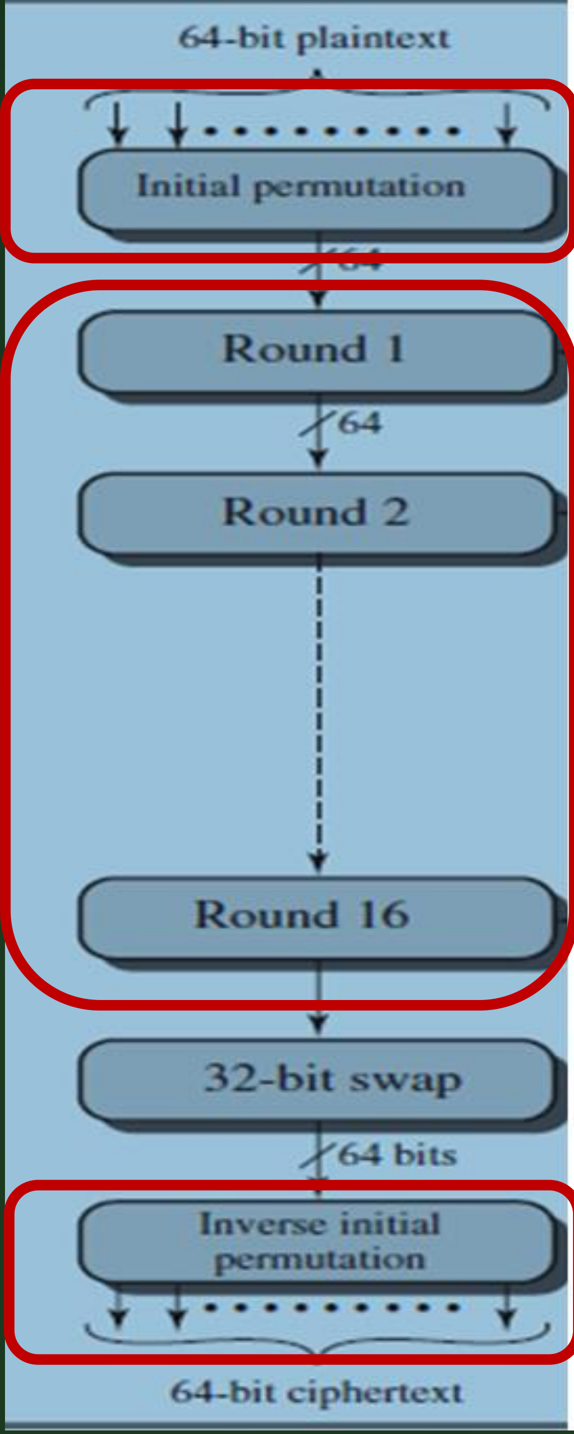


DES Encryption Algorithm

Processing of Plain Text

❖ Looking at the **left-hand side of the figure**, we can see that the **processing of the plaintext** proceeds in **Three Phases**.

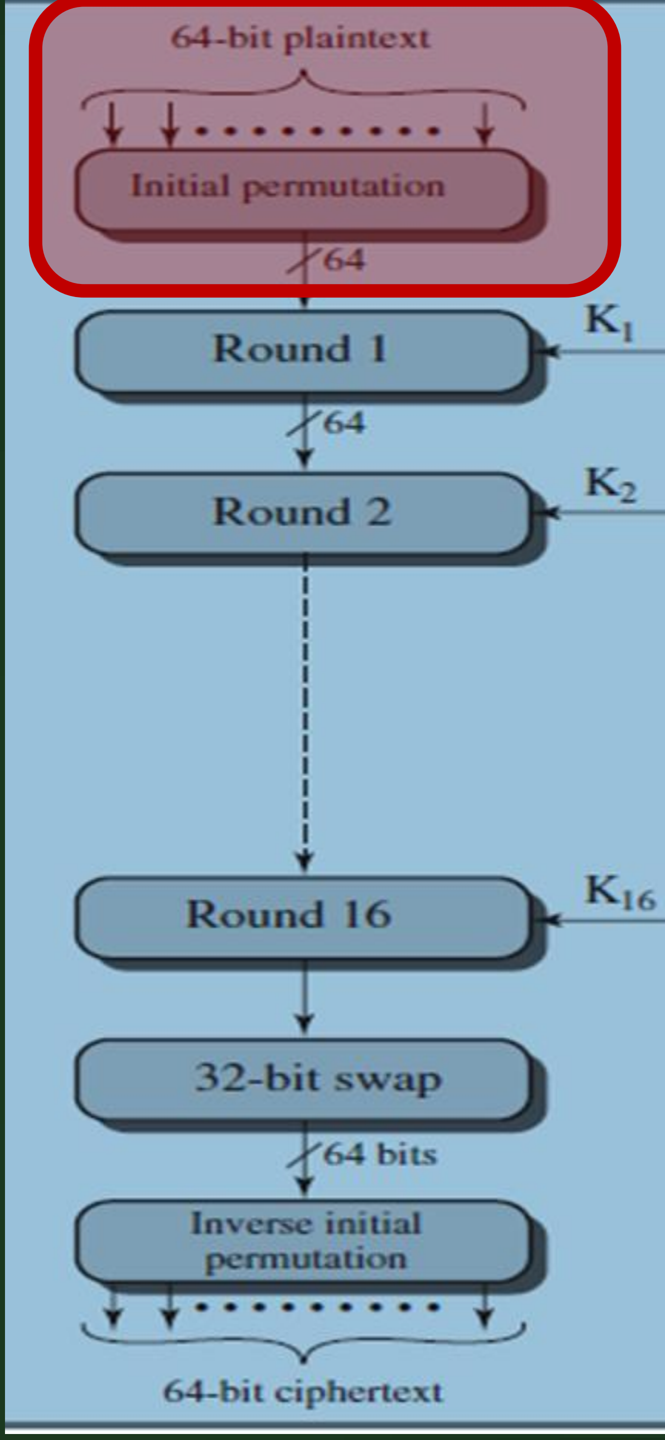
1. **Initial permutation (IP)**
2. **Sixteen rounds of the Processing using same function,**
3. **Final permutations,**



Initial Permutation (IP)

Initial Permutation (IP)

❖ The **64–bit Plaintext passes** through an **initial permutation (IP)** that **rearranges the bits** to produce the **permuted input.**



Initial Permutation

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 Input.

Initial Permutation (IP) Table

❖ The **input to a table** consists of **64 bits** numbered from **1 to 64.** The **64 entries** in the **permutation table** contain a **permutation of the numbers** from **1 to 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)

- Each **entry in the permutation table** indicates the **position of a numbered input bit** in the **output**.
- For Example, the **initial permutation** moves bit **58** of the **plaintext to bit position 1**, **bit 50 to bit position 2**, **bit 42 to bit position 3**, and so forth.

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

Reading the IP Table

- The **initial permutation** occurs before **First Round of Processing** ;
  - ✓ It transposes the **input block** which should be read **left to right, top to bottom** in the Table .

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)

❖ Consider the following **64-bit input M :**

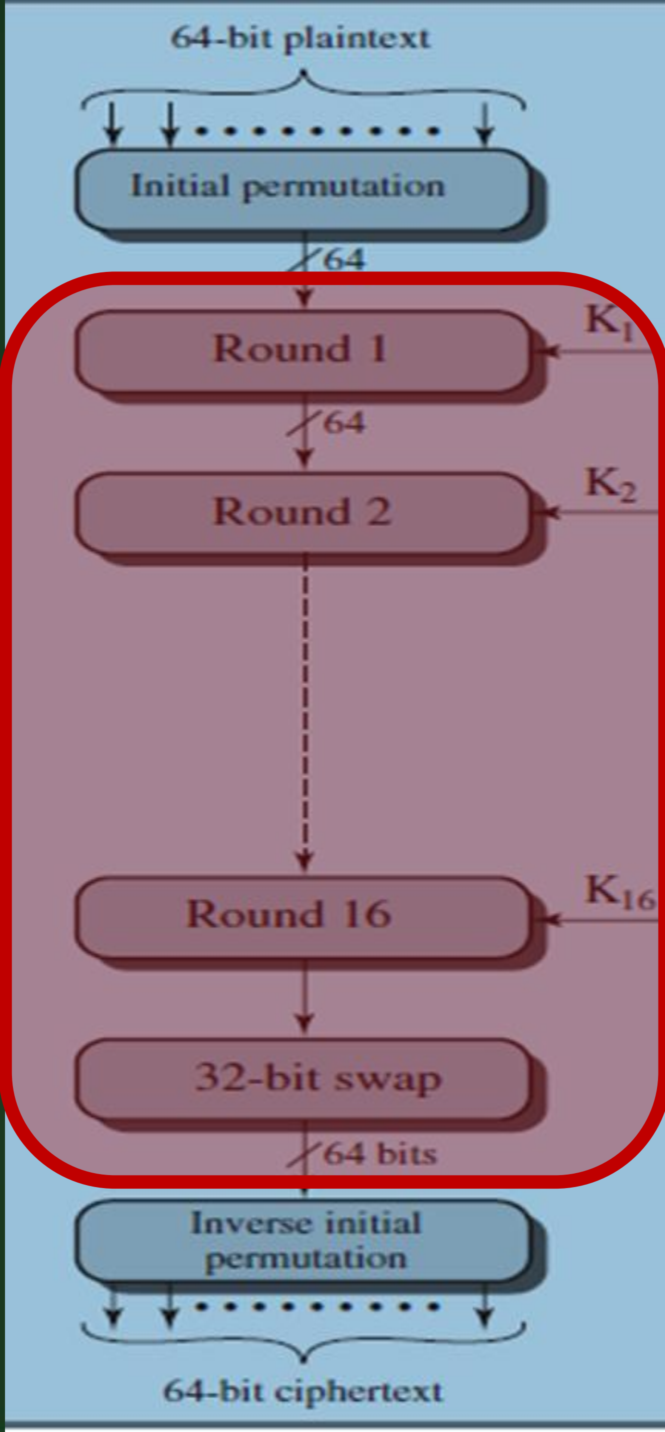
M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	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}

X = (IP(M)

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

64-bit plaintext → **Initial Permutation** → **Permuted Input.**

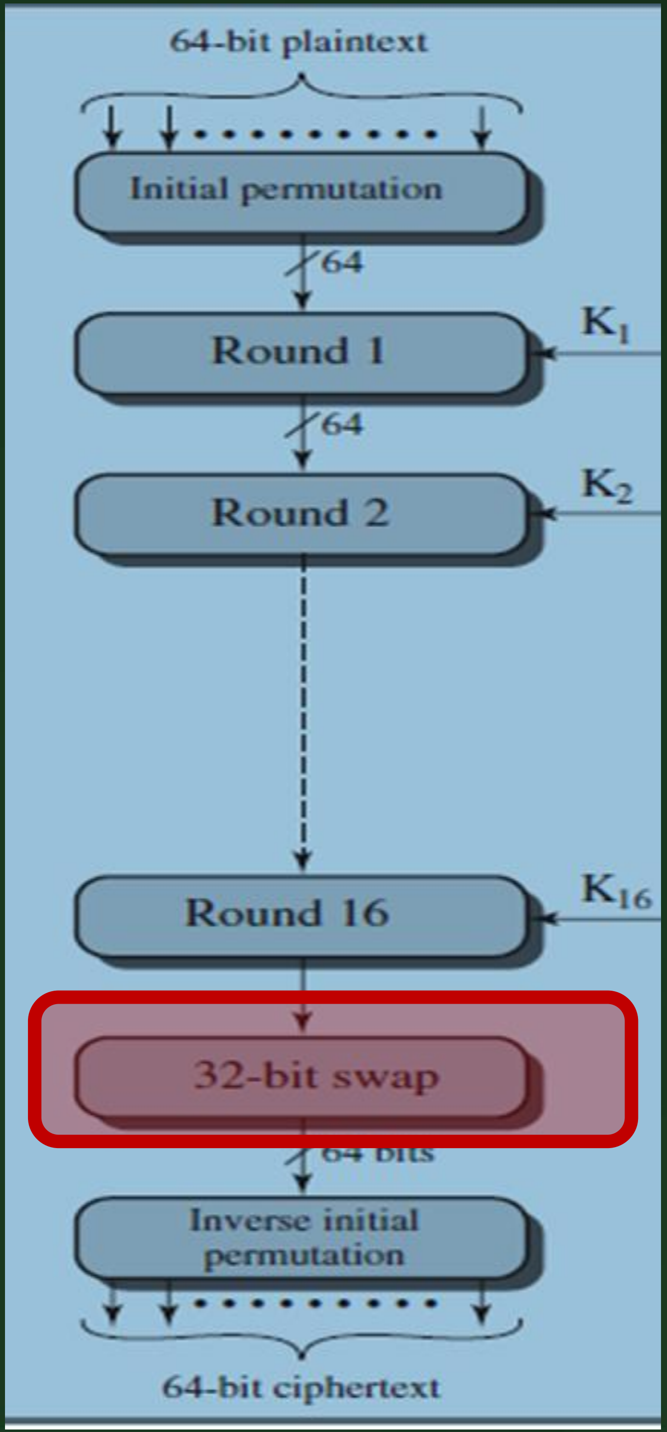
Sixteen Rounds of the Processing



Sixteen Rounds of the Processing

- ❖ The **Second phase** consisting of **sixteen rounds of processing** the same function,
  - ✓ It involves both **permutation and substitution functions.**
  - ✓ The output of the **last (sixteenth) round** consists of **64 bits** that are a **function of the input plaintext** and the **key.**

Swapping



64-bit plaintext
Initial permutation
64
Round 1
K_1
64
Round 2
K_2
Round 16
K_{16}
32-bit swap
64 bits
Inverse initial permutation
64-bit ciphertext

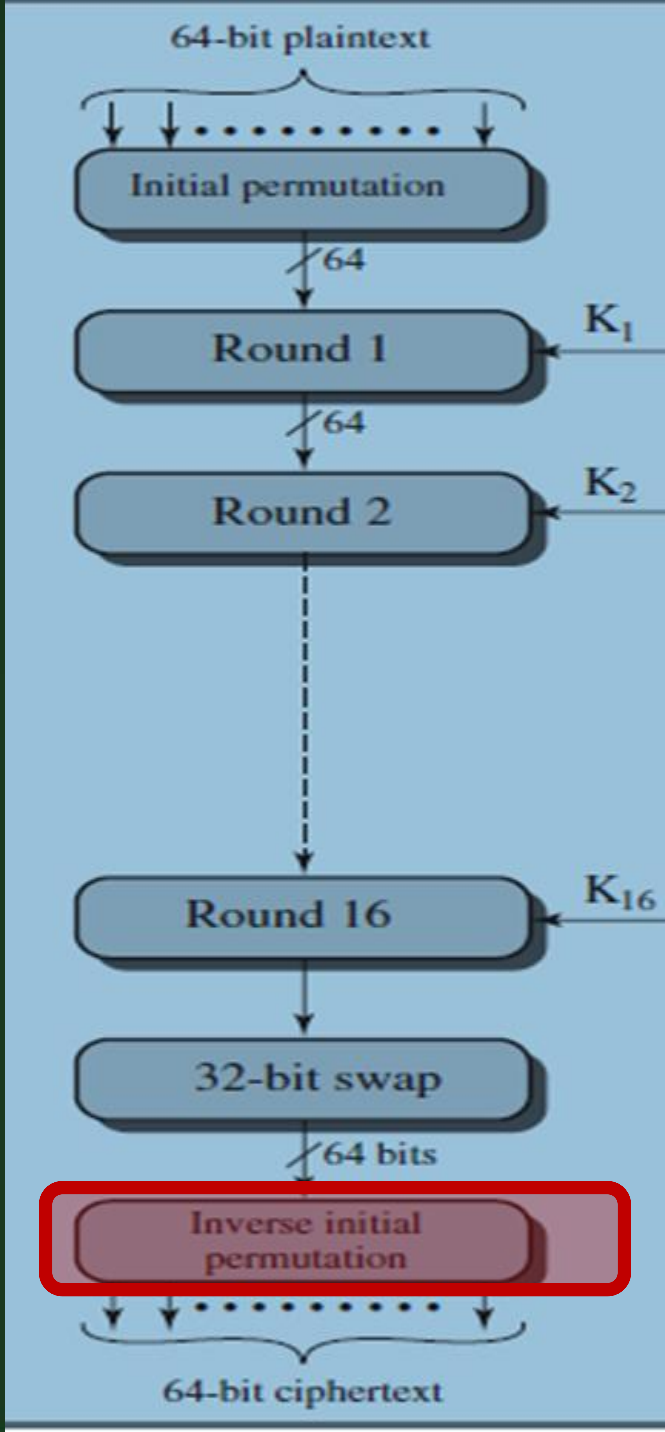
Swapping

- The **left and right halves** of the **output** are swapped to **produce the preoutput.**

Observation

- **Note that** the **left and right halves are not exchanged** after the **last round of DES**;
  - ✓ Instead the **concatenated block $R_{16}L_{16}$** is used as the **input to the final permutation**.
  - ✓ There's **nothing going on here;**

Final Permutation (FP)



Final Permutation (FP)

- The **final permutation** is the **inverse of the initial permutation**
  - ✓ it can be seen that the **original ordering of the bits** is restored
- After **Last Round of Processing DES**
  - ✓ It concatenate the **block $R_{16}L_{16}$** is used as the **input to the final permutation**.

Representation

64-bit plaintext After 16 Rounds

Final Permutation

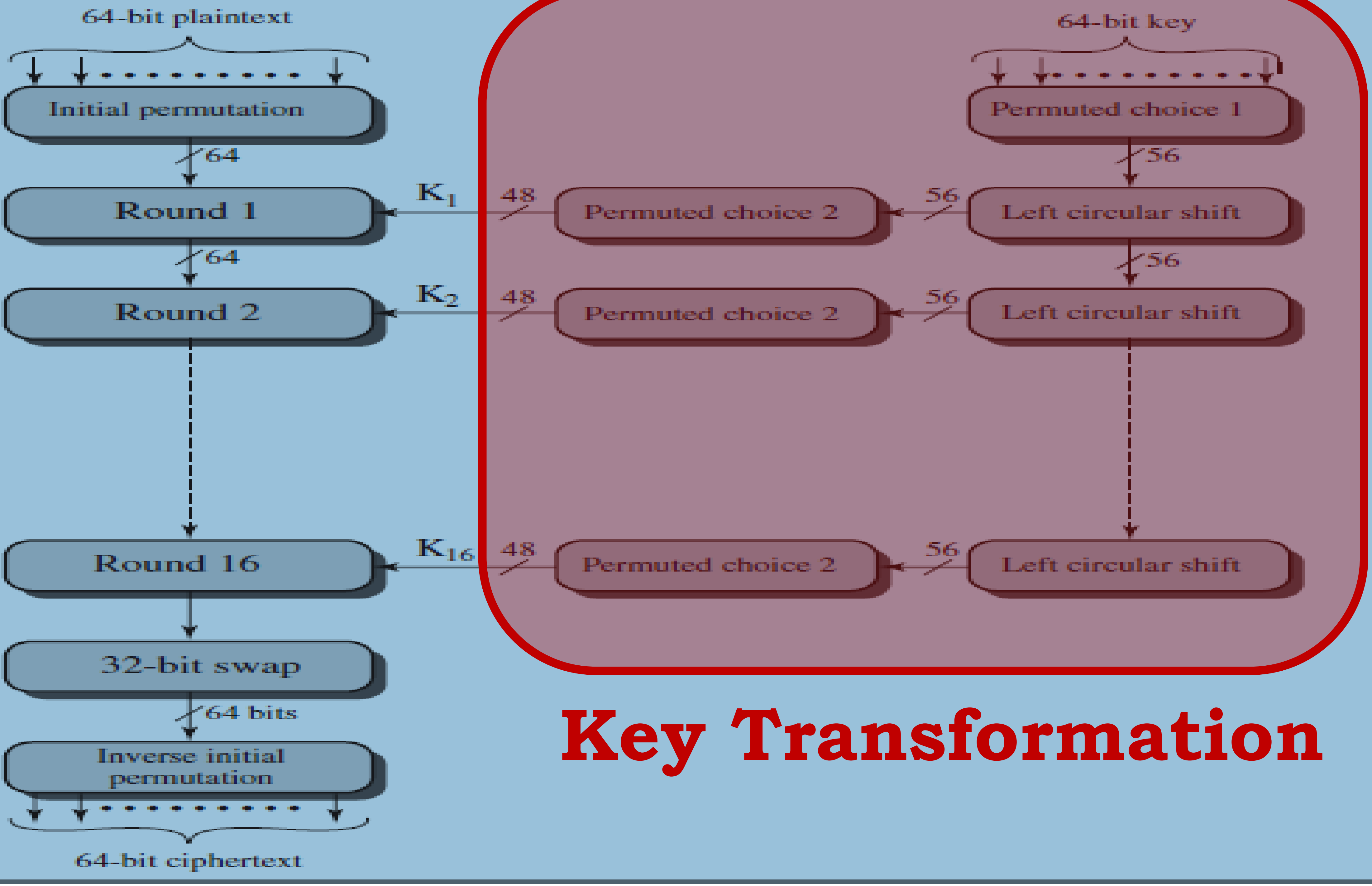
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Permuted output

Observation

- The **Initial Permutation** and the **corresponding Final permutation** do not affect **DES's security.**
  - ✓ The primary purpose is to make it easier to load **plaintext and cipher text data** into a **DES chip** in **byte-sized pieces**.
  - ✓ **Bit-wise permutation** is difficult in **software**

Key Transformation



Key Transformation

Question

- How the **round keys are derived** from the **main encryption key.**

  - The **round keys** are generated from the **main key** by a **sequence** of **permutations.**
  - Each **round key** is **48 bits in length.**

Key Length Reduction

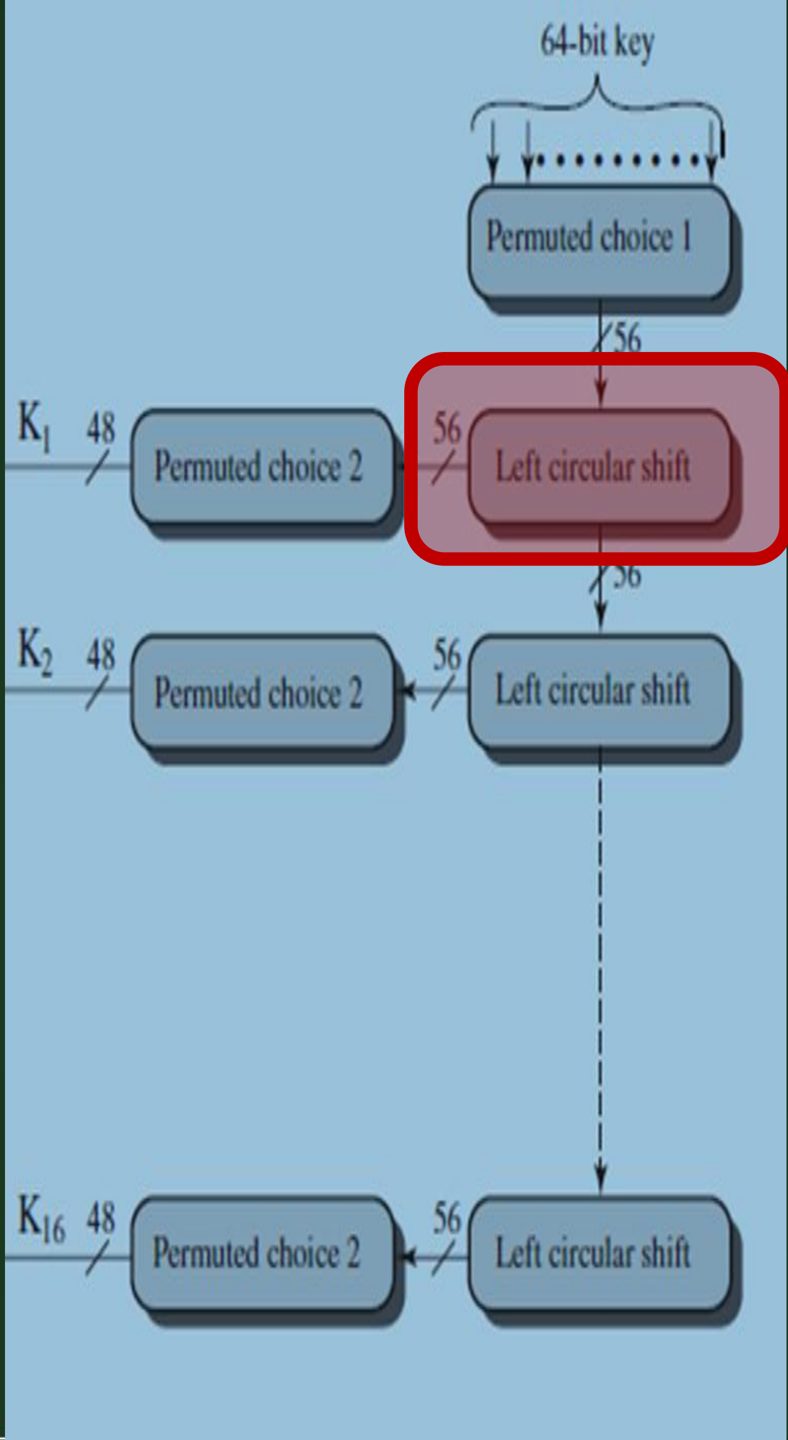
- ❖ The **Right-hand Portion** of **DES Encryption Algorithm** is as Follows.
  - ✓ Initially, the **64-bit DES key** is reduced to a **56-bit key** by **ignoring every eighth bit.**
  - ✓ **Remaining 8 bits** can be used as **parity check** to ensure the **key is error-free.**

Key Generation with Error-free

- We see that a **64-bit key** is used as **input to the algorithm**.
  - ✓ The **bits of the key** are numbered from **1 through 64**; every **eighth bit is ignored**, as indicated by the **lack of shading**.

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

Key Permutation Choice -1



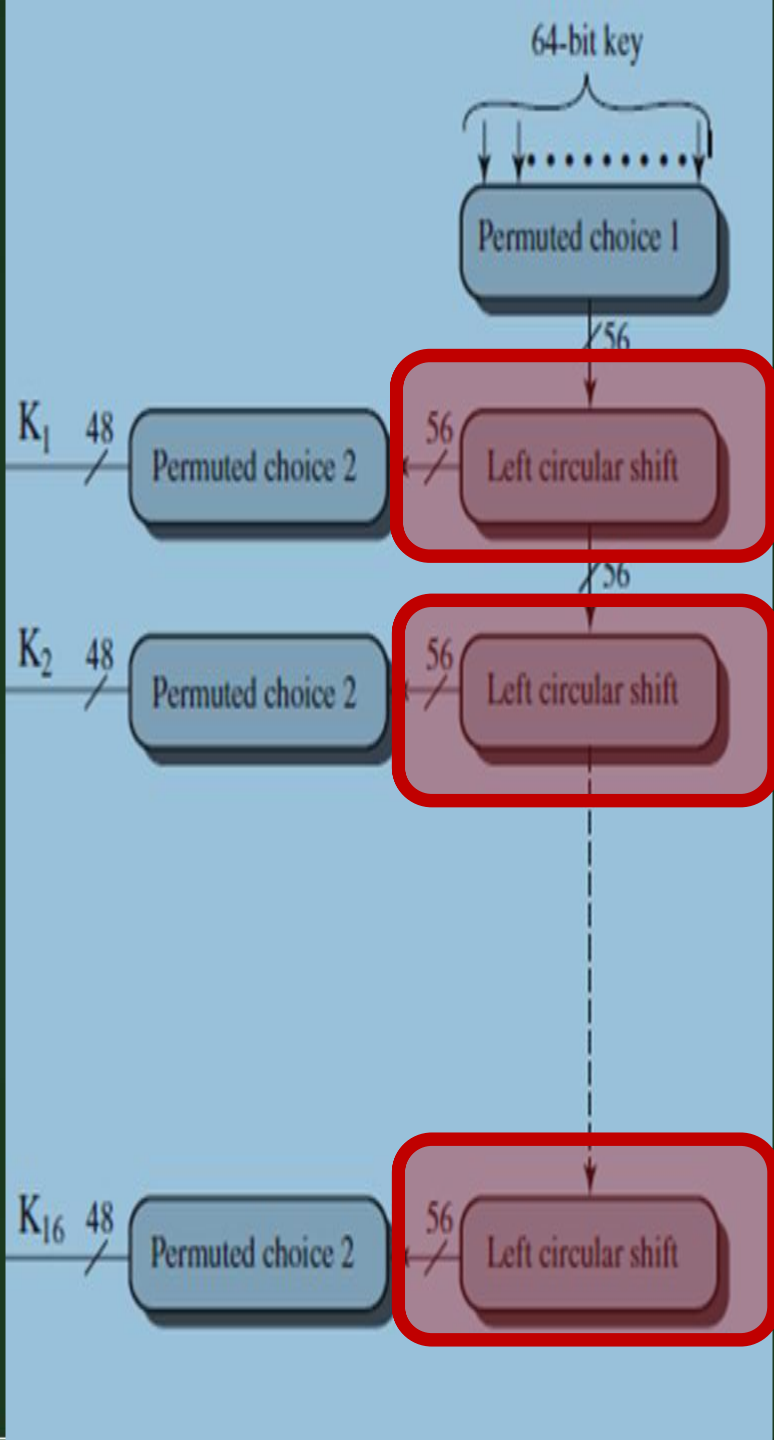
Key Permutation Choice -1

- The **key** is first subjected to a **permutation Choice One**
- The **relevant 56** bits are subject to a **permutation at the beginning** before **any round keys are generated.** This is referred to as **Permutation Choice 1**

Key Permutation

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

Divide the Key into Parts



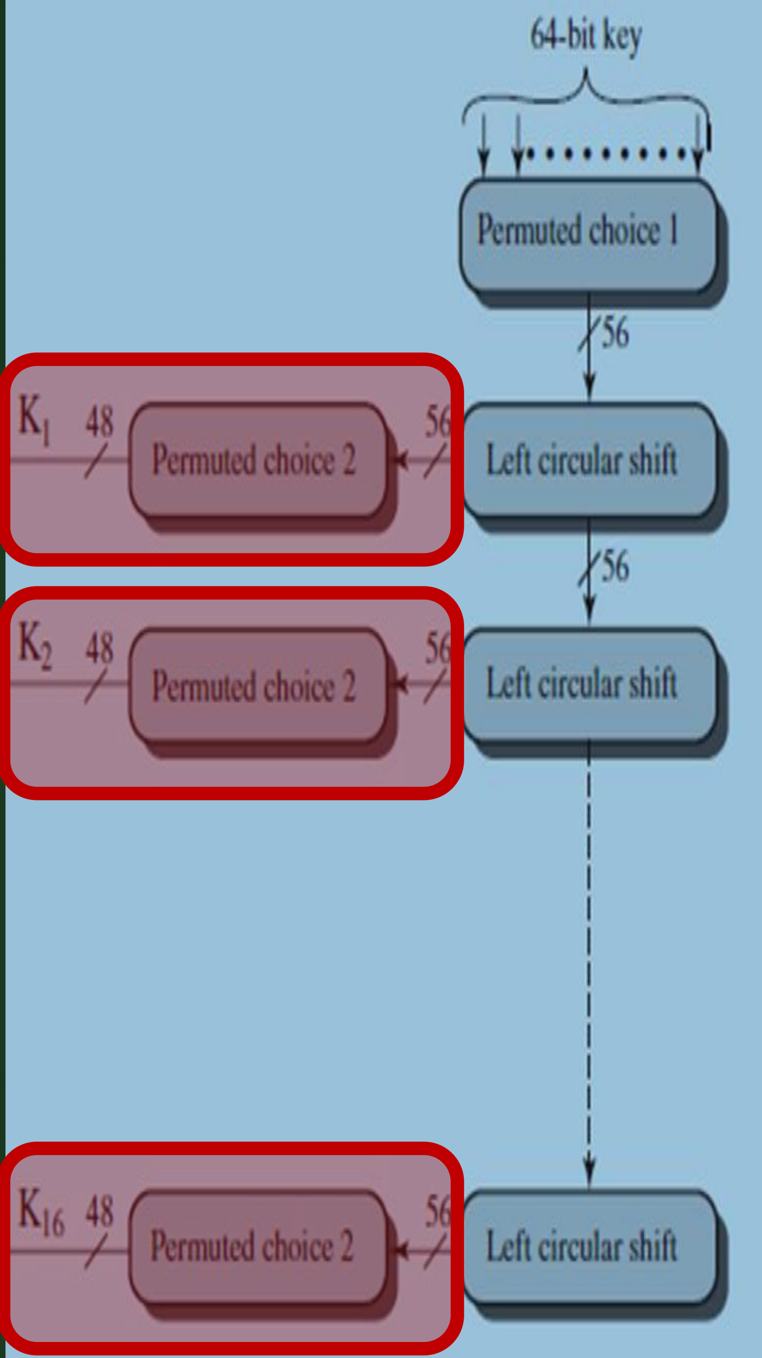
Divide the Key into Parts

- The resulting **56-bit key** is then treated as **two 28-bit quantities**, labeled C_0 and D_0.
- At each round, either **circular left shift** or (**rotation**) of **1 or 2 bits are performed** , as governed by Table

Schedule of Left Shifts

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-2



Permuted Choice-2

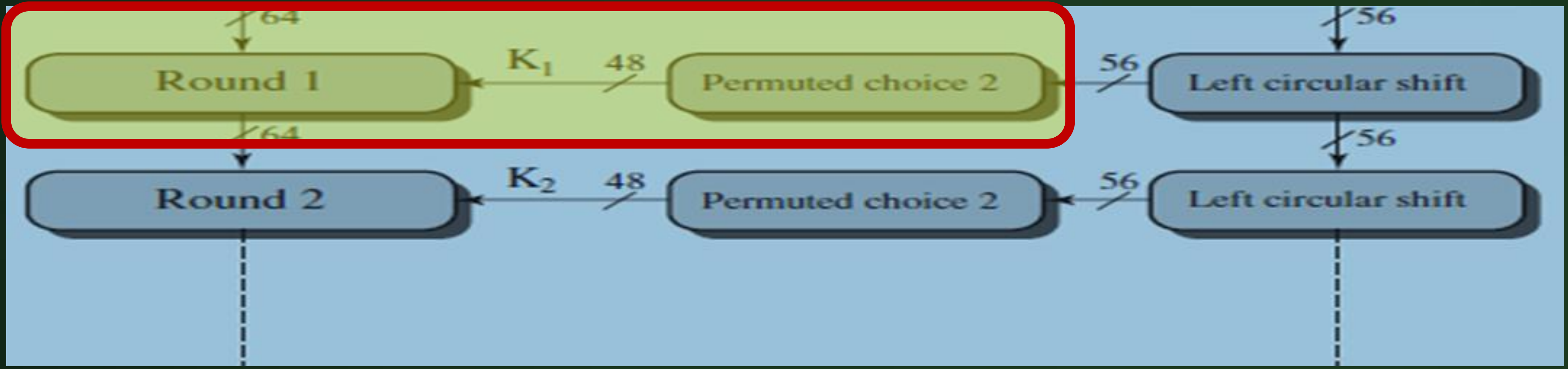
- After the **shifting the values** will serve as **input to the next round.**
- **Shifted Values** also serve as **input to the part labeled Permuted Choice Two** as shown in Table

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

Permuted Choice-2

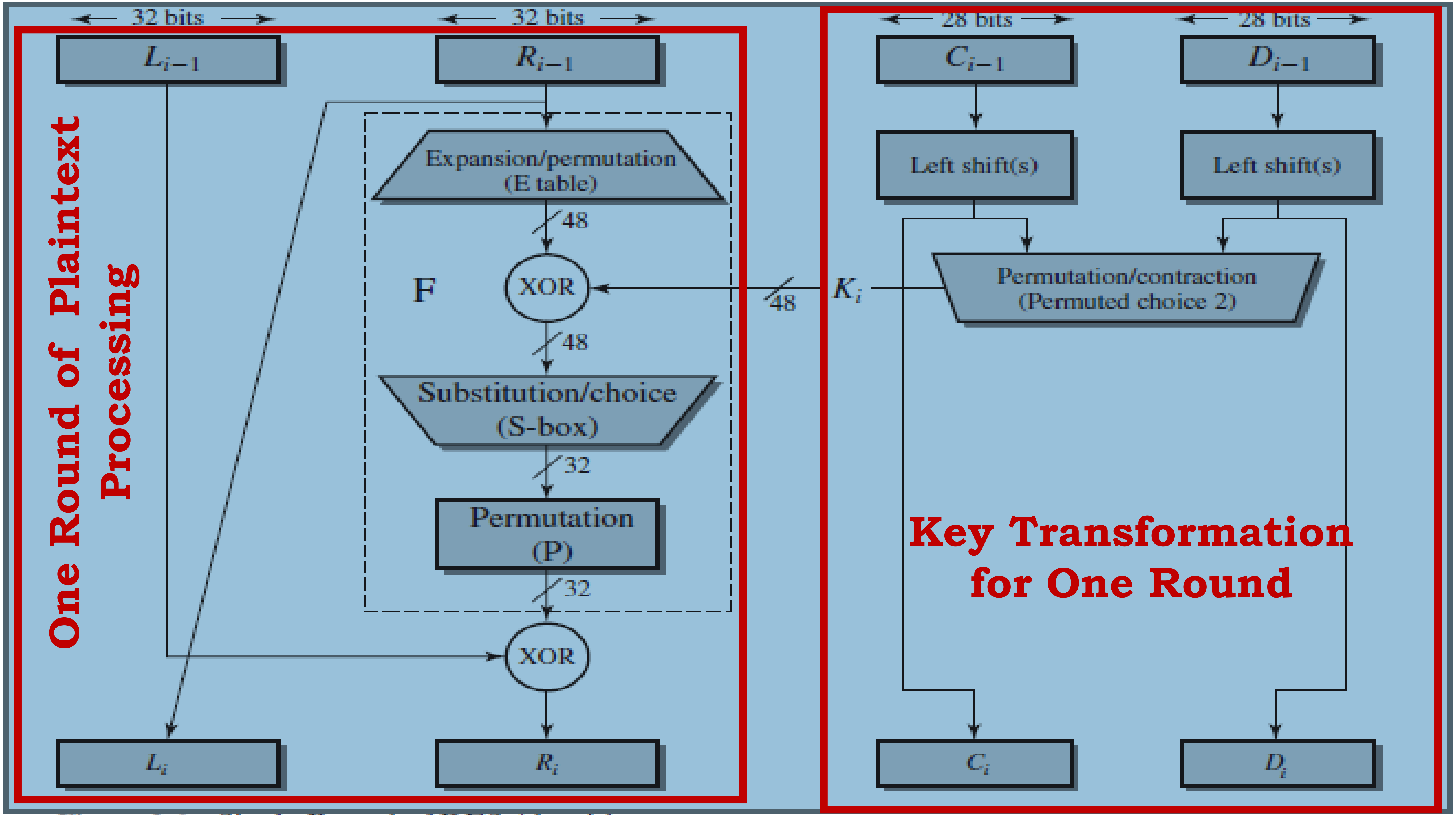
- **Permuted Choice Two** will **56-bit as a input** and produces a **48-bit output**
  - ✓ **48-bit output will** serves as **input to the Round function**.
  - ✓ The **resulting 48 bits** constitute our **round key.**



Observation

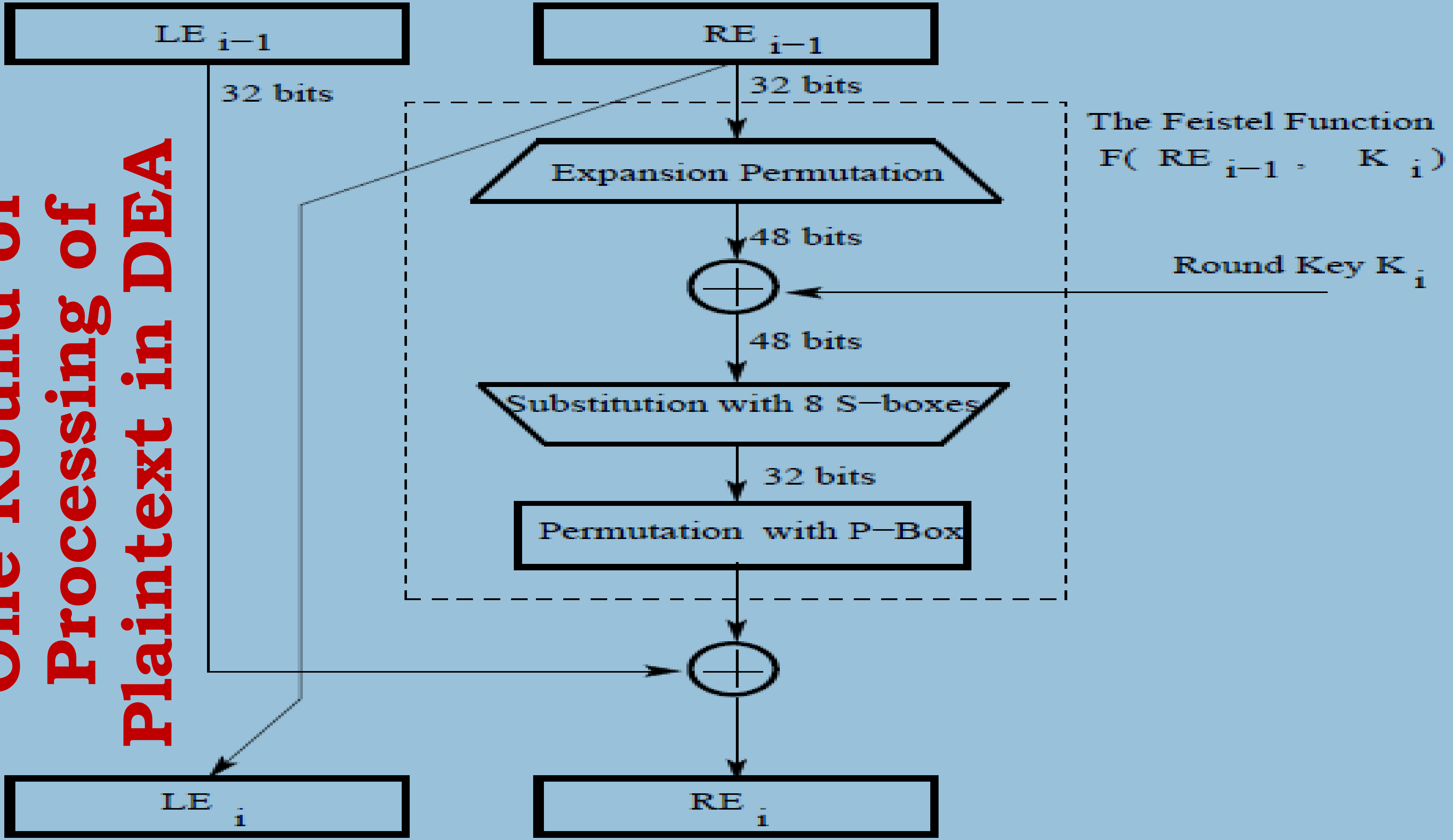
- The **two halves of the encryption key** generated in **each round** are fed as the **two halves** going into the **next round.**

One Round of Processing in DEA



One Round of Plaintext Processing
32 bits
32 bits
L_{i-1}
R_{i-1}
Expansion/permutation (E table)
48
F
XOR
48
Substitution/choice (S-box)
32
Permutation (P)
32
XOR
L_i
R_i
28 bits
28 bits
C_{i-1}
D_{i-1}
Left shift(s)
Left shift(s)
Permutation/contraction (Permuted choice 2)
K_i
48
Key Transformation for One Round
C_i
D_i

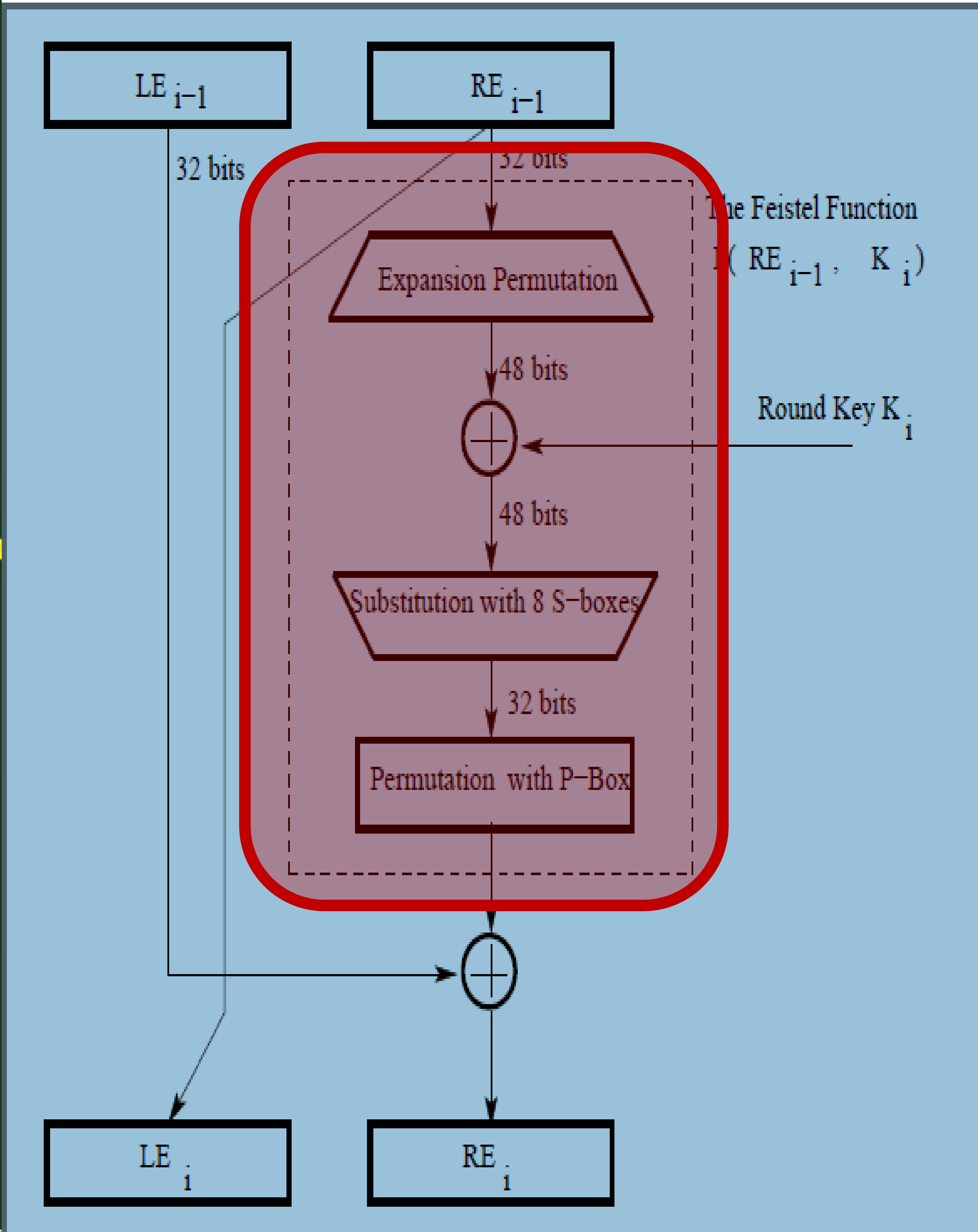
One Round of Processing of Plaintext in DEA



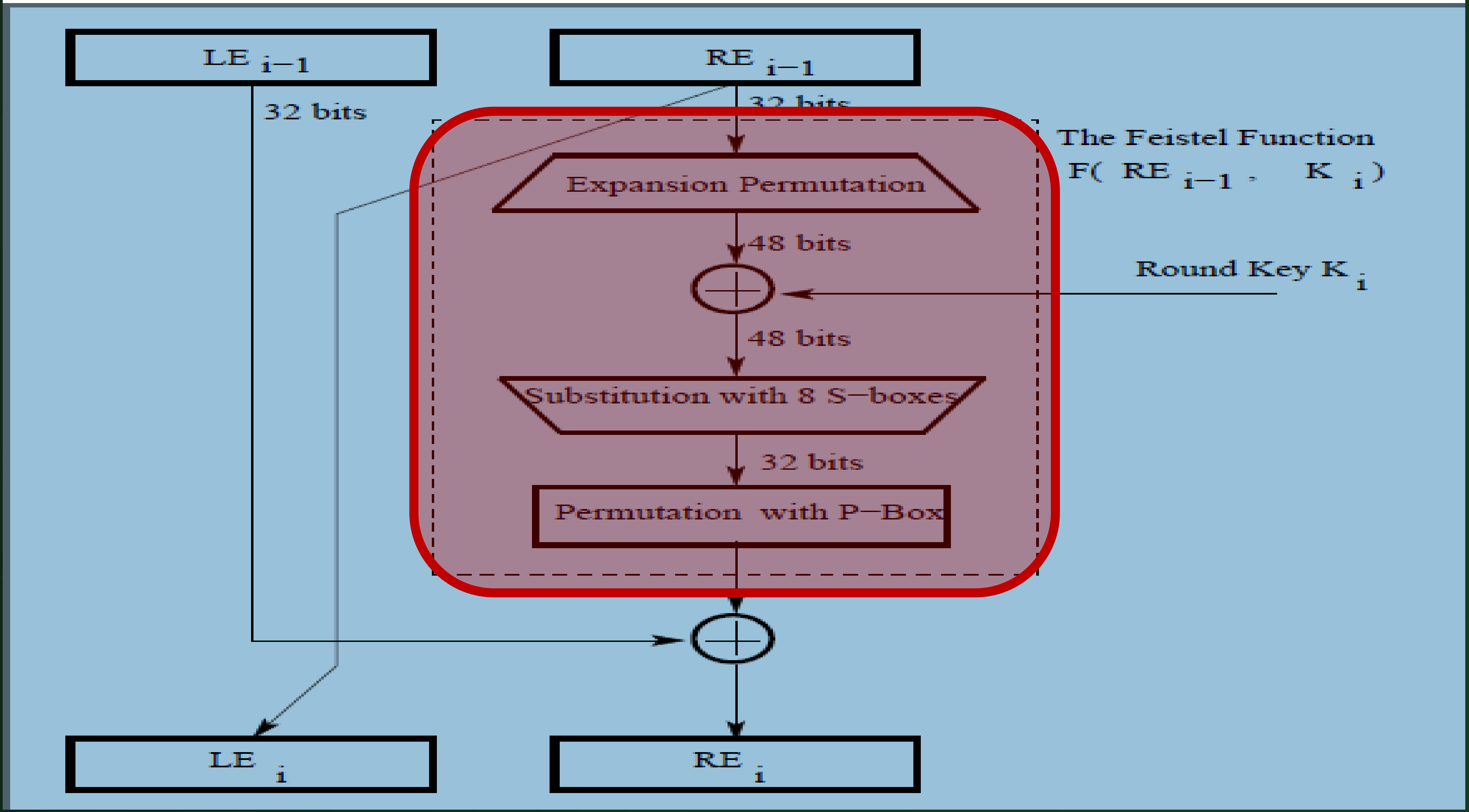
Data Encryption Algorithm.

- The **Algorithmic Implementation of DES** is known as **DEA for Data Encryption Algorithm.**

Round Function F

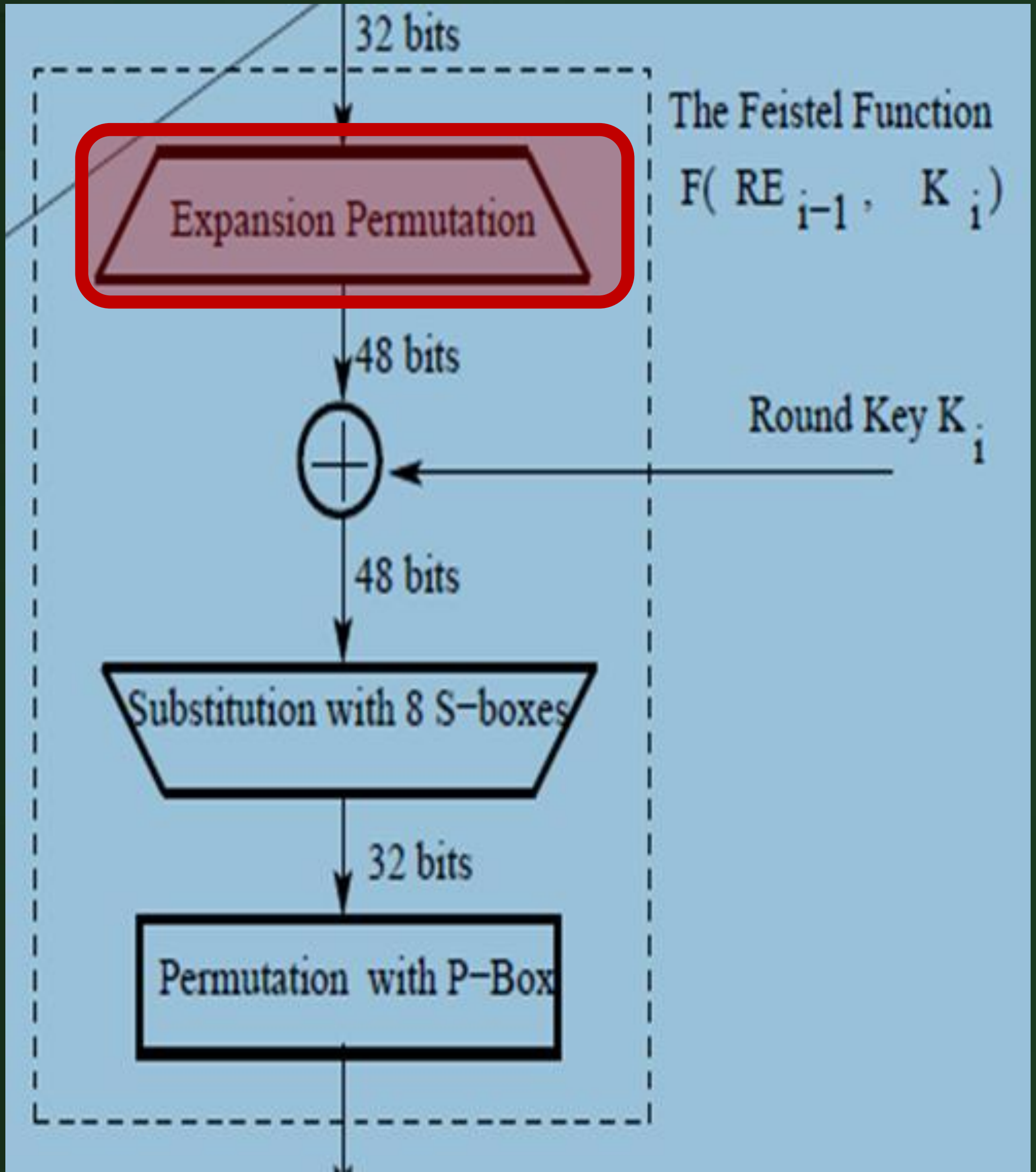


Round Function F



Expansion Permutation Step

Expansion Permutation Step



Expansion Permutation Step

❖ The **32-bit right half** of the **64-bit input data block** is **expanded** by into a **48-bit block**.

✓ This is referred to as the **Expansion Permutation Step, Or The E-step.**

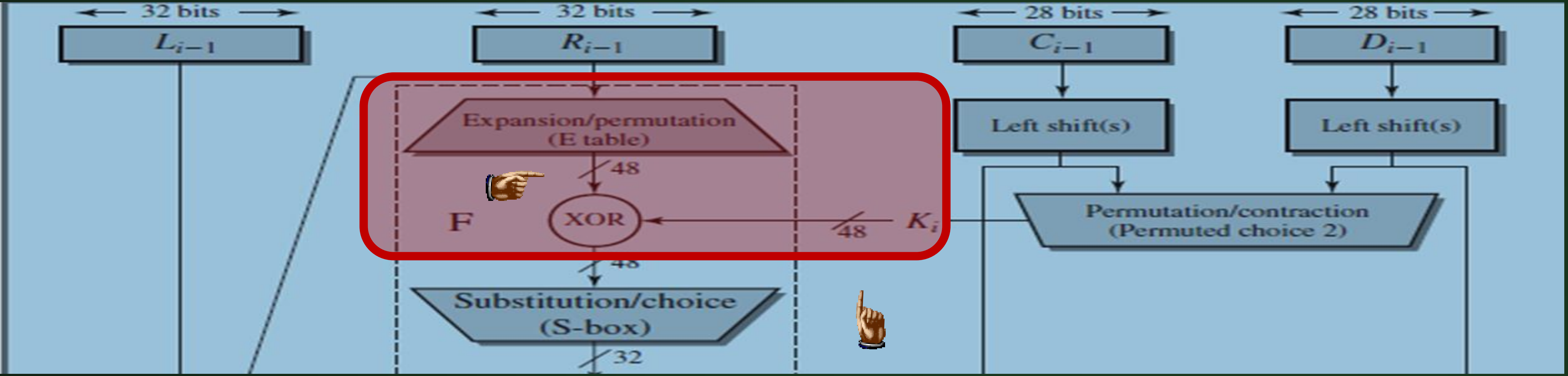
32-Bit Input

Expansion Permutation (E)

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

48-Bit Input

Purpose of E-Step



- ✓ The **same size as the key**
- ✓ **Longer result** that can be **compressed** during the **substitution operation.**

Purpose of E-Step

- This **operation changes** the **order of the bits** as well as **repeating certain bits**
- This operation has **two purposes**:
  - ✓ It makes the **right half** the **same size as the key** for the **XOR operation**
  - ✓ It provides a **longer result** that can be **compressed** during the **substitution operation.**

Working Process of the E-step.

- The **E-step** does the following:

1. First divide the **32-bit block** into **eight 4-bit words**
2. Attach an **additional bit on the left** to each **4-bit word** (i.e the **last bit of the previous 4-bit word**)
3. Attach an **additional bit** to the **right of each 4-bit word** (i.e the **beginning bit of the next 4-bit word.**)

Working Process of the E-step.

Expansion Permutation (E)

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

Construction of E-step

- When we examine the **expansion table**,
  - ✓ The **32 bits of input** are **split into groups of 4 bits** then
  - ✓ It will become **groups of 6 bits** by taking the **outer bits** from the **two adjacent groups.**

The E-step : Example

- For example, if part of the **input word** is

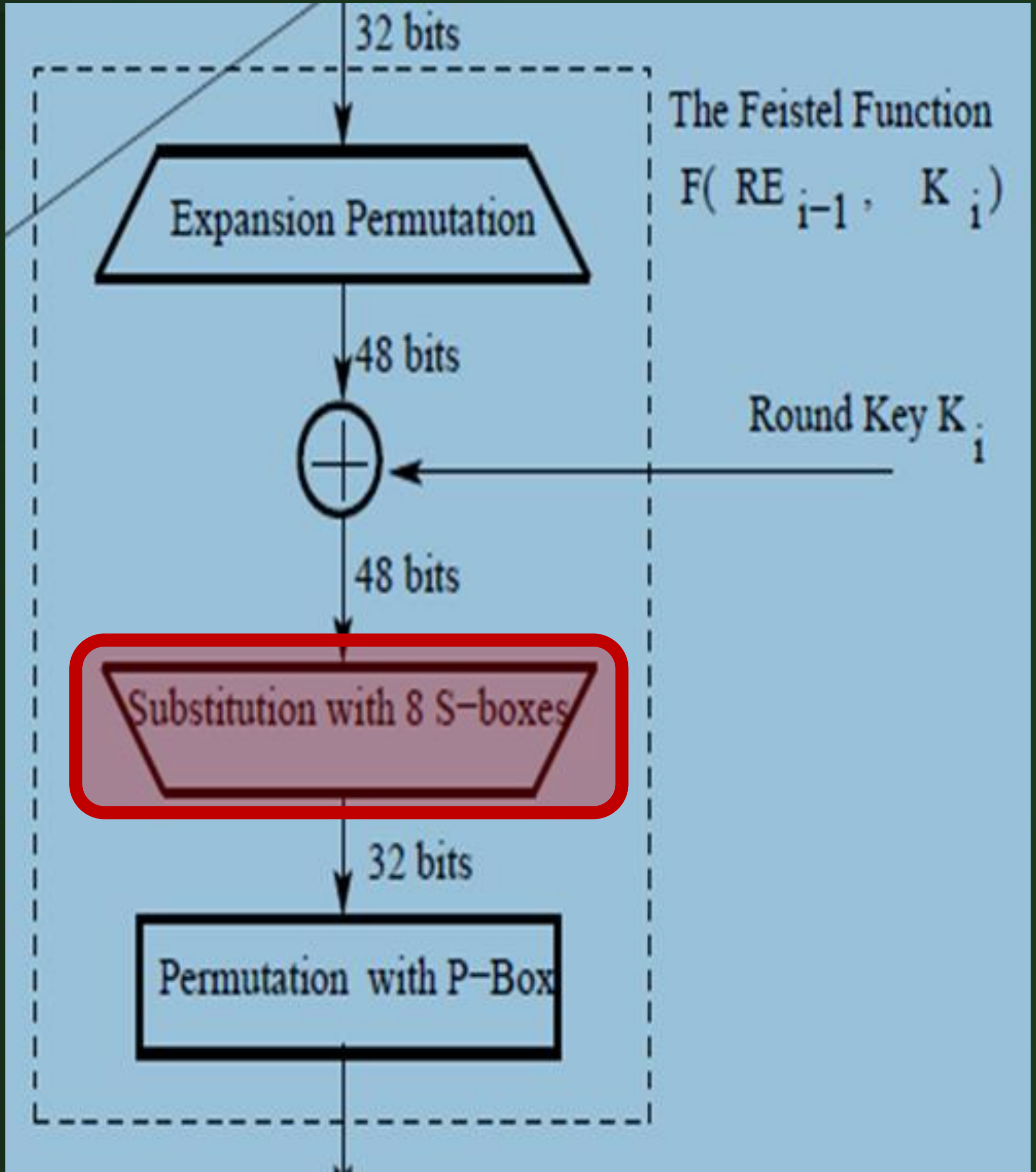
  … efgh ijkl mnop …

  this becomes

  … defghi hijklm lmnopq …

Substitution Boxes (S-boxes)

Substitution Boxes (S-boxes)



Key Mixing.

- The **48 bits** of the **Expanded Output** produced by the **E-step** are **XORed** with the **round key**.
  - ✓ This is referred to as **KEY MIXING.**

Substitution Boxes (S-boxes)

- The **output** of **E-Step** is broken into **Eight Six-bit Words**.
  - ✓ Each of which **accepts 6 bits as input** and produces **4 bits as output**.
  - ✓ The **substitution** is carried out with an **S-box**

Purpose of S-Box Step in Each Round

- The goal of the **substitution step** is to introduce **DIFFUSION** in the **generation of the output** from the **input.**
  - ✓ **Diffusion** means that each **plaintext bit** must **affect as many ciphertext bits** as possible.

Confusion is provided with Different Round Keys

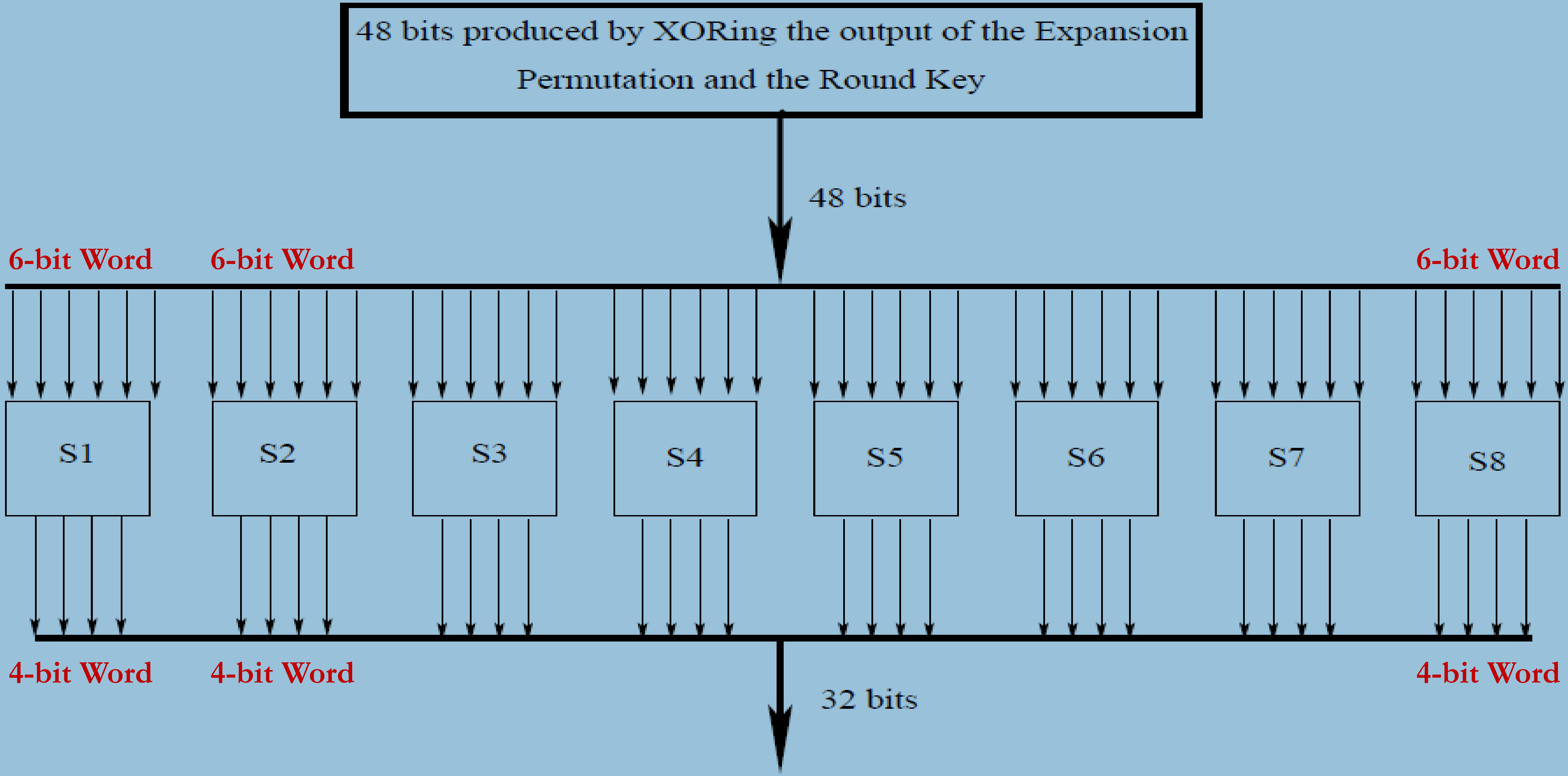
- The main strategy used for **creating the different round keys** from the **main encryption key**
  - ✓ To introduce **CONFUSION into the encryption process.**

What is Confusion?

- ❖ **Confusion** is that the **relationship between** the **encryption key** and the **cipher text** must be as **complex as possible.**

- ❖ **Confusion** would be that each bit of the **key must affect as many bits** as possible of the **output cipher text block.**

The S-Box Step in Each Round



Working Process of S-Box Step in Each Round

- From the Figure , the **48-bit input word** is divided into **eight 6-bit words** and
  - ✓ Each **6-bit word** fed into a **separate S-box**.
  - ✓ Each **S-box** produces a **4-bit output.**
  - ✓ Therefore, the **8 S-boxes together** generate a **32-bit output.**
- The **overall substitution step** takes the **48-bit input** back to a **32-bit output.**

S-Box Table in the Substitution Step

- Each of the **eight S-boxes** consists of a **4×16 table** lookup for an output **4-bit word**.
  - ✓ The **first and the last bit** of the **6-bit input** word are decoded into **one of 4 rows** and
  - ✓ The **middle 4 bits decoded** into **one of 16 columns** for the table lookup.

Finally : Usage of 8 S-Box in the Substitution Step

- Thus, the **row lookup** for each of the **eight S-boxes** becomes a **function of the input bits** for the **previous S-box** and the **next S-box.**

For Example

- Suppose the **input 011001** for the **table S1,**
  - ✓ The row is **01** (**row 1**) and
  - ✓ The column is **1100** (**column 12**).
  - ✓ The value in **row 1**, **column 12** is **9**, so the **output is 1001.**

S_1	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

Substitution(S-Boxes) Table

The 4 × 16 substitution table for S-box S_1

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 S_2

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

S-box S_3

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

S-box S_4

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

S-box S_5

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

S-box S_6

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

S-box S_7

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

S-box S_8

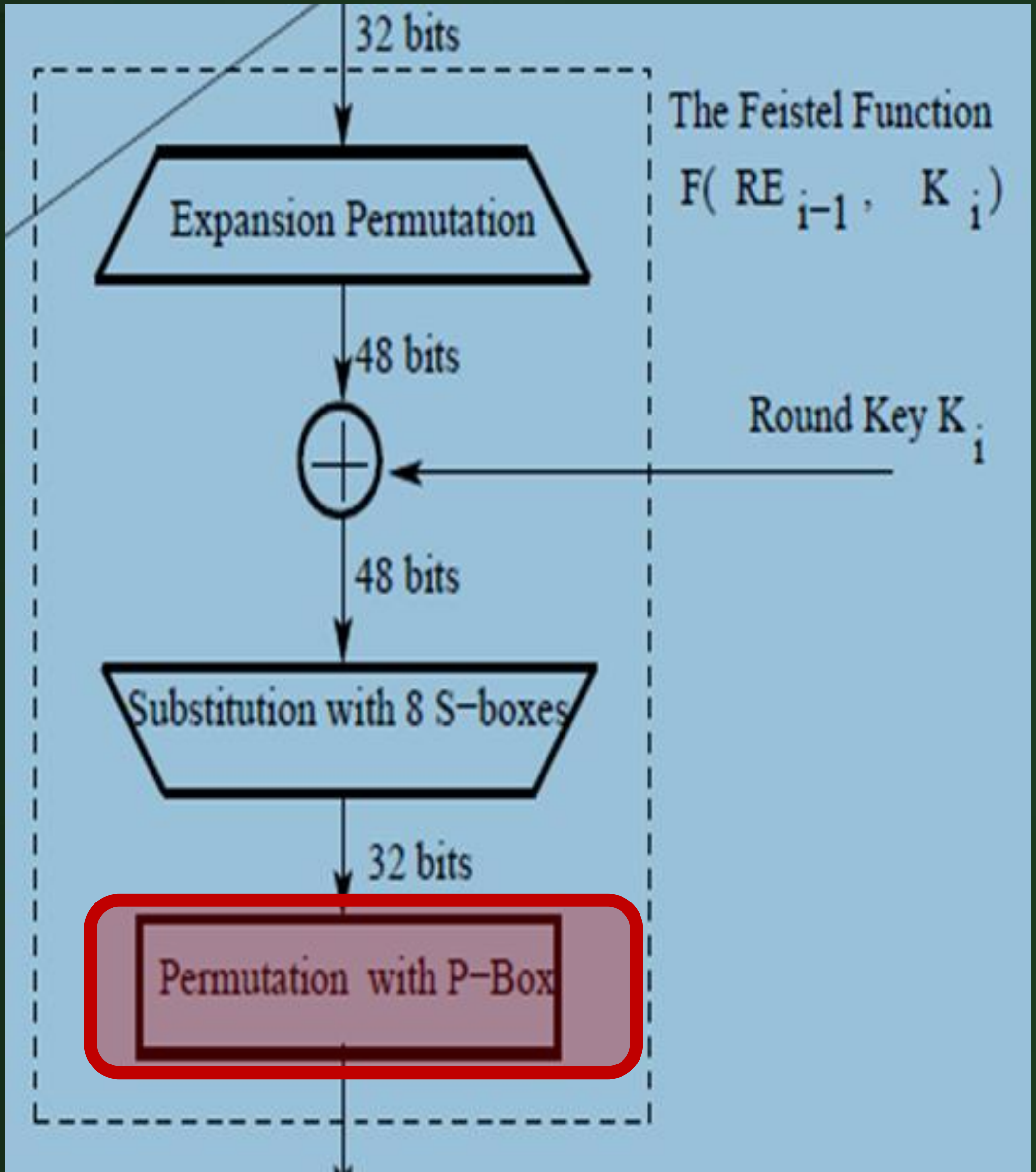
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Substitution Table

- **Eight S-boxes**, **S1 through S8**
  - ✓ Each **S-box being a 4×16 substitution table** that is used to convert **6 incoming bits** into **4 outgoing bits.**

Permutation Step or P-Box

Substitution Boxes (S-boxes)



The P-Box Permutation Step in F Function

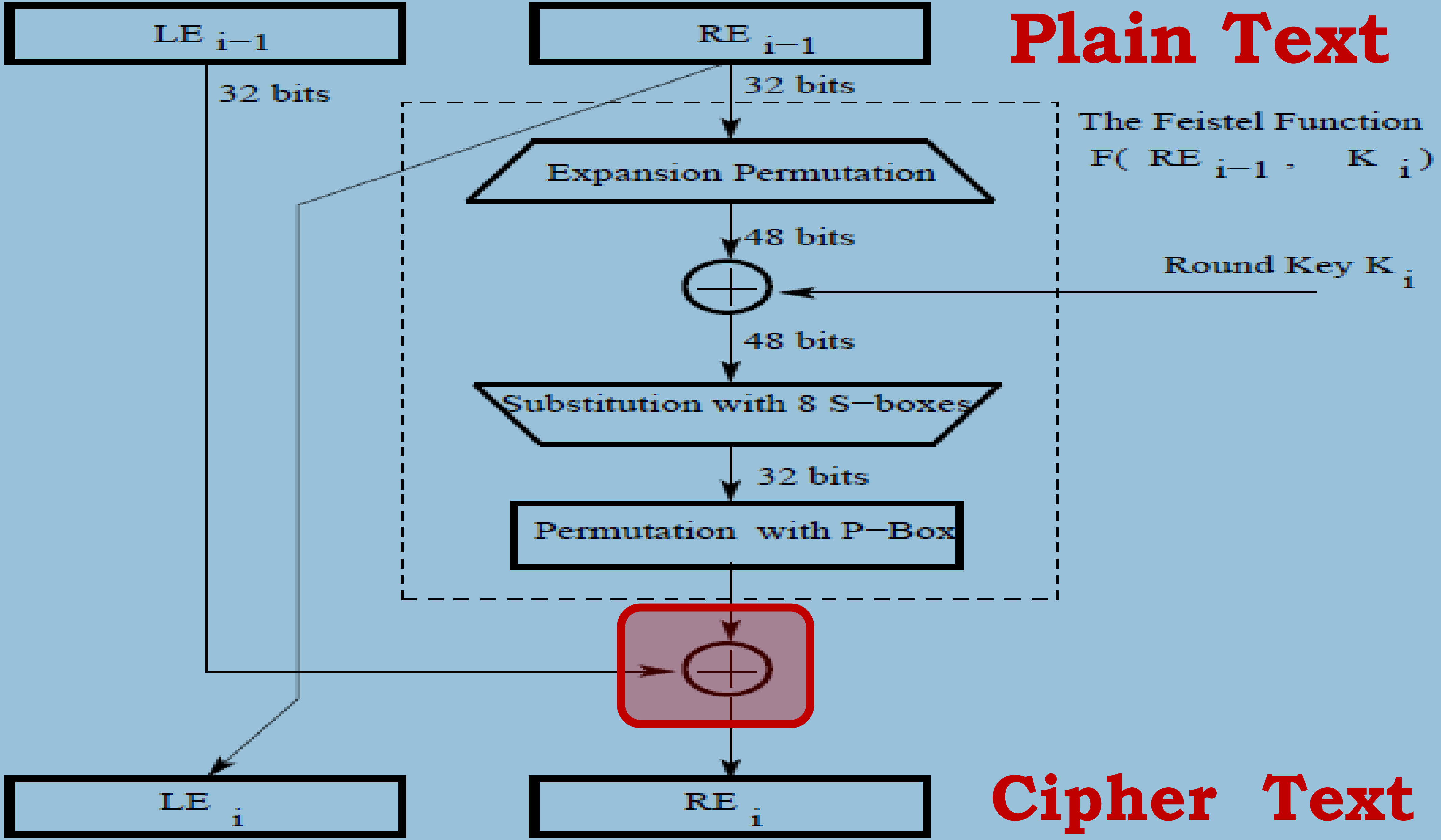
❖ The **last step in the Feistel function** is "**Permutation with P-Box**". The permutation table is shown below.

P-Box Permutation							
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

Purpose of P-Box Step

- The **permutation table** says
  - ✓ The **first output bit will be the 16th bit of the input**
  - ✓ The **second output bit the 7th bit of the input,** and so on, for all of the **32 bits of the output** that are obtained from the **32 bits of the input**.
- ✓ Note that **bit indexing starts with 1 and not with 0.**

DES Encryption Algorithm for One Round of Processing


LE i−1
RE i−1
Plain Text
32 bits
32 bits
The Feistel Function
F(RE i−1 , K i)
Expansion Permutation
48 bits
Round Key K i
48 bits
Substitution with 8 S−boxes
32 bits
Permutation with P−Box
LE i
RE i
Cipher Text

Decryption of DES

Decrypting DES

- After performing all the **substitutions, permutations, XORs,** and **shifting** around encryption algorithm
- One might think that the **decryption algorithm is completely different** and just as **confusing as the encryption algorithm**.
- With **DES** it is possible to **use the same function** to **encrypt or decrypt a block**.

Bird View : Decrypting DES

- The only difference is that the **keys must be used in the reverse order.**
  - ✓ If the **encryption keys** for each round are K_1 K_2 K_3,…, K_{16} then the **decryption keys** are K_{16} K_{15} K_{14}, …, K_1.

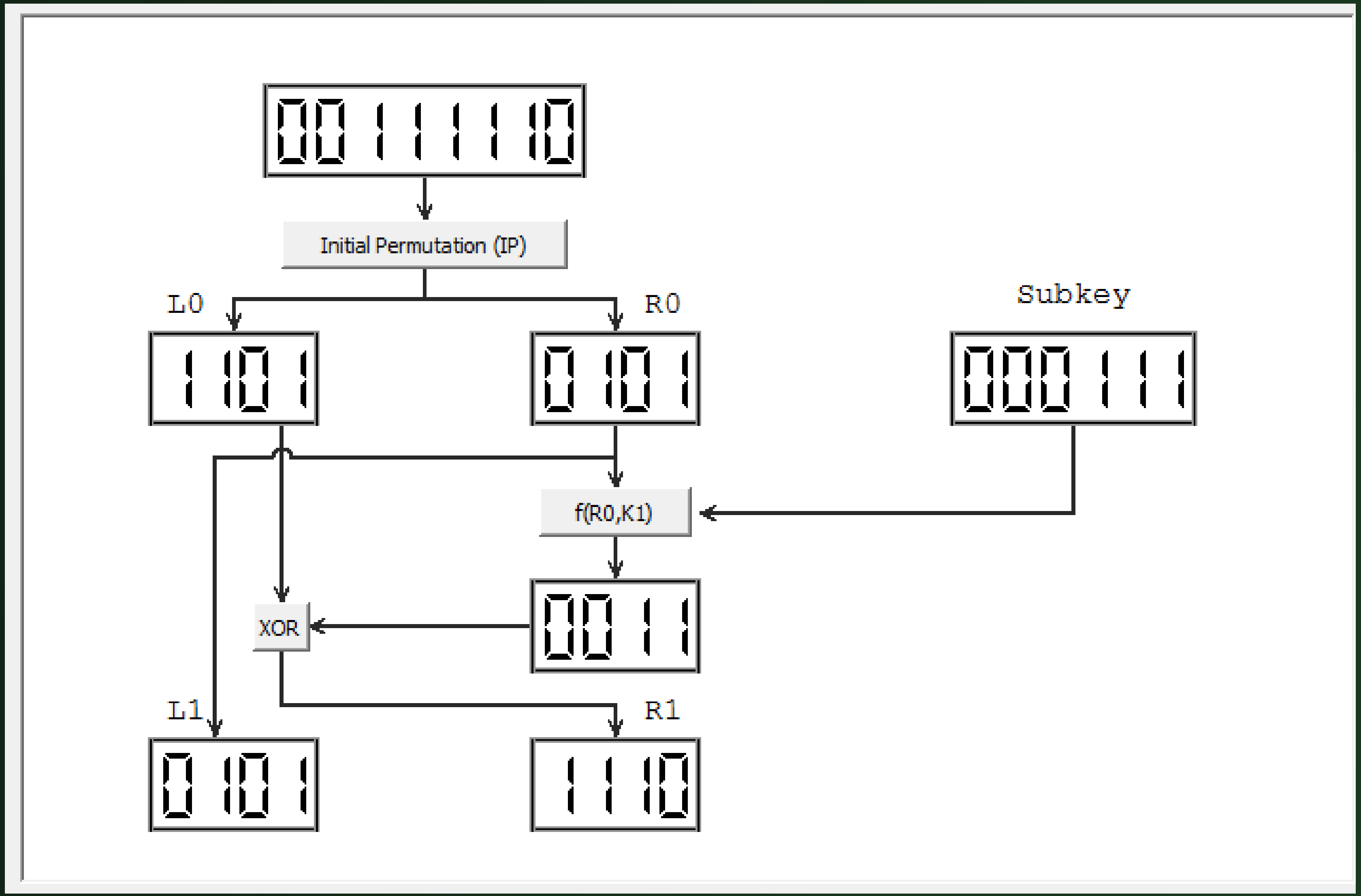
Bird View : Decrypting DES

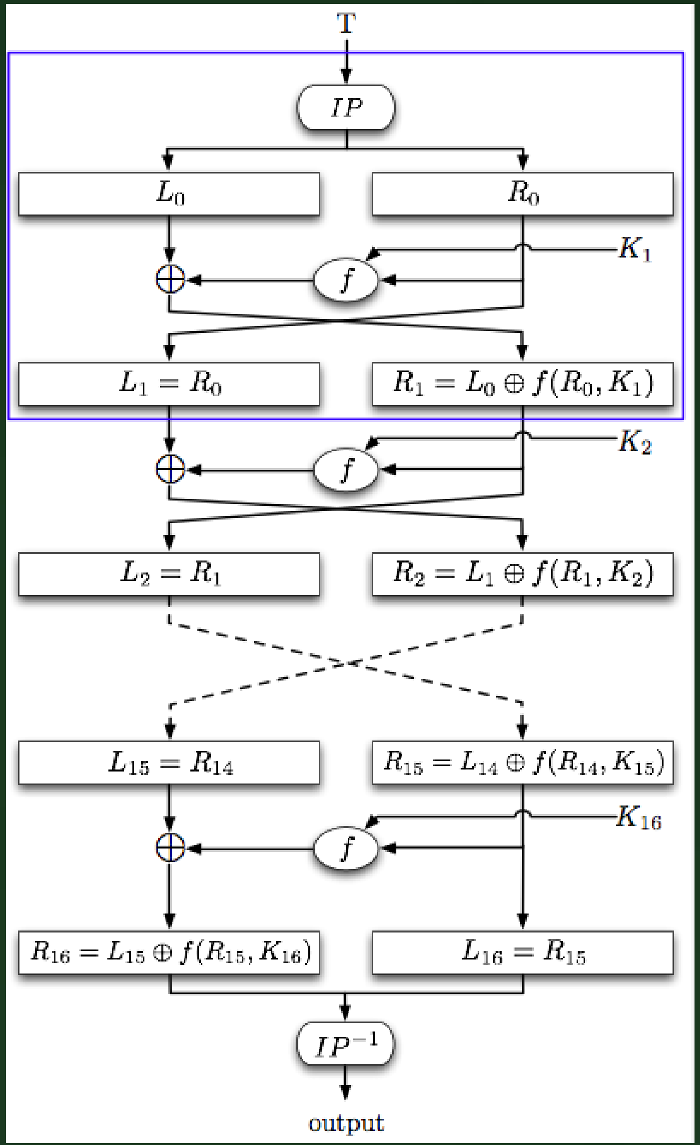
- The algorithm that **generates the key** used for **each round** is **circular as well.**
  - ✓ The **key shift** is a **right shift** and the **number of positions shifted** is **0 , 1 , 2 , 2 , 2 , 2 , 2 , 2 , 1 , 2 , 2 , 2 , 2 , 2, 2 , 1.**

DES Example

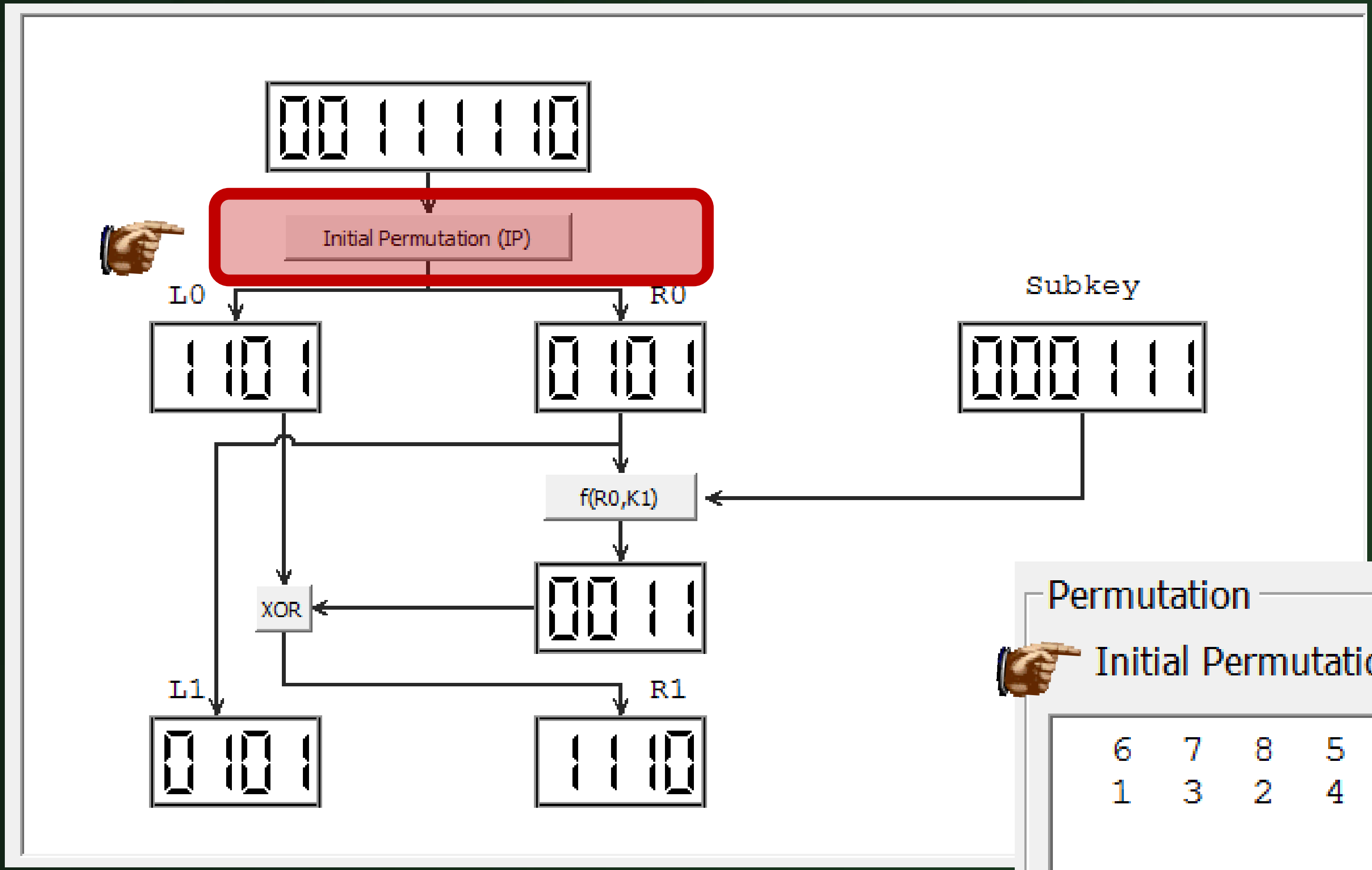
DES Example: Encryption

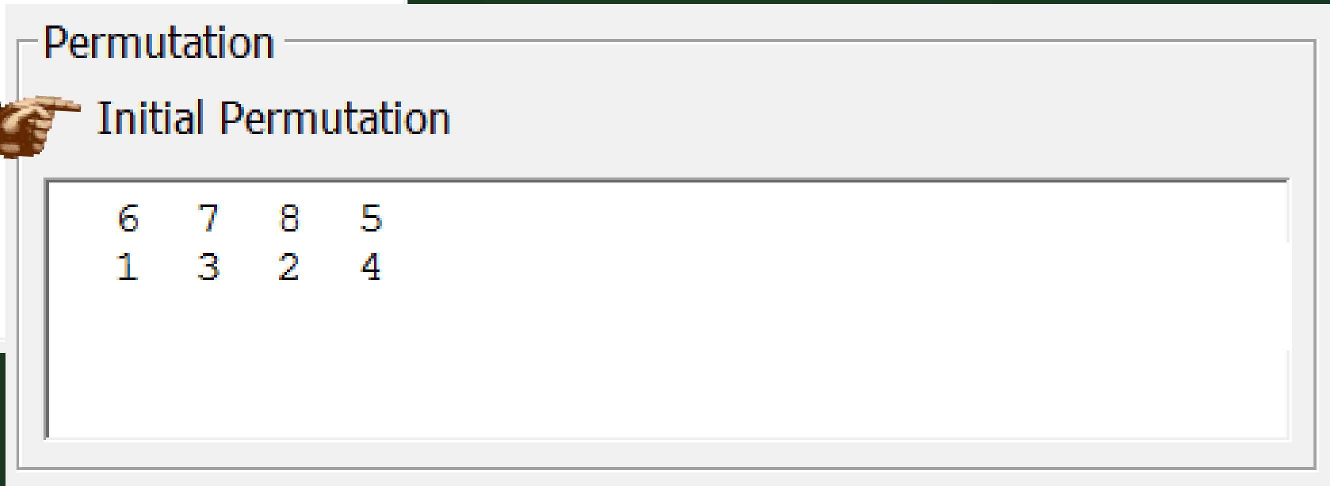
DES Example (8-Bit) Plain Text: 64 ; Key: 7



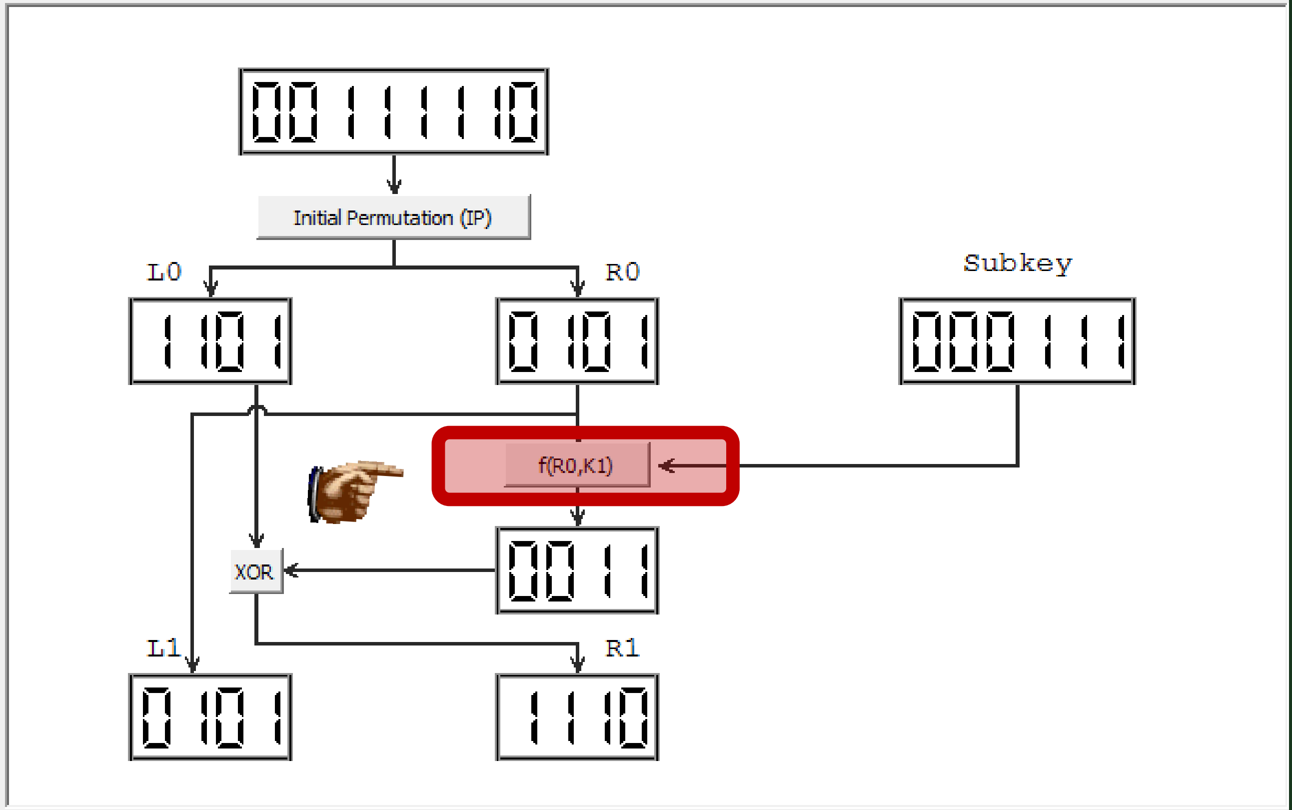


Initial Permutation

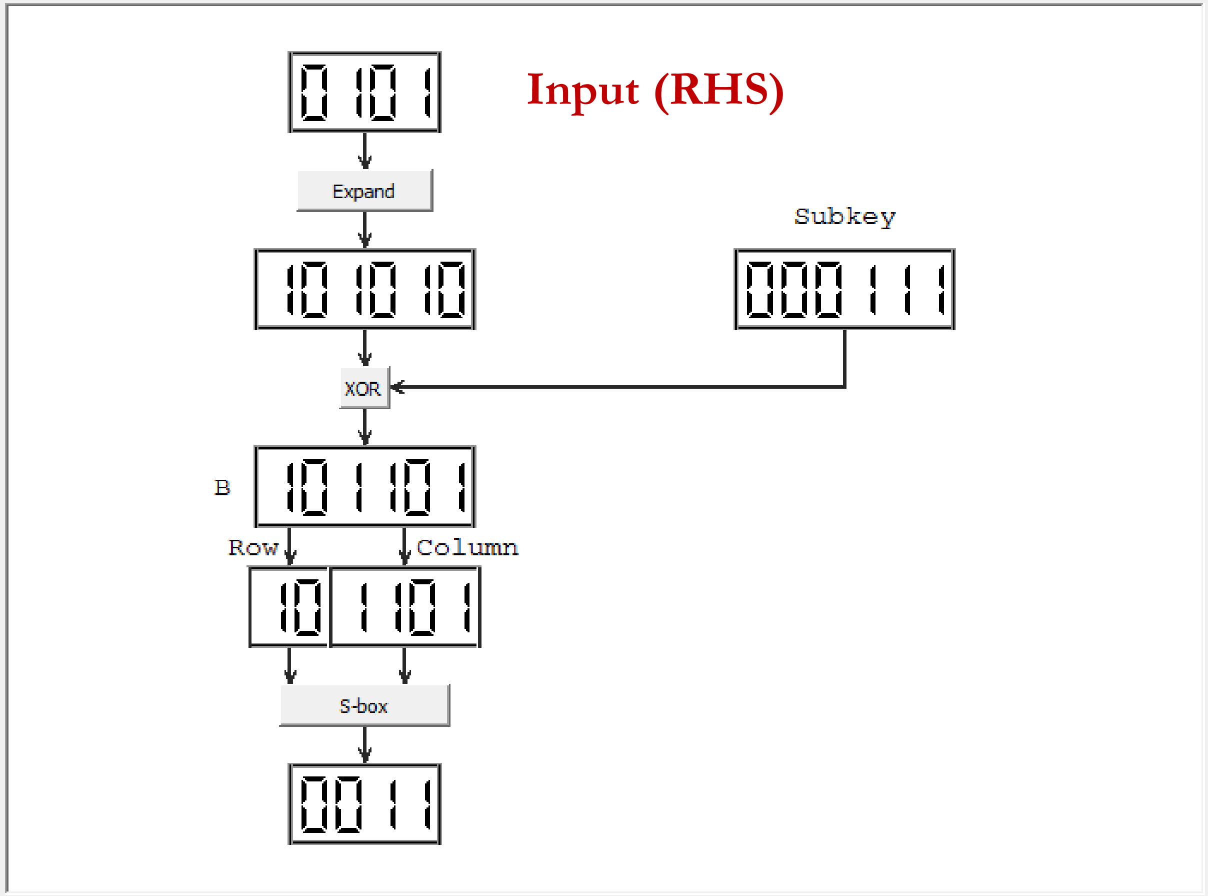




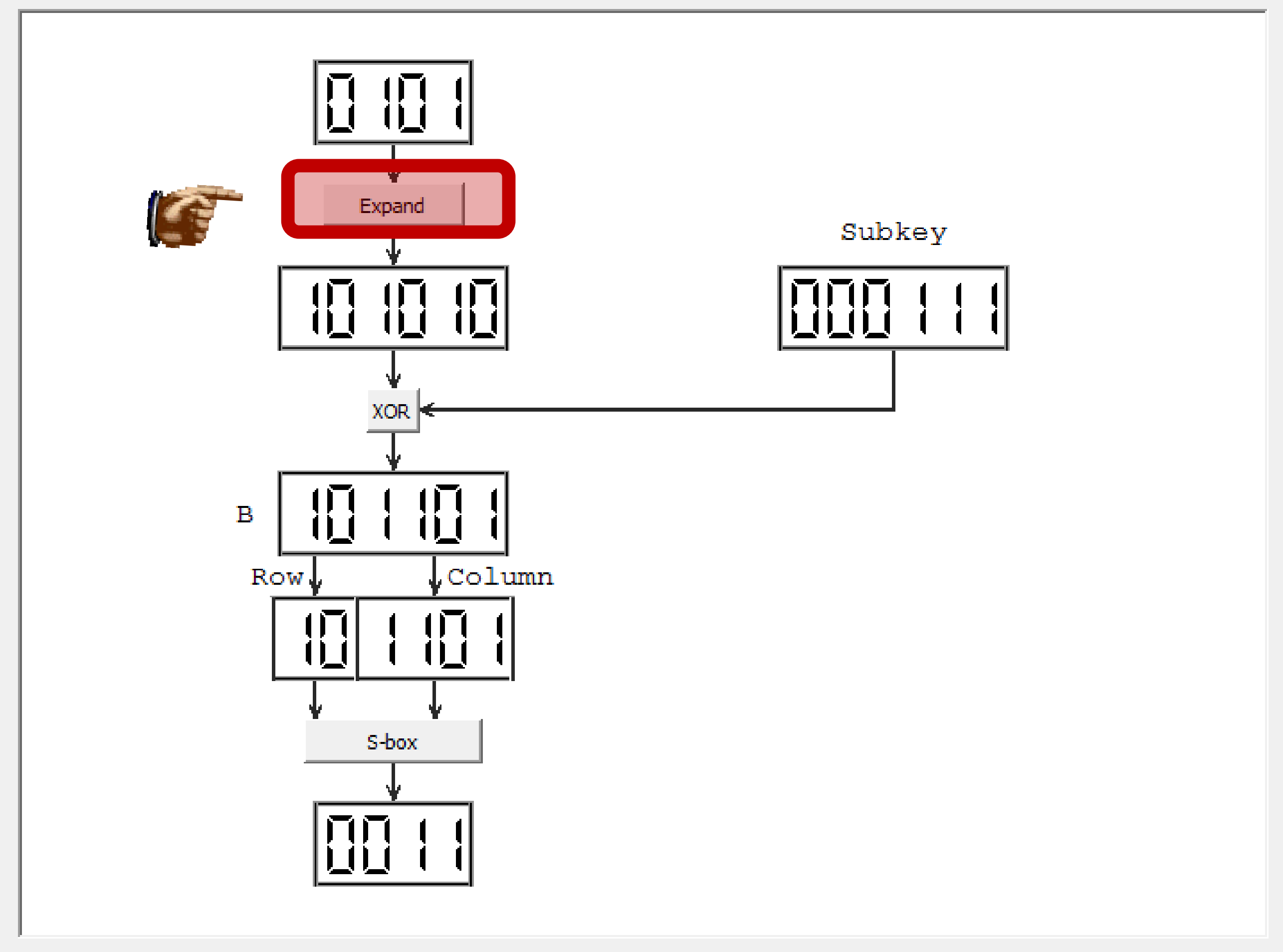
Function: F(R0,K1)



F(R0,K1)



Function: Expand



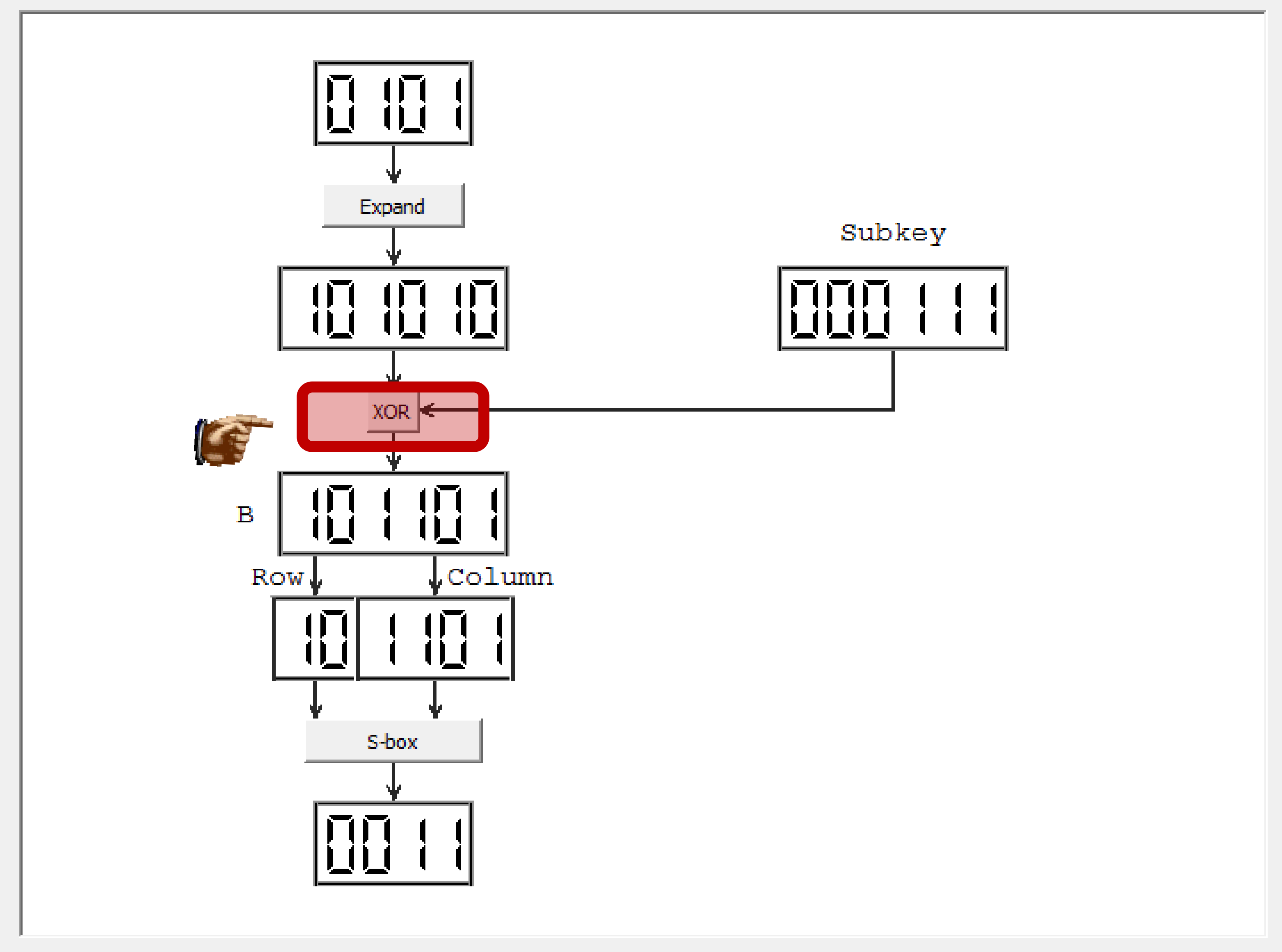
f Function

This table determines how the 4-bit data block expands to 6-bit.

Expand Table

4 1 2 3 4 1

Function: XOR



Function: S-BOX

f Function

S-Box is a 4x16 table, in which each cell is a 4-bit data block.

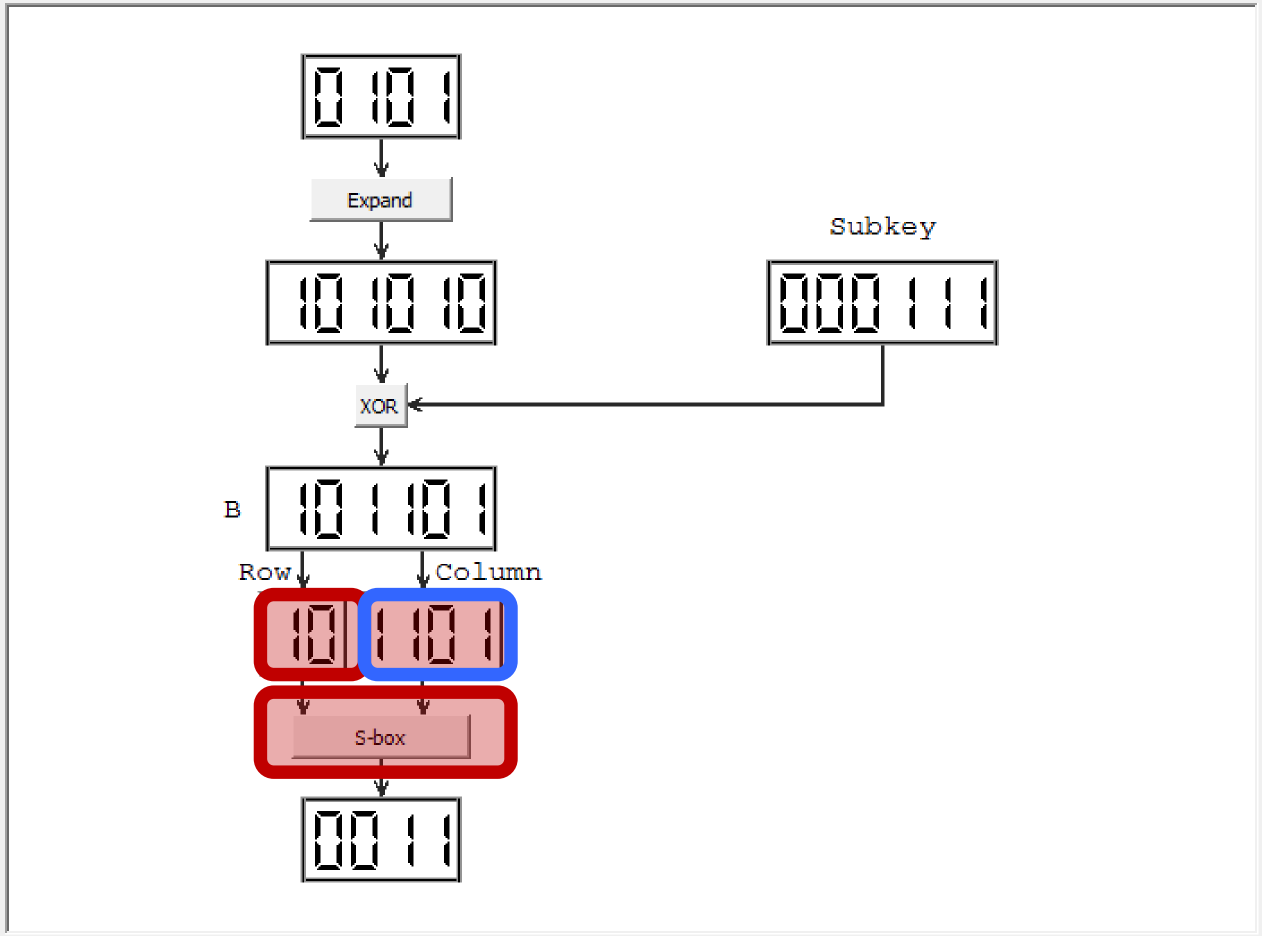
S-box:

Column

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	2	8	12	6	10	14	9	3	7	13	4	15	11	1	0	5	
1	7	2	15	5	8	1	0	14	6	4	13	12	11	9	3	10	S1
2	9	7	1	14	4	13	2	10	8	6	11	5	12	3	0	15	
3	14	6	7	9	2	3	11	4	15	12	0	10	13	5	8	1	

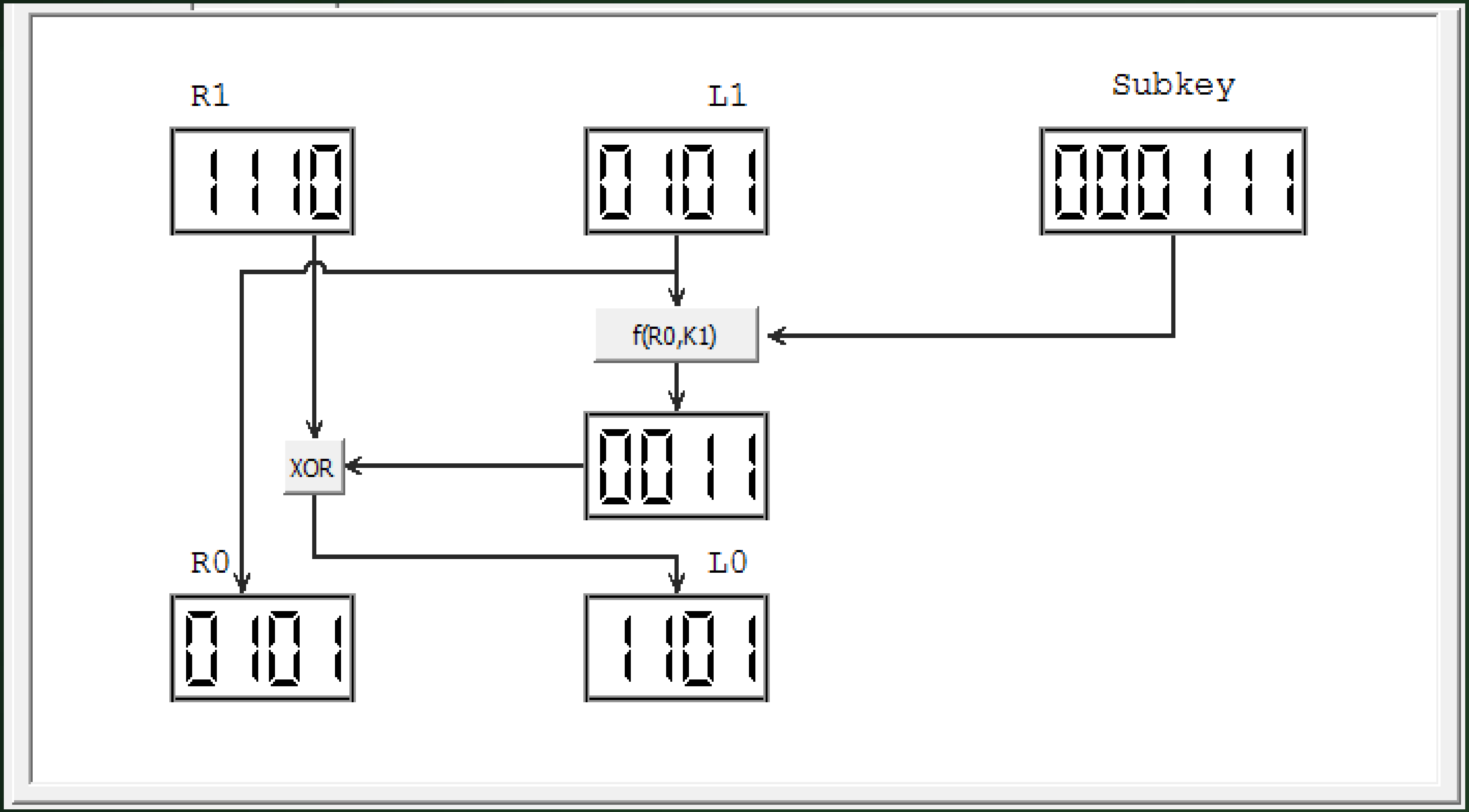
Calculation:

Row1:(10b): 2
Column1:(1101b): 13
Return Value in S-Box 1 at row 2, and col 13: 3
Represent this value in binary: 0011



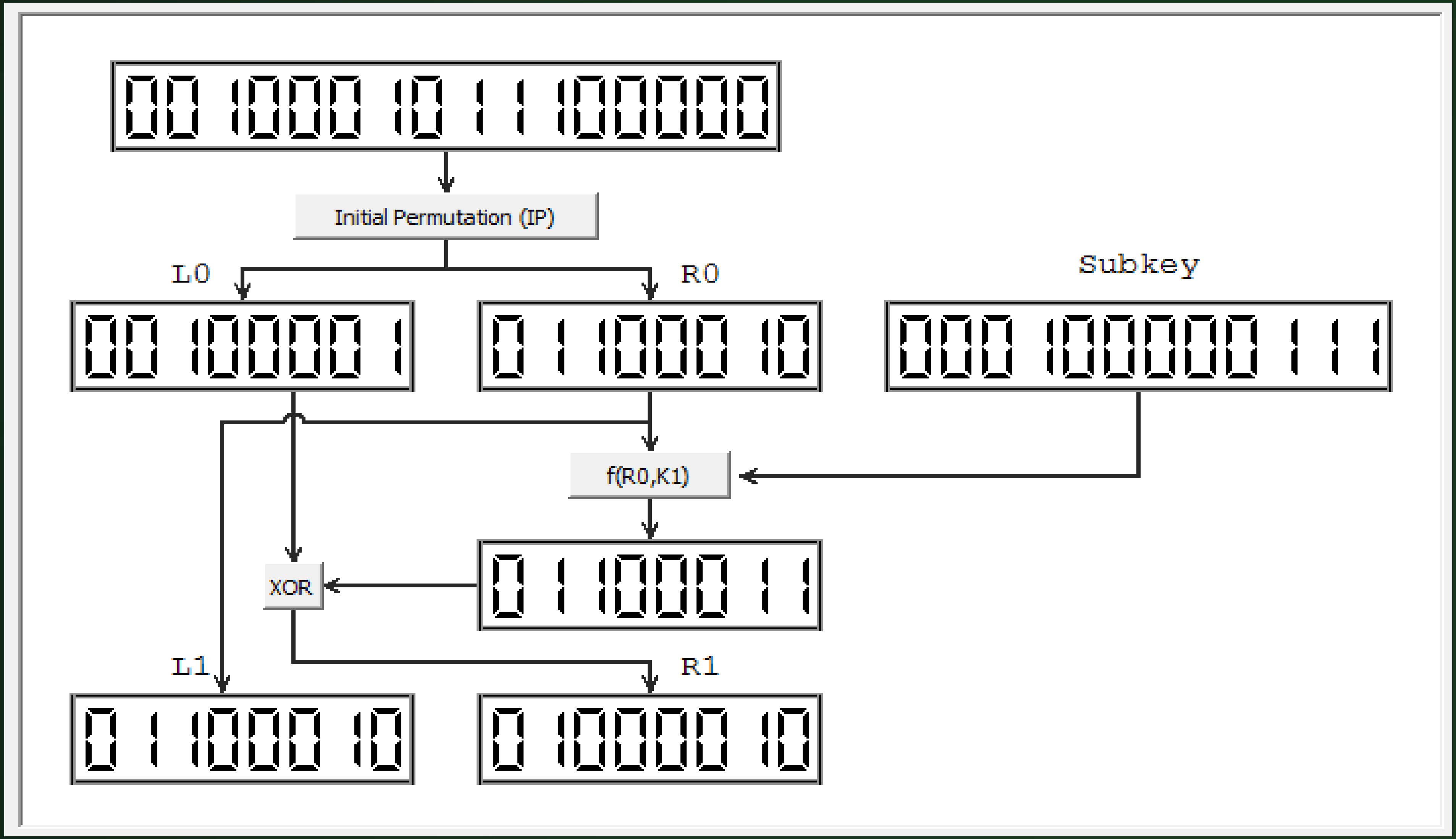
DES Example: Decryption

DES Example: Decryption

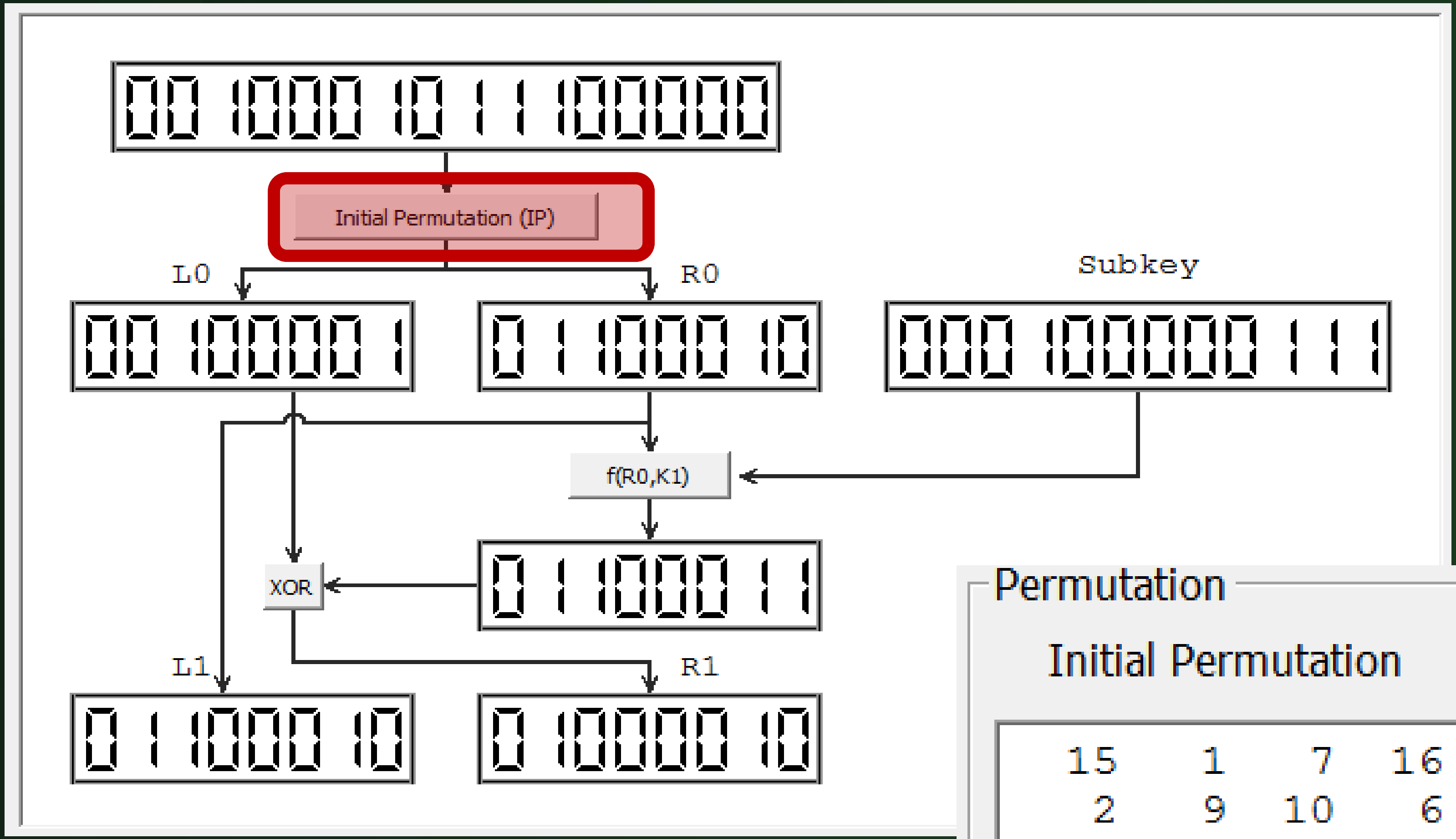


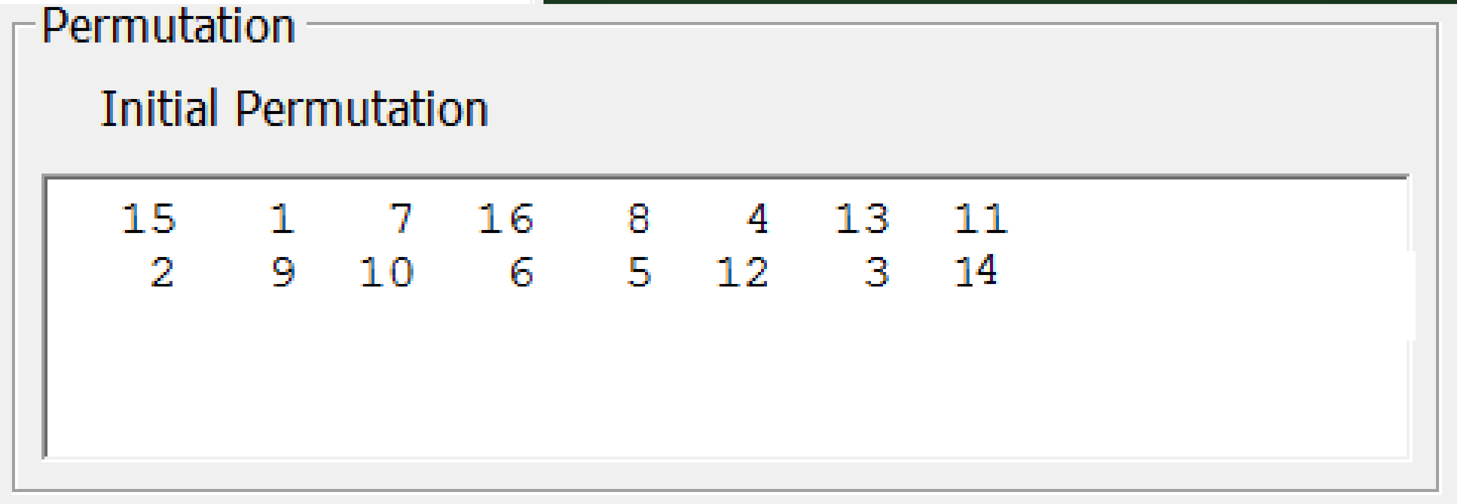
DES Example: (16-bit)

DES Example: (16-bit) Encryption

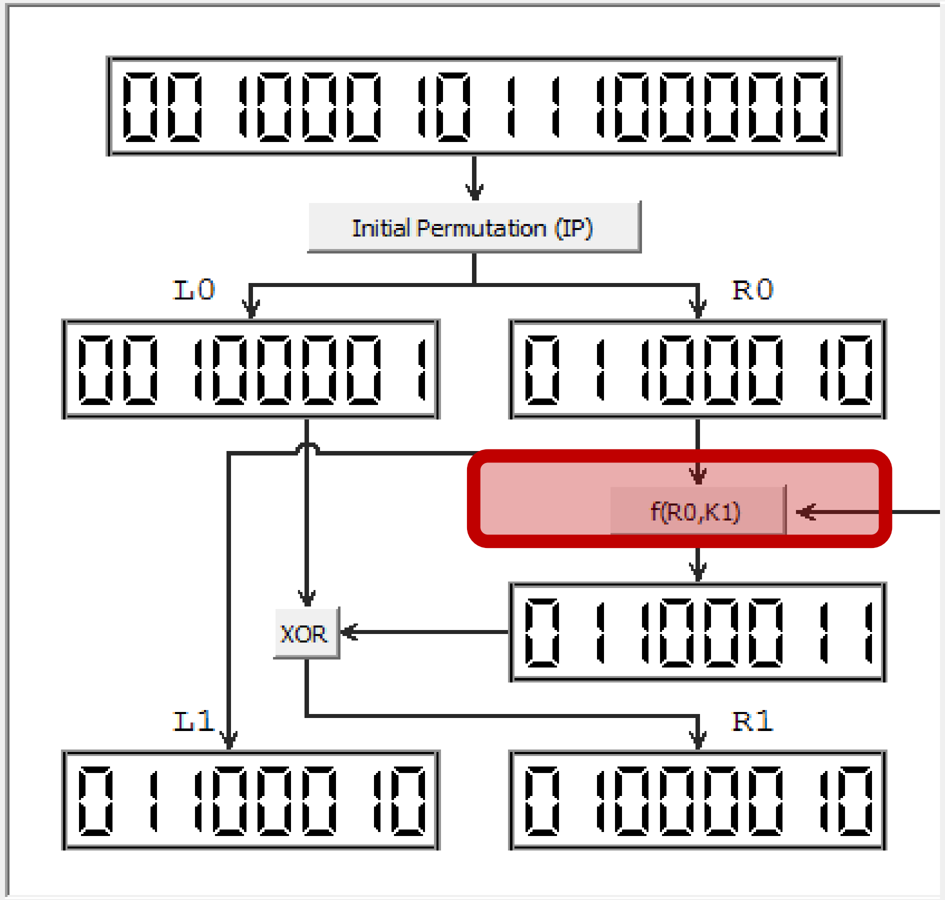


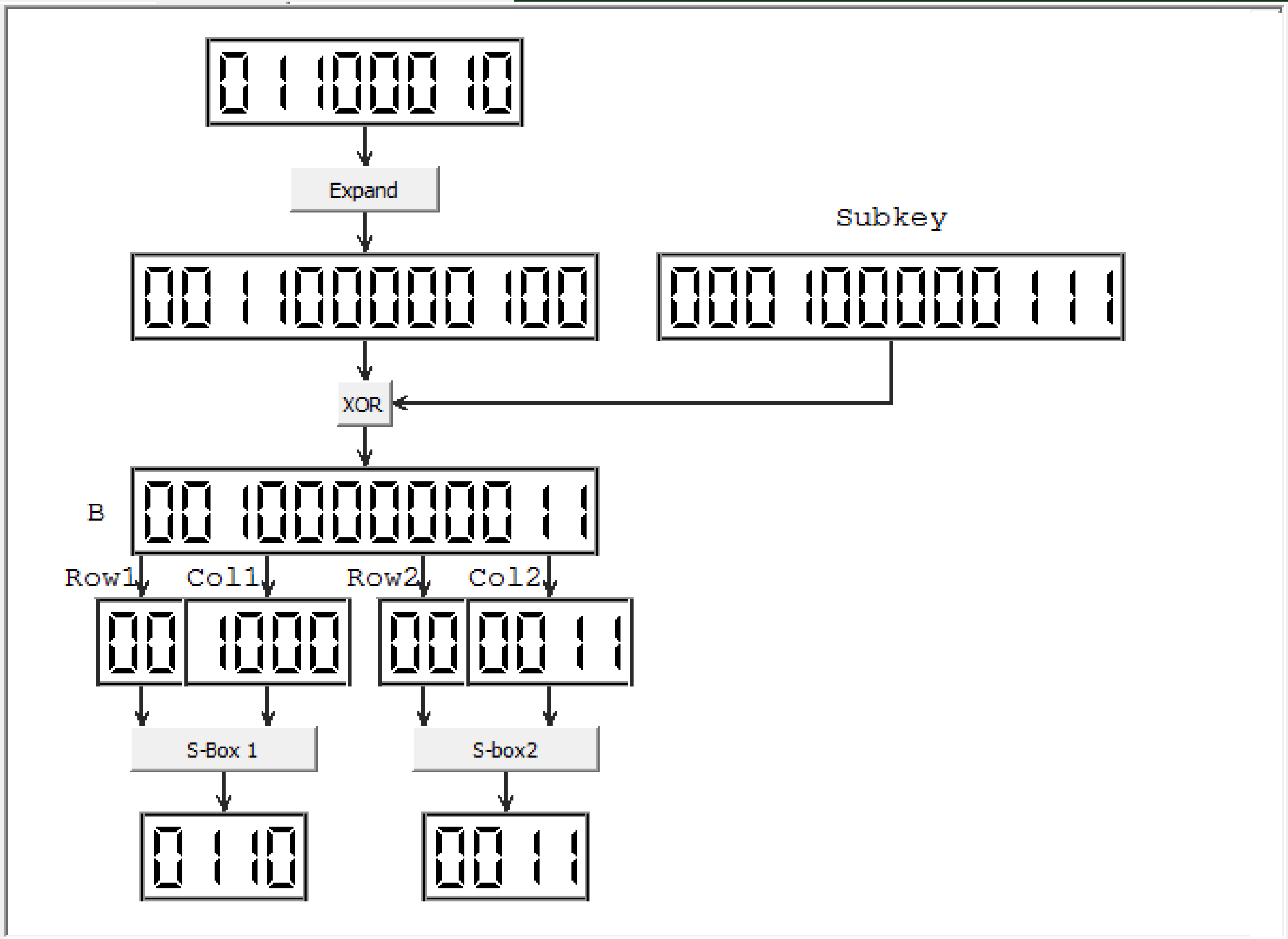
Initial Permutation



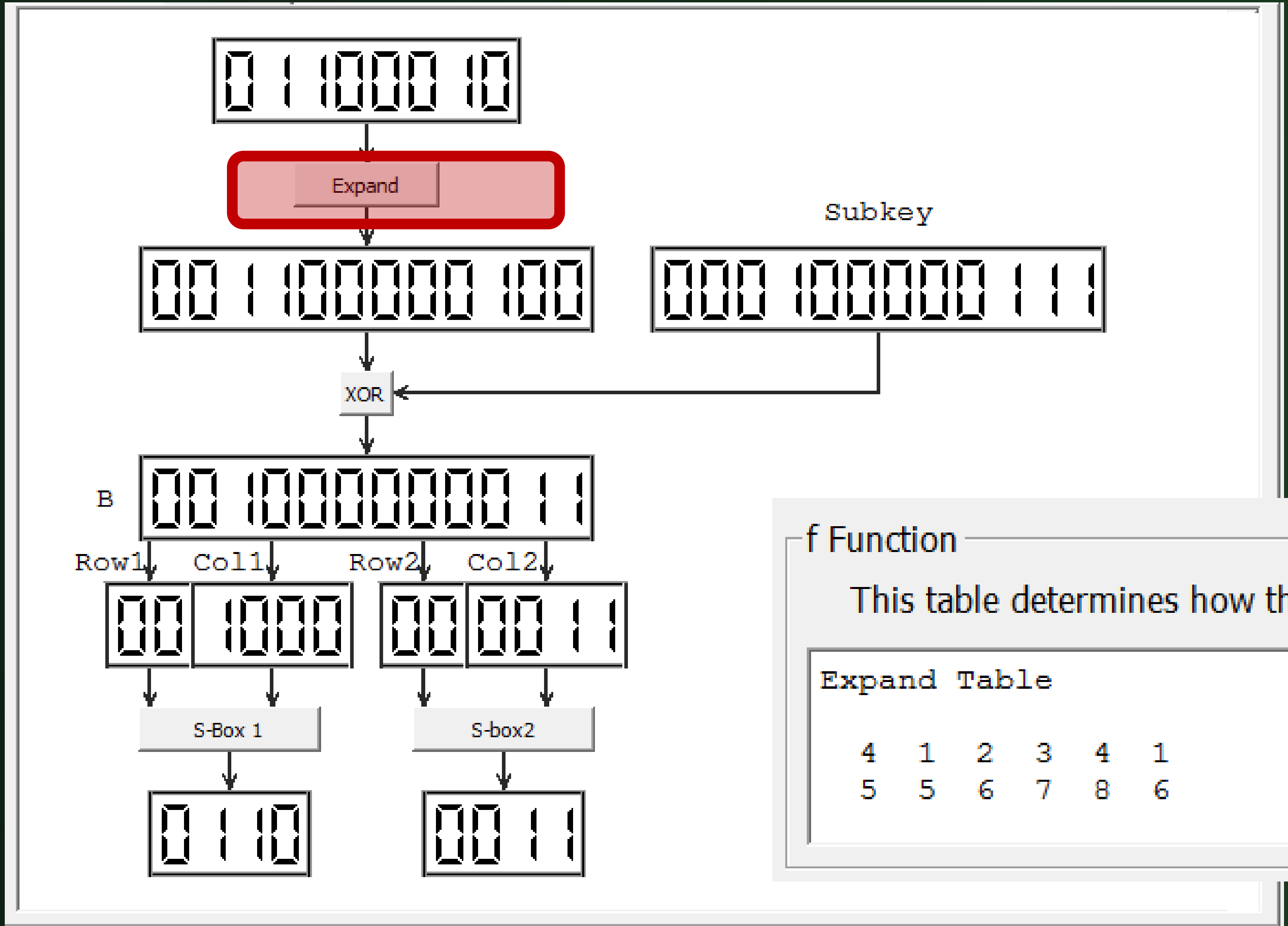


Function: F(R0,K1)





Function: F(R0,K1) : Expand



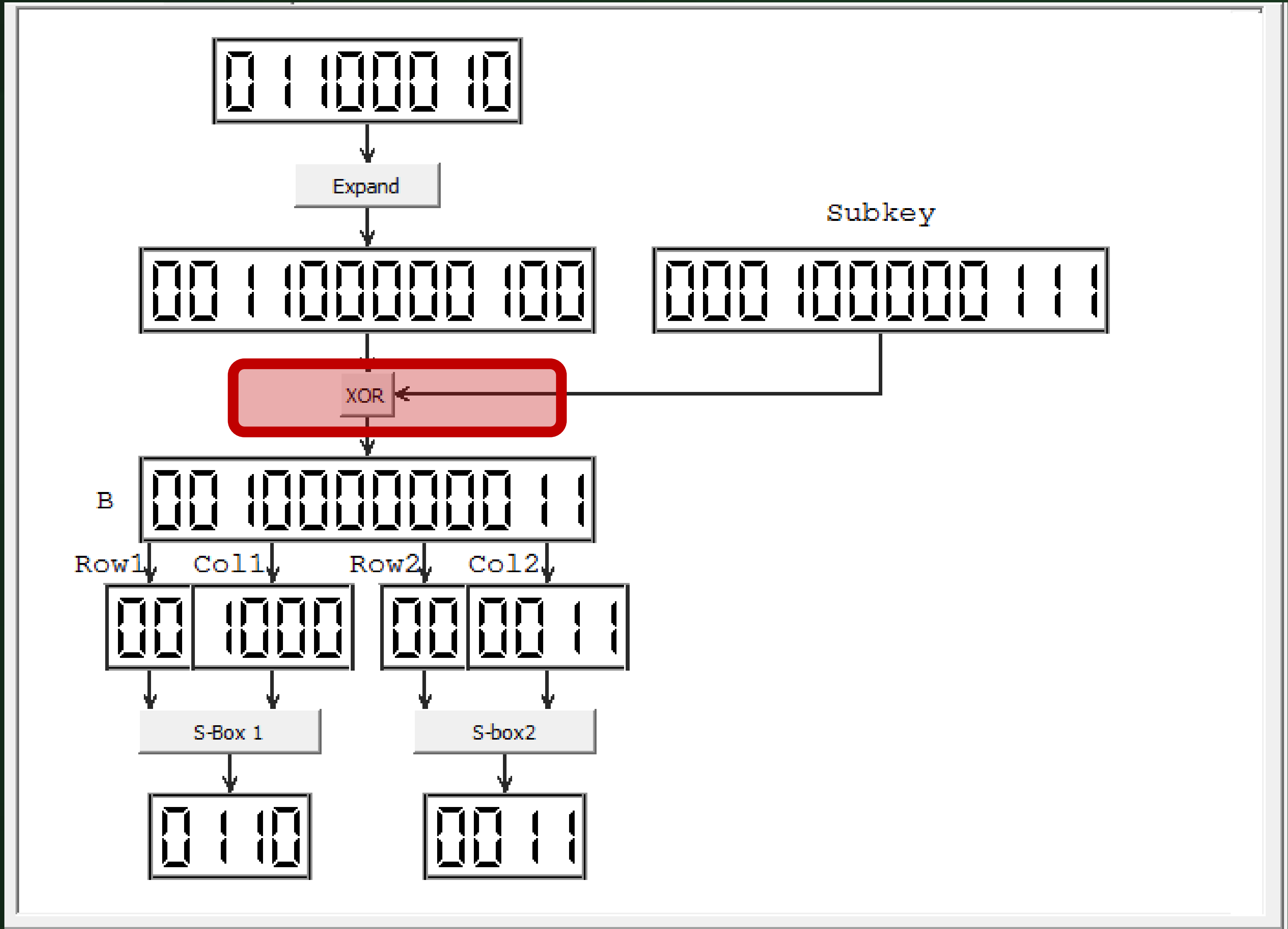
f Function

This table determines how the 8-bit data block expands to 12-bit.

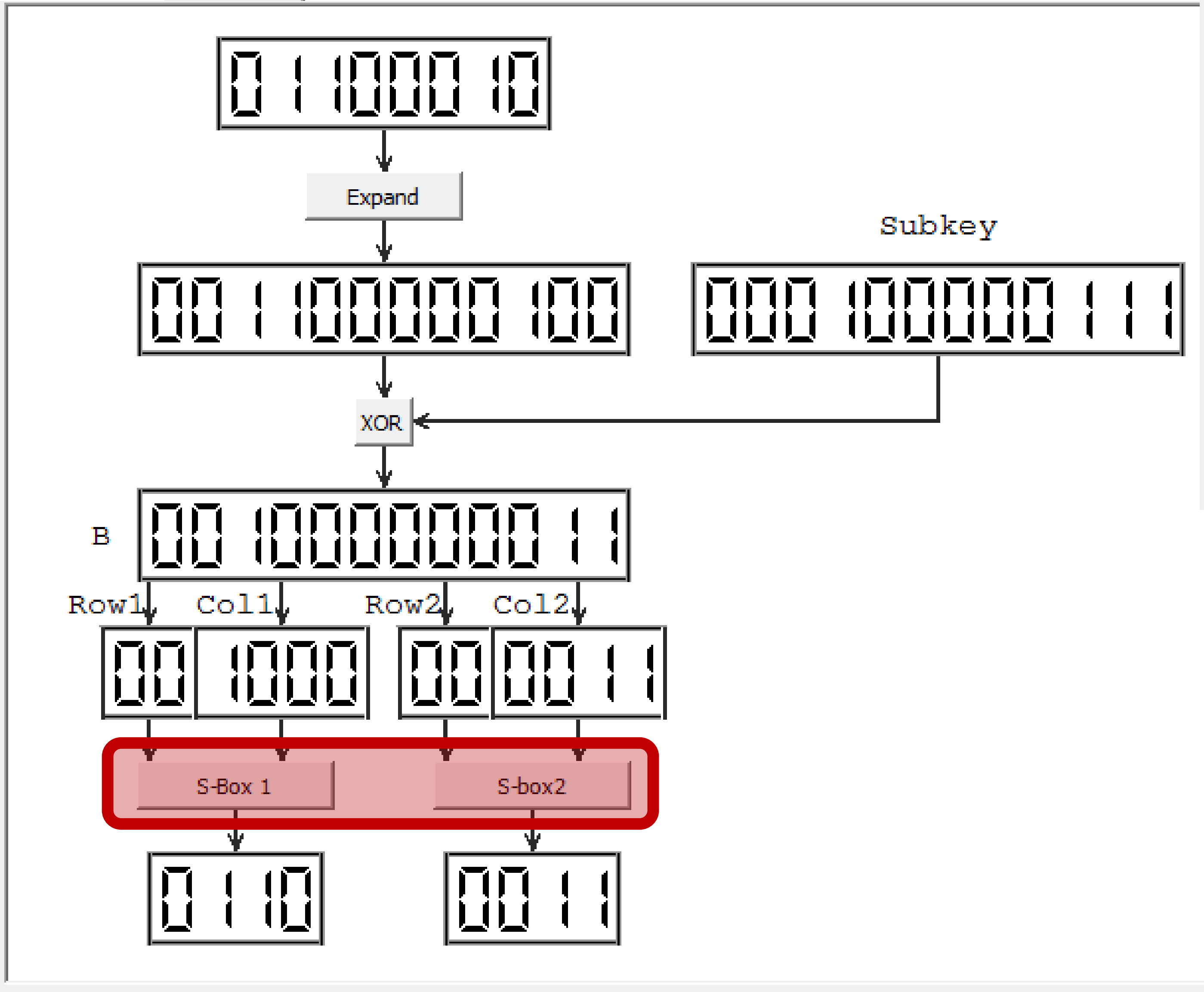
Expand Table

4	1	2	3	4	1
5	5	6	7	8	6

Function: F(R0,K1) : Perform XoR



Function: F(R0,K1) : Perform S-Box



f Function

S-Box is a 4x16 table, in which each cell is a 4-bit data block.

S-box:

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	0	10	3	7	2	13	11	6	5	12	9	8	1	4	15	
1	15	2	14	4	1	7	5	12	11	0	10	8	9	13	3	6	S1
2	0	9	10	14	1	3	15	8	11	7	12	4	5	6	13	2	
3	14	8	9	6	11	15	5	3	2	13	10	0	1	7	4	12	
0	15	1	8	3	5	6	14	10	13	11	9	7	12	0	2	4	
1	1	7	8	10	9	13	12	3	15	11	4	6	14	2	0	5	S2
2	5	14	10	2	11	6	4	1	12	7	9	0	15	3	8	13	
3	9	4	5	10	12	3	7	6	0	11	1	15	2	13	8	14	

f Function

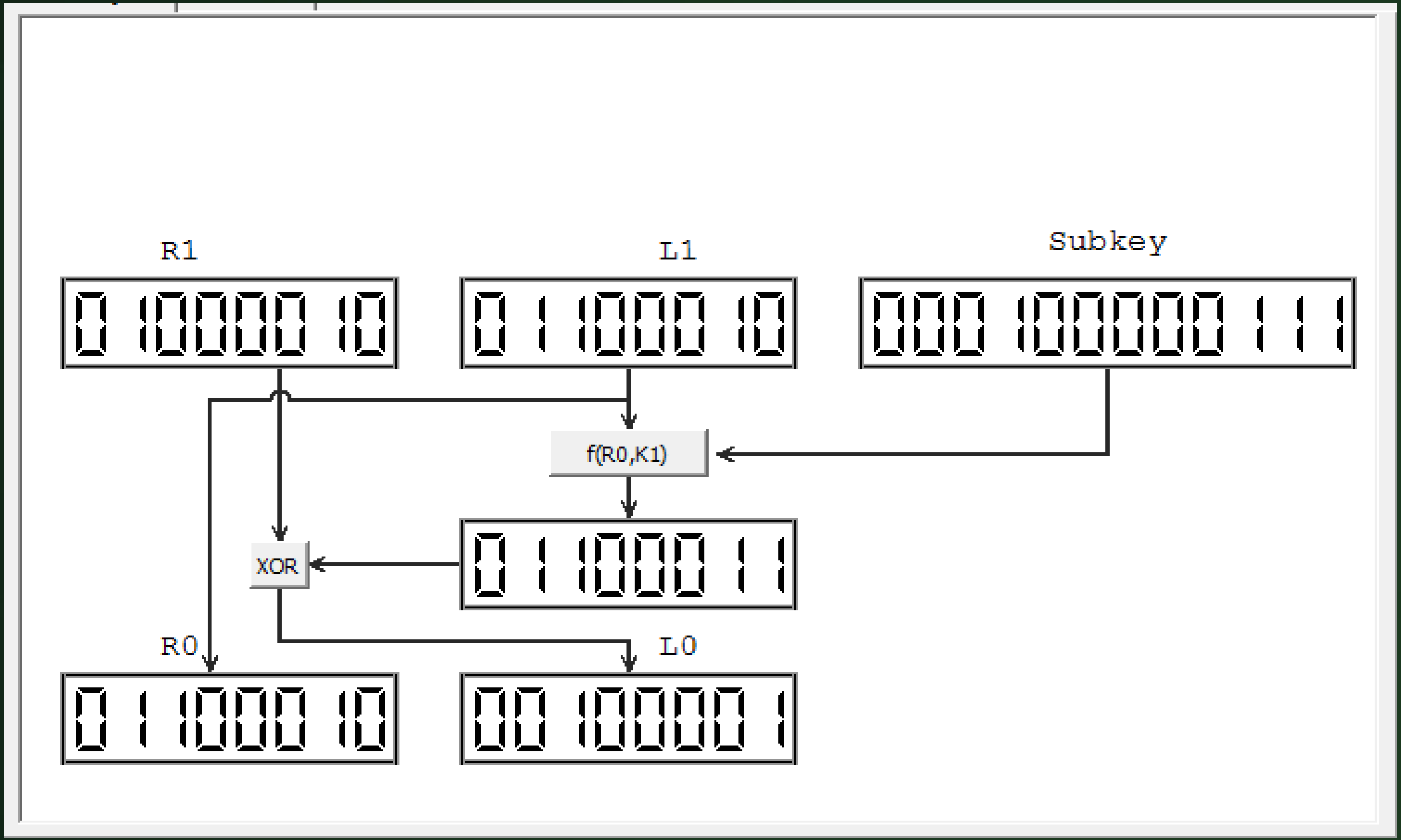
S-Box is a 4x16 table, in which each cell is a 4-bit data block.

Calculation:

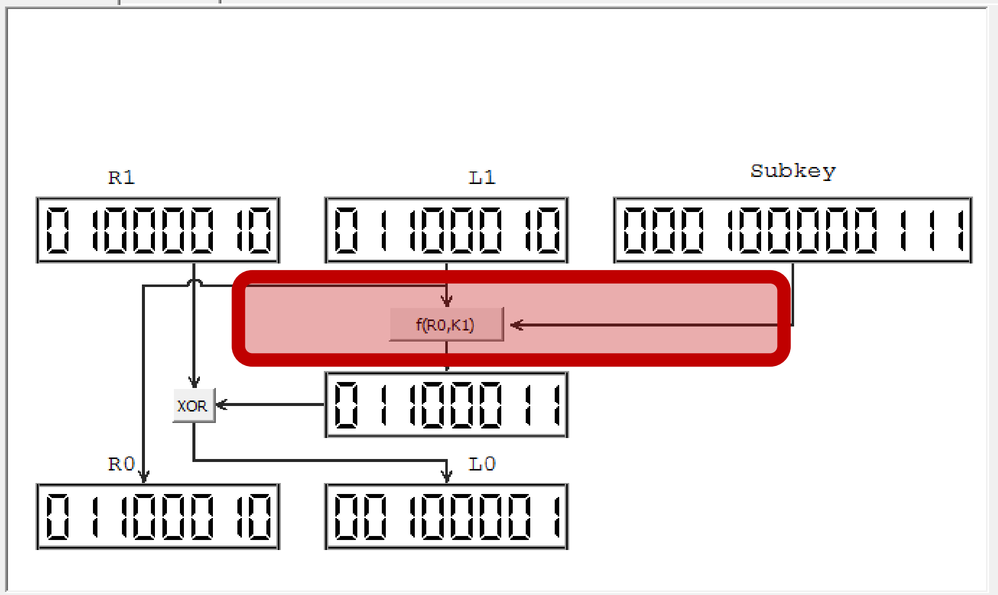
Row1:(00b): 0
Column1:(1000b): 8
Return Value in S-Box 1 at row 0, and col 8: 6
Represent this value in binary: 0110

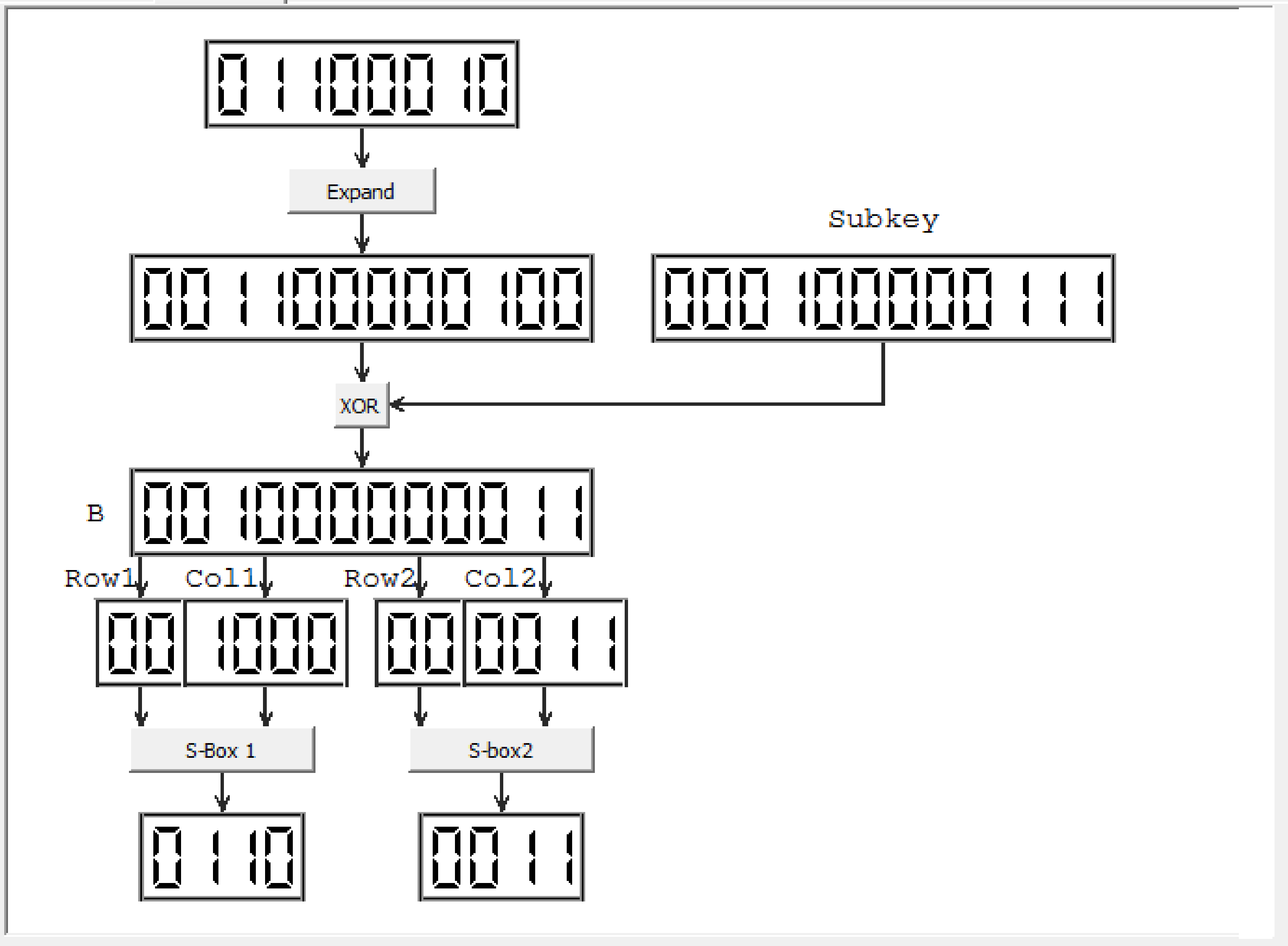
Row2:(00b): 0
Column2:(0011b): 3
Return Value in S-Box 2 at row 0, and col 3: 3
Represent this value in binary: 0011

DES Example: (16-bit) Decryption

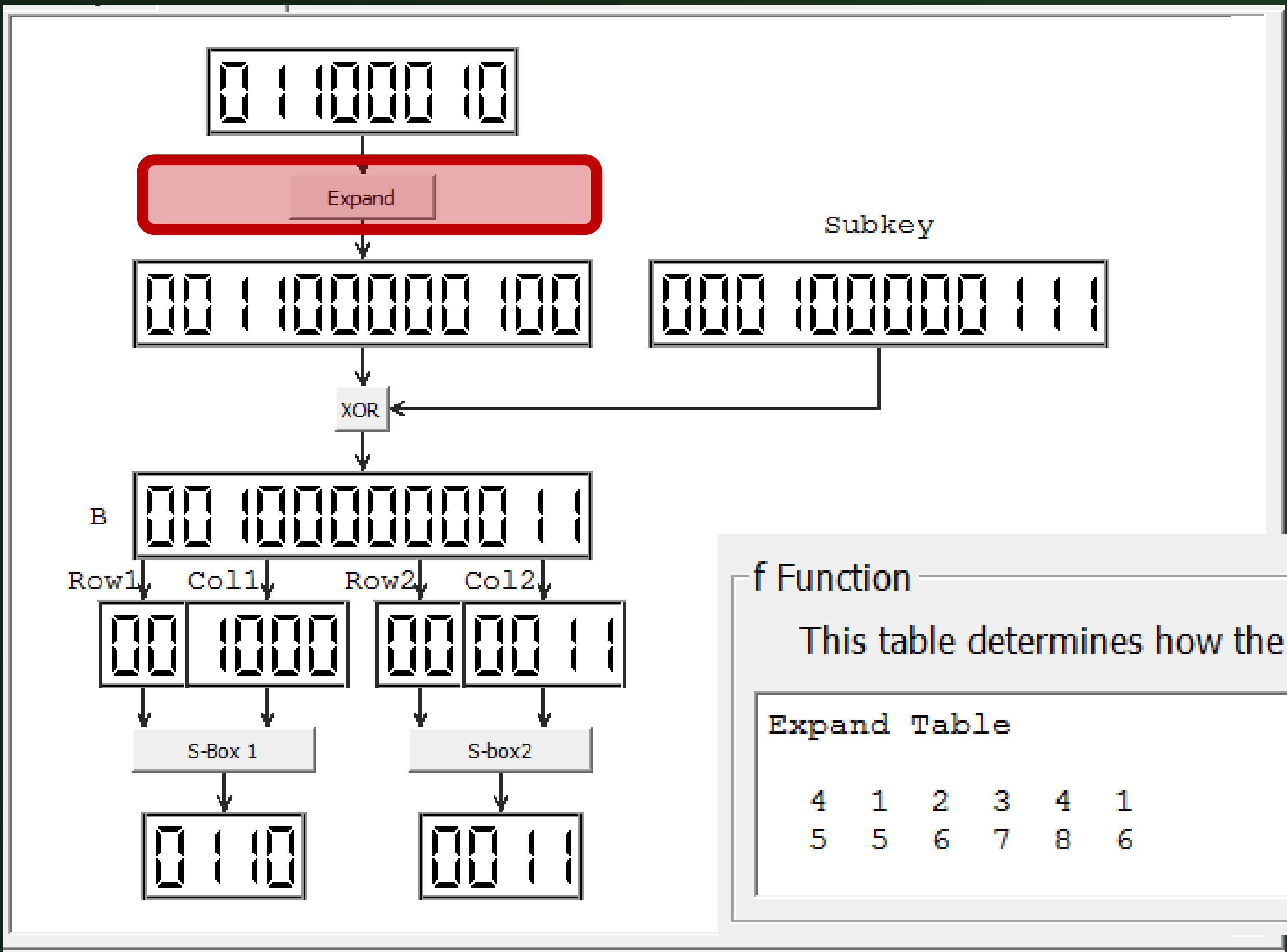


DES Example: Function(R0,K1)





DES Example: Expand



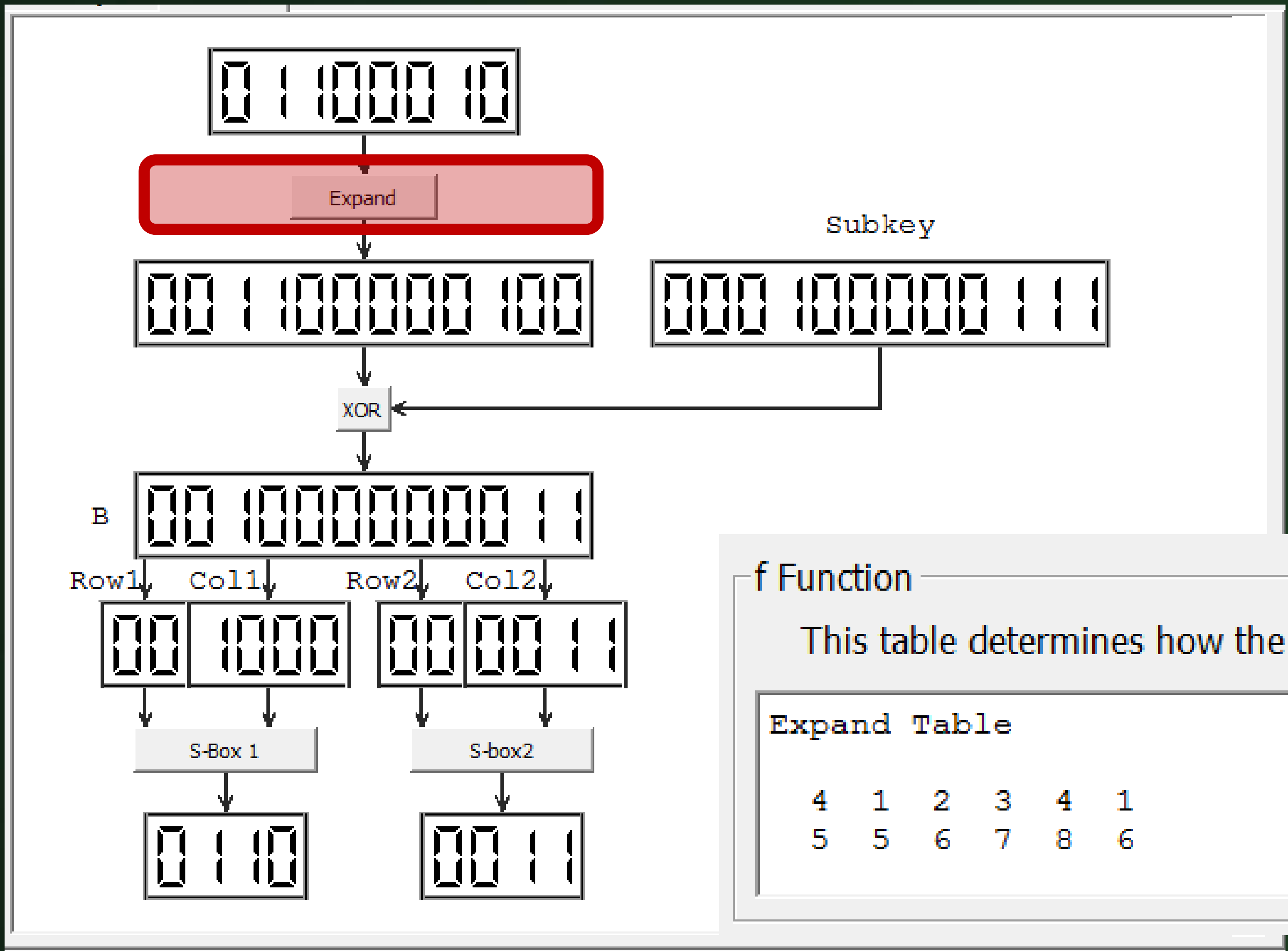
f Function

This table determines how the 8-bit data block expands to 12-bit.

Expand Table

4	1	2	3	4	1
5	5	6	7	8	6

DES Example: Expand



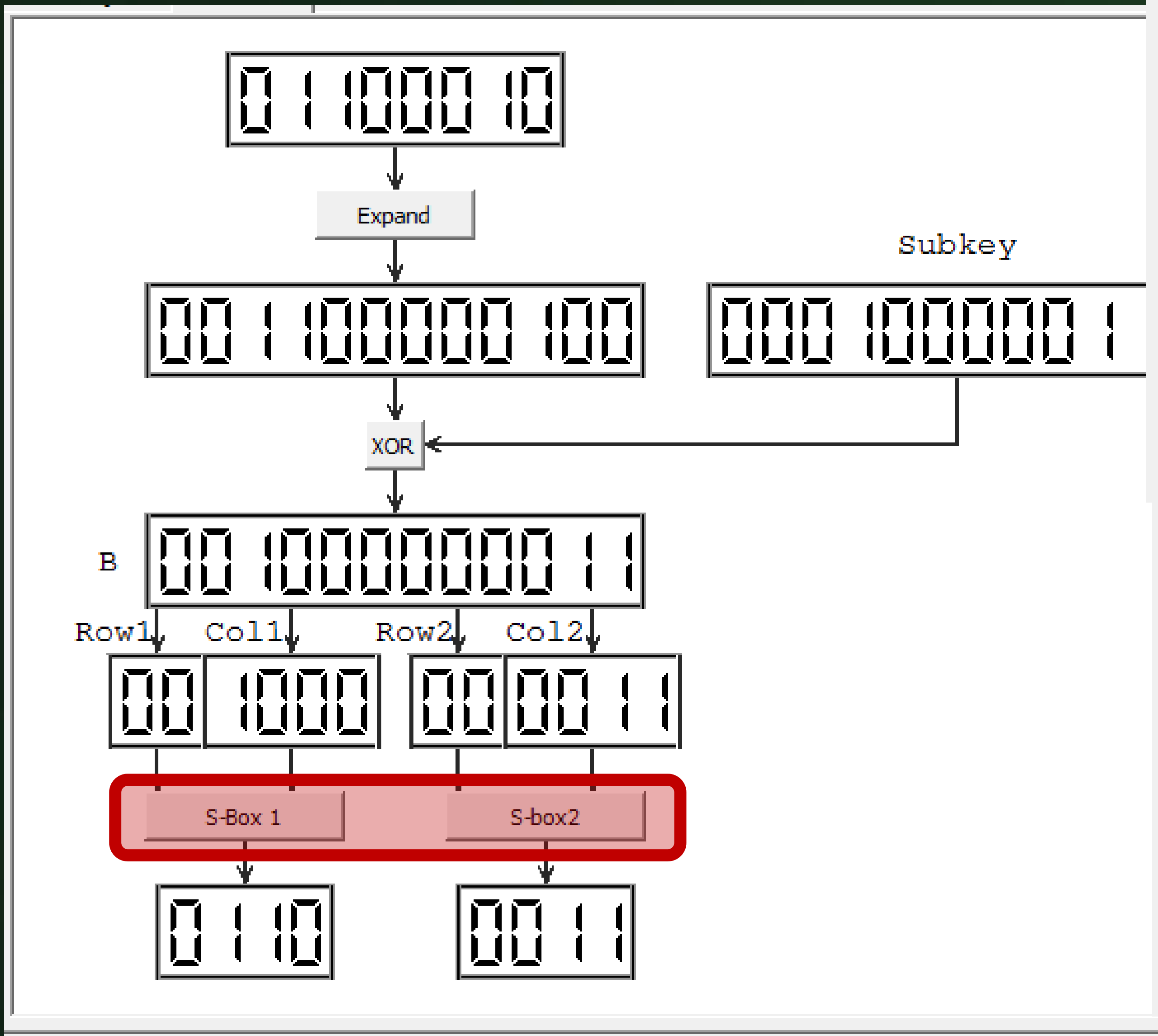
f Function

This table determines how the 8-bit data block expands to 12-bit.

Expand Table

4	1	2	3	4	1
5	5	6	7	8	6

DES Example: Expand



f Function

S-Box is a 4x16 table, in which each cell is a 4-bit data block.

S-box:

Column

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	0	10	3	7	2	13	11	6	5	12	9	8	1	4	15	
1	15	2	14	4	1	7	5	12	11	0	10	8	9	13	3	6	S1
2	0	9	10	14	1	3	15	8	11	7	12	4	5	6	13	2	
3	14	8	9	6	11	15	5	3	2	13	10	0	1	7	4	12	
0	15	1	8	3	5	6	14	10	13	11	9	7	12	0	2	4	
1	1	7	8	10	9	13	12	3	15	11	4	6	14	2	0	5	S2
2	5	14	10	2	11	6	4	1	12	7	9	0	15	3	8	13	
3	9	4	5	10	12	3	7	6	0	11	1	15	2	13	8	14	

f Function

S-Box is a 4x16 table, in which each cell is a 4-bit data block.

Calculation:

Row1:(00b): 0
Column1:(1000b): 8
Return Value in S-Box 1 at row 0, and col 8: 6
Represent this value in binary: 0110

Row2:(00b): 0
Column2:(0011b): 3
Return Value in S-Box 2 at row 0, and col 3: 3
Represent this value in binary: 0011

DES Example: Hexadecimal

DES Example

- We now work **through an example**. you are **not expected to duplicate** the **example by hand**, *you will find it informative to study the hex patterns that occur from one step to the next.*
- For this Example, the plaintext is a **hexadecimal palindrome**.

Plaintext:	**02468aceeca86420**
Key:	**0f1571c947d9e859**
Cipher text:	**da02ce3a89ecac3b**

DES Example

Round	K_i	L_i	R_i
IP	**Initial Permutation**	5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}	**Final Permutation**	da02ce3a	89ecac3b

16 Round Keys

Note: **DES subkeys** are shown as **eight 6-bit values** in hex format

Observation

- Table from previous slide shows the **progression of the algorithm**.
  - ✓ The **first row** shows the **32-bit values** of the **left and right halves** of data after the **initial permutation**.
  - ✓ The **next 16 rows show** the results **after each round**.
  - ✓ The **Key value** are shown is the **48-bit subkey generated** for **each round**.
  - ✓ The **final row** shows the **left and right-hand values** after the **inverse initial permutation**. These **two values combined** form the **ciphertext**

The Avalanche Effect

- A **desirable property of any encryption algorithm** is that a **small change** in either the **plaintext or the key** should produce a **significant change in the ciphertext**.
- In particular, a **change in one bit of the plaintext** or **one bit of the key** should produce a change in **many bits of the ciphertext**. This is referred to as the **Avalanche Effect**

Example of the Avalanche Effect

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Example of the Avalanche Effect

- The Result shows that when the **fourth bit of the plaintext** is **changed**, so that the plaintext is **12468aceeca86420.**
  - ✓ The **second column** of the table shows the **intermediate 64-bit values** at the end of **each round for the two plaintexts**.
  - ✓ The **third column** shows the **number of bits** that differ between the **two intermediate values.**

Example of the Avalanche Effect

- The table shows that, **after just three rounds, 18 bits differ between the two blocks.**
- On completion, **the two cipher texts differ in 32 bit positions.**

Example of the Avalanche Effect with Change in Key

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

Example of the Avalanche Effect with Change in Key

- Similar test using the **original plaintext of with two keys** that differ in only the **fourth bit position**:
  - ✓ The Original key → **0f1571c947d9e859**, and
  - ✓ The Altered key → **1f1571c947d9e859**.
- The results show that about **half of the bits in the ciphertext** differ and that the **avalanche effect** is pronounced **after just a few rounds**

THE STRENGTH OF DES

- The **substitution step** is **very effective** as far as **diffusion** is concerned.
  - ✓ It has been shown that if you **change just one bit** of the **64-bit input data block,** on the average that **alters 34 bits of the ciphertext block.**
- The manner in which the **round keys are generated** from the **encryption key** is also **very effective** as far as **confusion** is concerned.
  - ✓ It has been shown that if you **change just one bit** of the encryption key, on the average that changes **35 bits of the ciphertext.**

THE STRENGTH OF DES

- The **level of security** provided by **DES** that falls into **two areas:**
  - ✓ **Key Size** and
  - ✓ The **Nature of the Algorithm**.

Key Size

- With a **key length of 56 bits**, there are **possible keys 2^{56}**, which is approximately **7.2×10^{21} keys**.
  - ✓ Brute-force attack appears impractical.
  - ✓ Assuming that, on average, **half 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.**

Key Size-Observation

- The assumption of **one encryption per microsecond** is overly conservative.
  - ✓ In 1977, **Diffie and Hellman** proposed a **parallel machine** with **1 million encryption devices**
  - ✓ Each of which could perform **one encryption per microsecond**
  - ✓ **Average search** time down to **about 10 hours**. The authors estimated that the cost would be about **$20 million** in 1977.

Bird View-DES is Insecure

- **DES** finally and definitively **proved insecure in July 1998**,
  - ✓ When the **Electronic Frontier Foundation (EFF)** announced that it had **broken a DES encryption** using **a special-purpose "DES cracker" machine** that was built for less than **$250,000.**
  - ✓ The attack took **less than three days**.
  - ✓ **Hardware prices** will continue to **drop as speeds increase**, making **DES virtually worthless.**

The Nature of the DES Algorithm

- The **cryptanalysis** is possible by **exploiting the characteristics of the DES algorithm**.
  - ✓ The focus of concern has been on the **eight substitution tables, or S-boxes**, that are used in each iteration.
  - ✓ **S-boxes** and **Entire algoithm** were not made public.
  - ✓ There is a **suspicion** that **the boxes were constructed** in such a way that **cryptanalysis is possible** for an opponent who knows the **weaknesses in the S-boxes**

Thank U