Lecture 5 DES

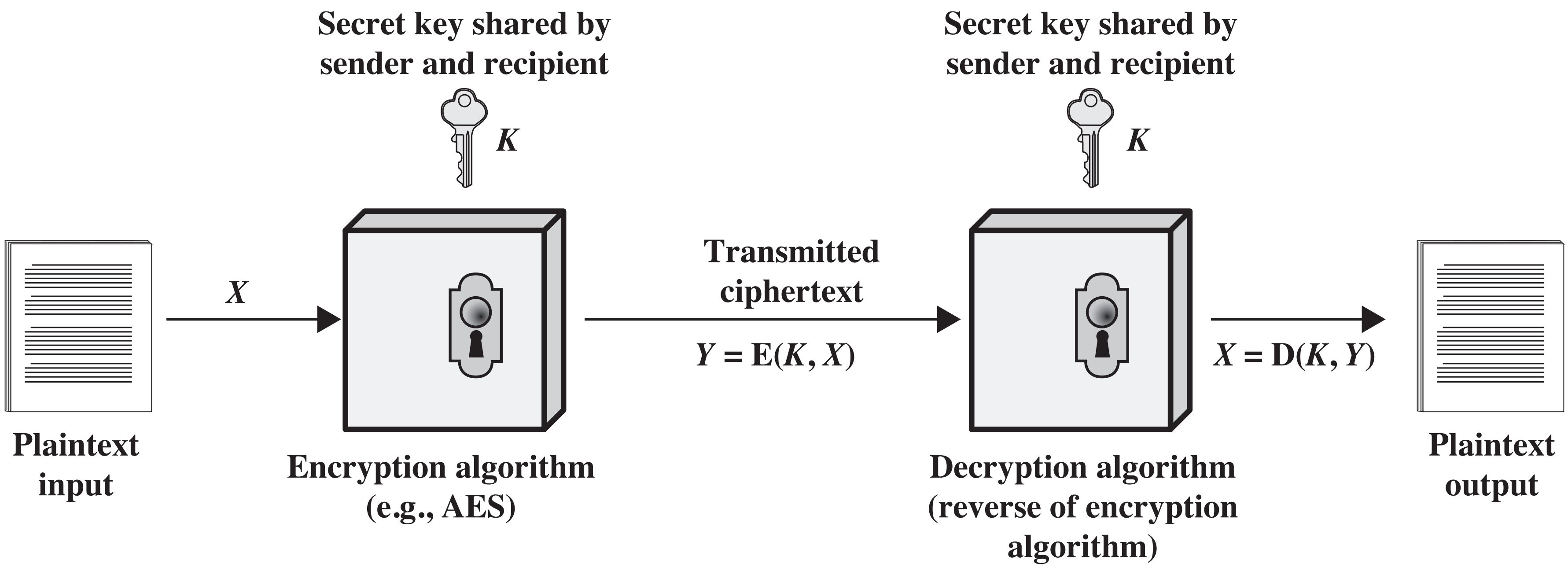
Modern Symmetric Ciphers--Data Encryption Standard (DES)

Encryption Technique Overview

- Classic encryption and modern encryption
  - Classic encryption
    - or classic cipher
    - dated back to Greek and Roman times
  - Modern encryption
    - mainly relies on Number Theory
    - was developed in 70s
- Symmetric encryption and asymmetric encryption
  - if the sender and receiver use the same key

- What kind of cryptosystem we are using now?
  - Symmetric
    - DES, AES, etc.
  - Asymmetric
    - RSA, Diffie-Hellman, etc.

Model of Symmetric Encryption



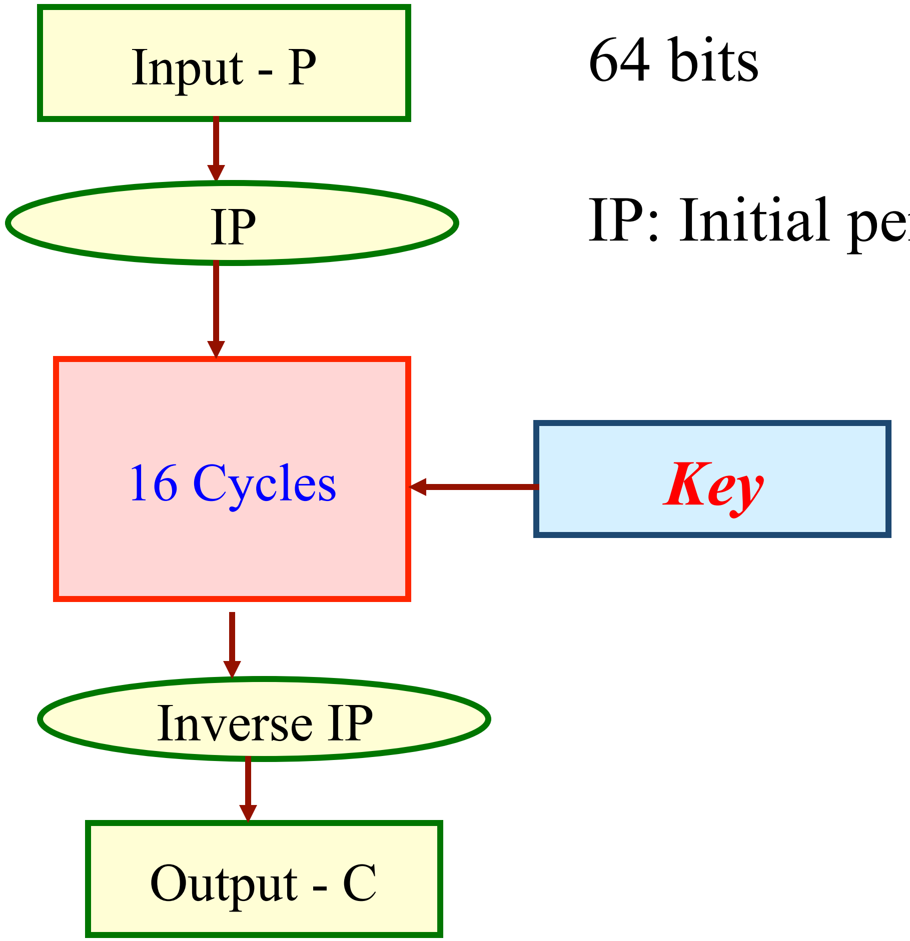
Symmetric Encryption

- **Stream ciphers**
  - e.g. Vernam cipher, One-time-pad, etc.
- **Block ciphers**
  - e.g. DES, etc.

Data Encryption Standard (DES)

- **Combination of substitution and permutation**
  - Provides diffusion and confusion
    - Confusion: Relationship between plain & ciphertext is obscured. Ex. Substitution table (look-up table)
    - Diffusion: The influence of one of each plaintext bit is spread over many ciphertext bits. Ex. Permutation
- **Product cipher**
  - Confusion and diffusion applied multiple times to have a strong cipher

A High Level Description of DES

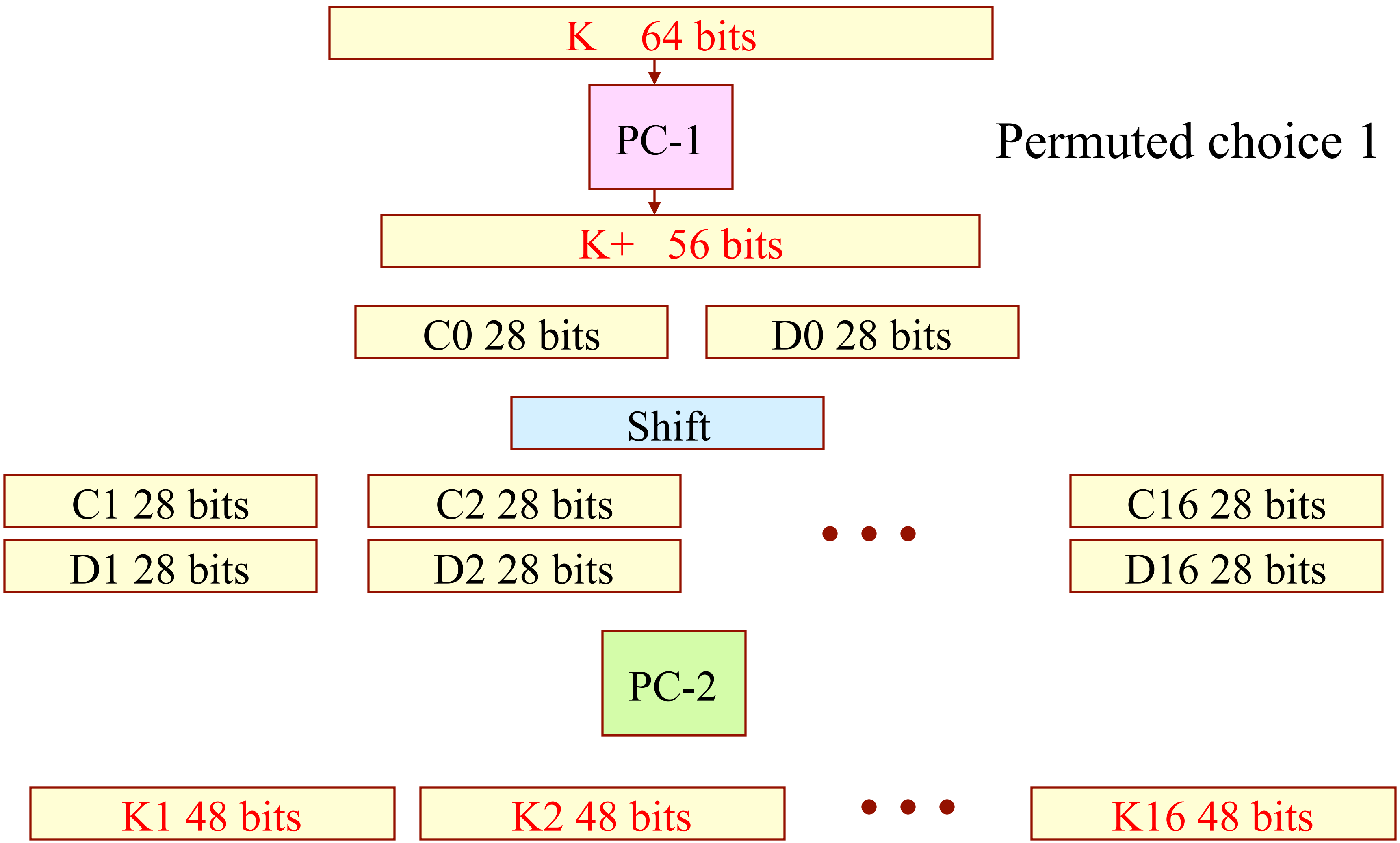


64 bits

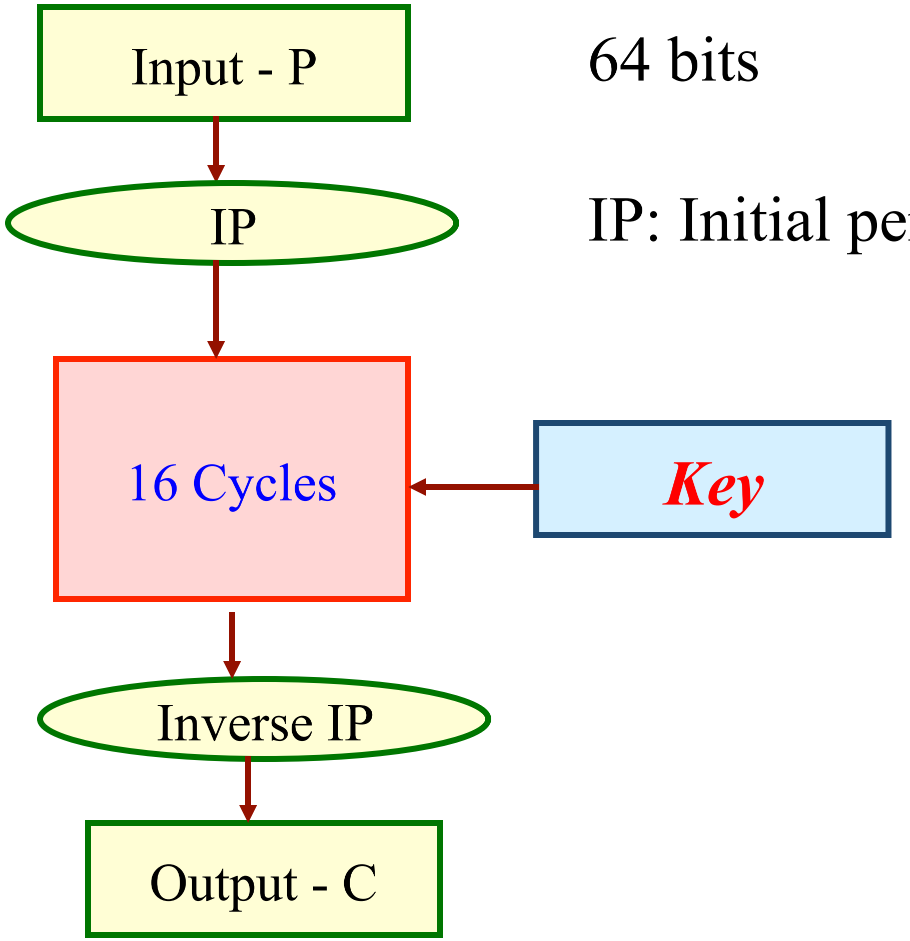
IP: Initial permutation

64 bits

Key Summary


K 64 bits
PC-1
Permuted choice 1
K+ 56 bits
C0 28 bits
D0 28 bits
Shift
C1 28 bits
D1 28 bits
C2 28 bits
D2 28 bits
• • •
C16 28 bits
D16 28 bits
PC-2
K1 48 bits
K2 48 bits
• • •
K16 48 bits

A High Level Description of DES



64 bits

IP: Initial permutation

56 bits

Detailed DES Example

Plain text message M (64 bits)

M = 0123456789ABCDEF (hexadecimal format)

M in binary format:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000
1001 1010 1011 1100 1101 1110 1111

Key

Original Key K (64 bits)

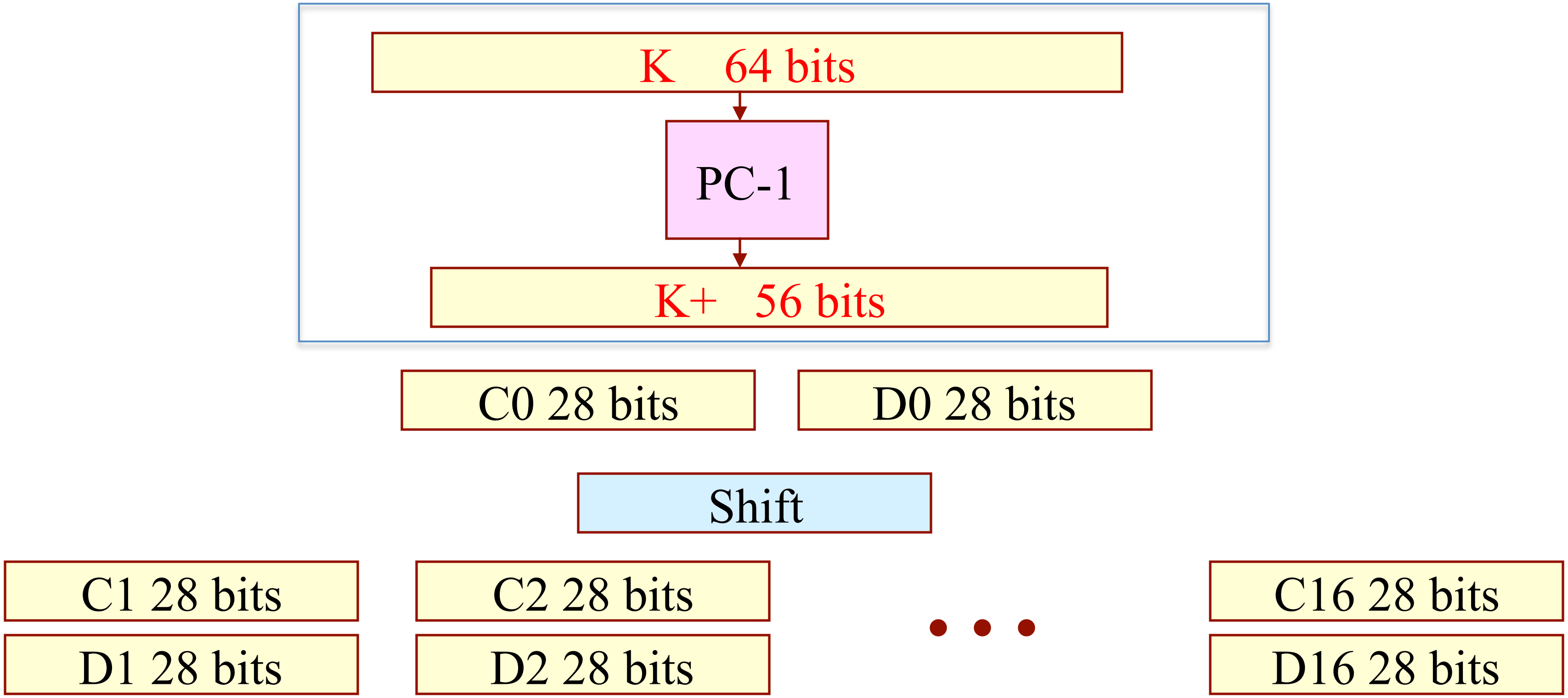
K = 133457799BBCDFF1 (hexadecimal format)

K in binary format:

K = 00010011 00110100 01010111 01111001
10011011 10111100 11011111 11110001

Step 1: Create 16 sub-keys (48-bits)

- **1.1 The 64-bit key is permuted according to table PC-1.**



Step 1: Create 16 sub-keys (48-bits)

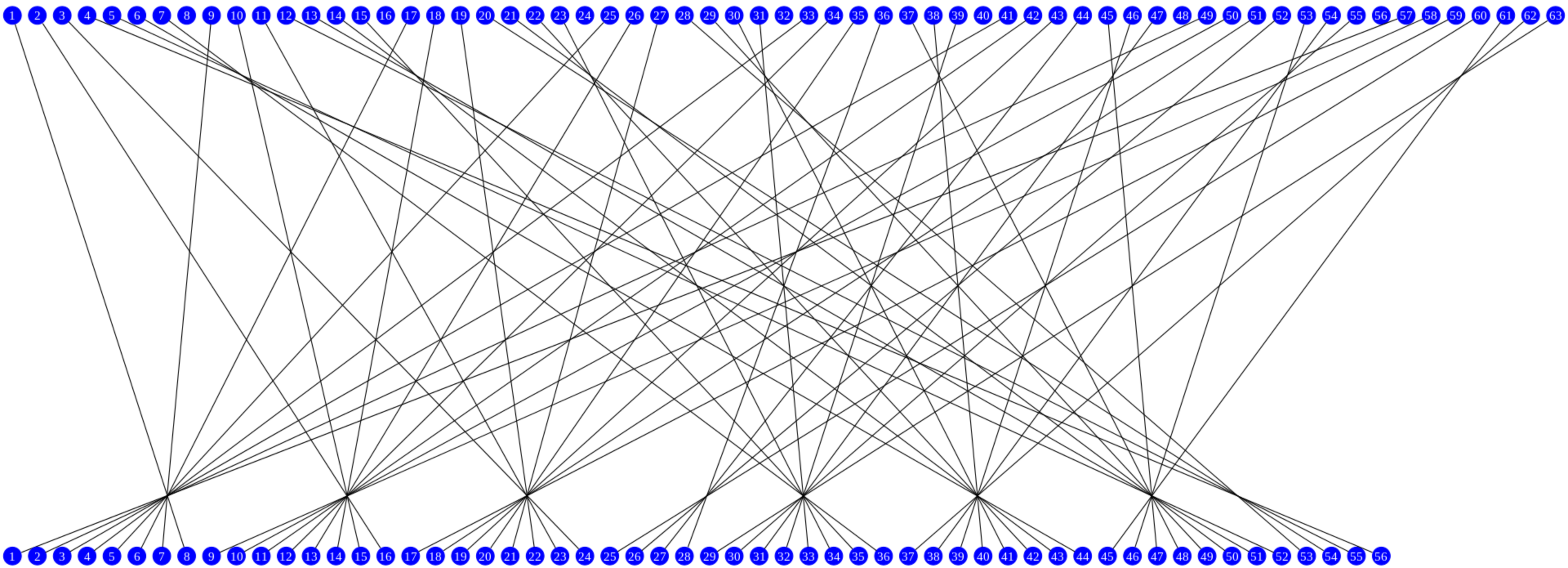
- **1.1 The 64-bit key is permuted according to table PC-1.**

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

The first bit of the output is taken from the 57th bit of the input

Step 1: Create 16 sub-keys (48-bits)

- **1.1 The 64-bit key is permuted according to table PC-1.**



Example (cont.)

From the original 64-bit key

K = 00010011 00110100 01010111 01111001
10011011 10111100 11011111 11110001

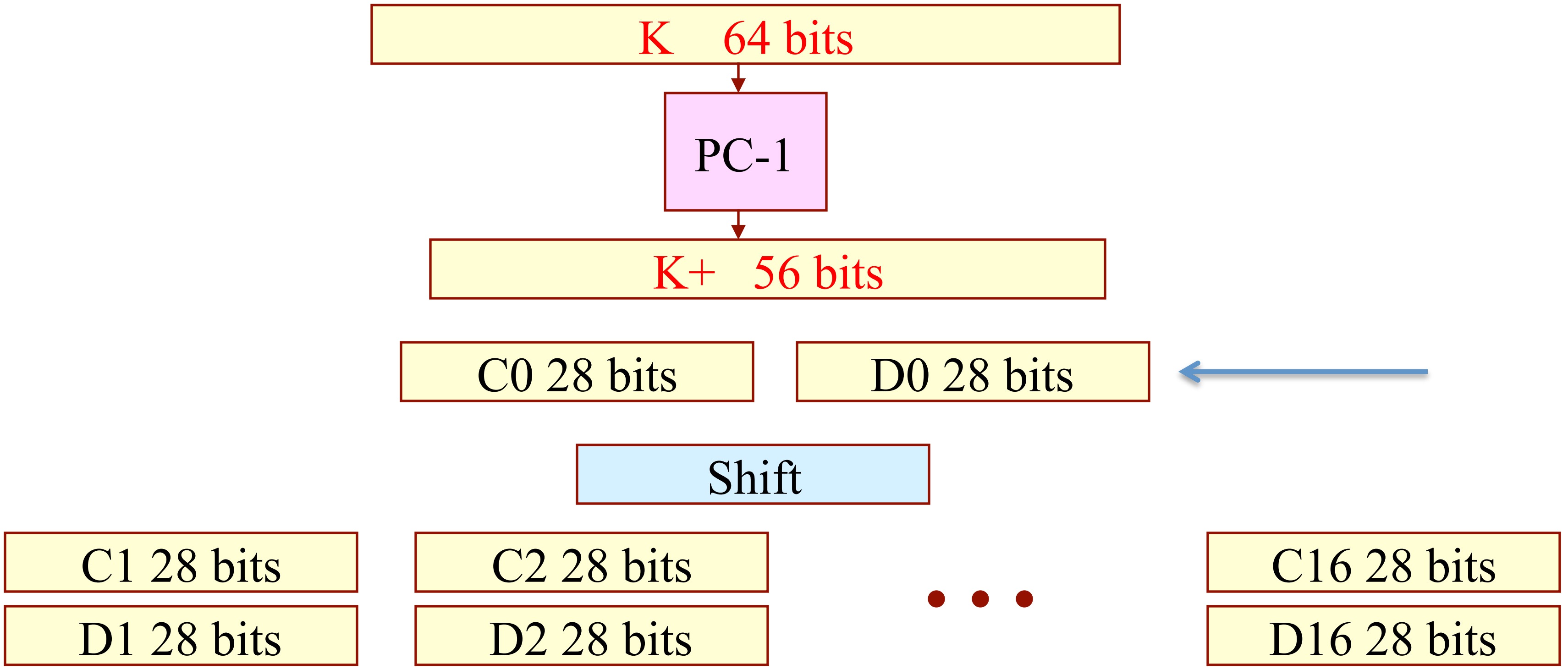
57th

Using PC-1, we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111
0101010 1011001 1001111 0001111

Split this Key

1.2 Split this key into left and right halves, C_0 and D_0, where each half has 28 bits



Split this key

1.2 Split this key into left and right halves, C_0 and D_0, where each half has 28 bits

K+ = 1111000 0110011 0010101 0101111
0101010 1011001 1001111 0001111

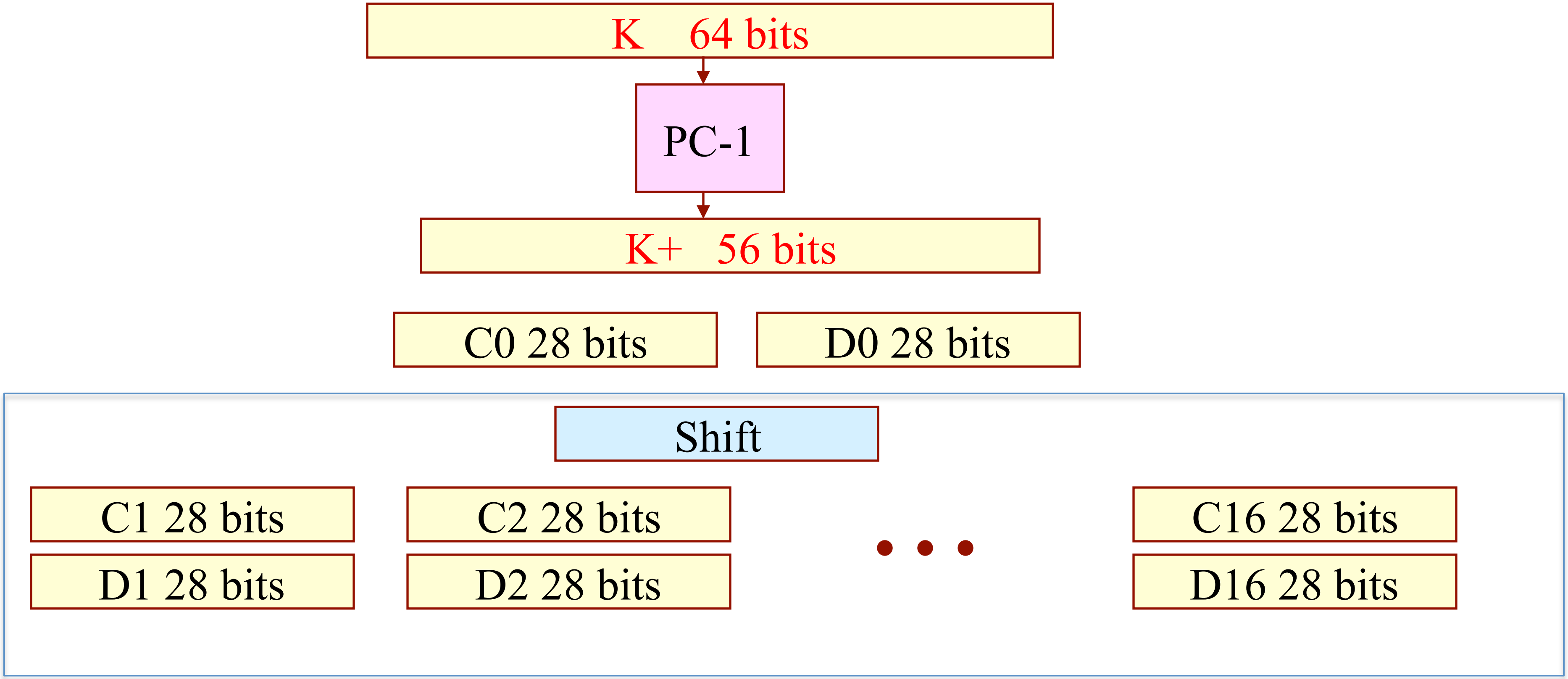
From the permuted key **K+**, we get

C_0 = 1111000 0110011 0010101 0101111

D_0 = 0101010 1011001 1001111 0001111

Create 16 Blocks

1.3 Create 16 blocks C_n and D_n, 1<=n<=16.



Create 16 blocks

1.3 Create 16 blocks C_n and D_n, $1<=n<=16$.

C_n and D_n are obtained from C_{n-1} and D_{n-1} using the following schedule of "left shifts".

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
shift	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

C1 and D1 are obtained by left shifting C0 and D0 1 bit…

Example (Cont.)

C_0 = 1111000 0110011 0010101 0101111

D_0 = 0101010 1011001 1001111 0001111

C_1 = 1110000110011001010101011111

D_1 = 1010101011001100111100011110

C_2 = 1100001100110010101010111111

D_2 = 0101010110011001111000111101

C_3 = 0000110011001010101011111111

D_3 = 0101011001100111100011110101

Example (Cont.)

C_4 = 0011001100101010101111111100

D_4 = 0101100110011110001111010101

C_5 = 1100110010101010111111110000

D_5 = 0110011001111000111101010101

C_6 = 0011001010101011111111000011

D_6 = 1001100111100011110101010101

C_7 = 1100101010101111111100001100

D_7 = 0110011110001111010101010110

Example (Cont.)

C_8 = 0010101010111111110000110011

D_8 = 1001111000111101010101011001

C_9 = 0101010101111111100001100110

D_9 = 0011110001111010101010110011

C_{10} = 0101010111111110000110011001

D_{10} = 1111000111101010101011001100

C_{11} = 0101011111111000011001100101

D_{11} = 1100011110101010101100110011

Example (Cont.)

C_{12} = 0101111111100001100110010101

D_{12} = 0001111010101010110011001111

C_{13} = 0111111110000110011001010101

D_{13} = 0111101010101011001100111100

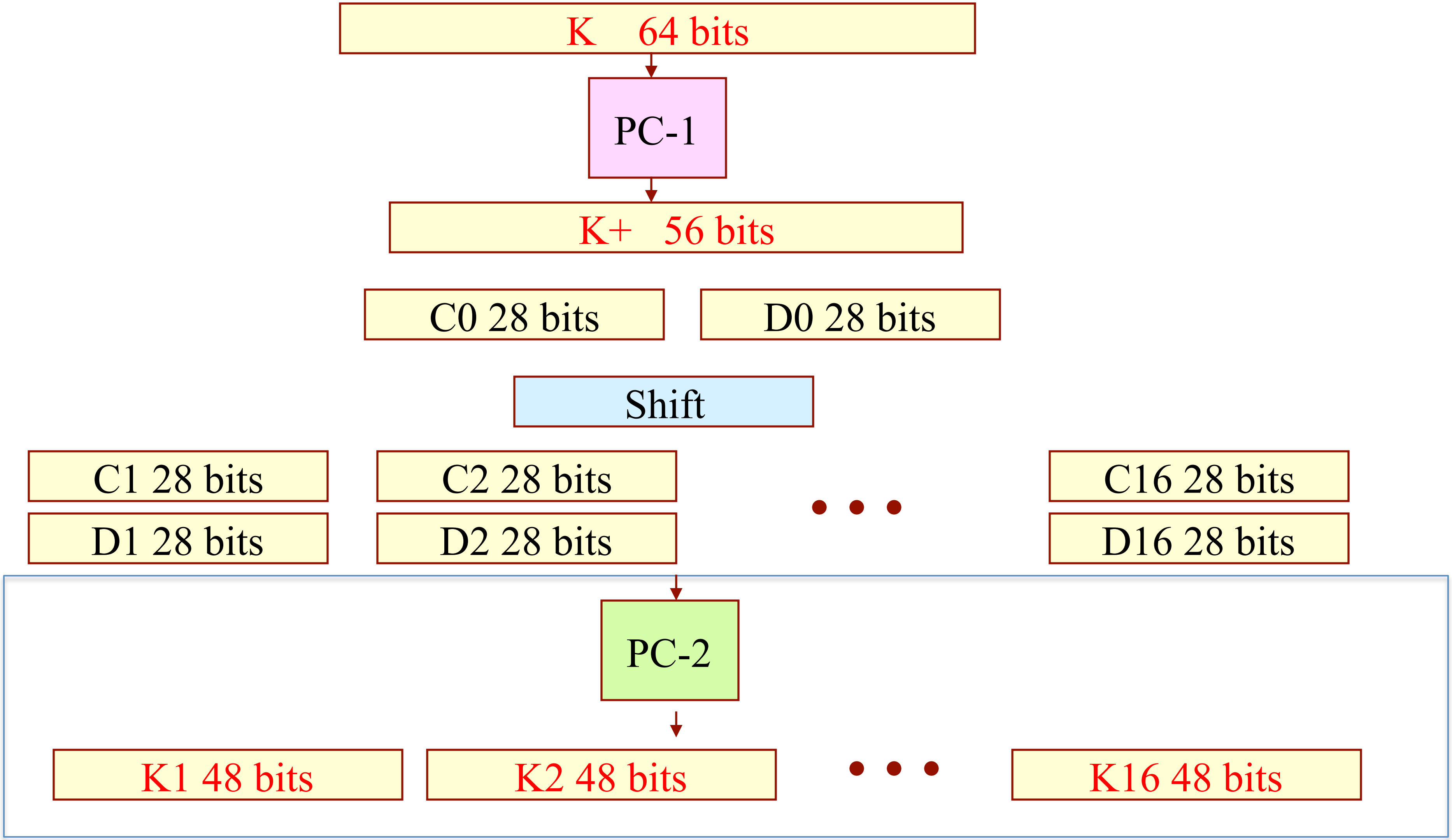
C_{14} = 1111111000011001100101010101

D_{14} = 1110101010101100110011110001

C_{15} = 1111100001100110010101010111

D_{15} = 1010101010110011001111000111

Form the keys K_n



Form the keys K_n

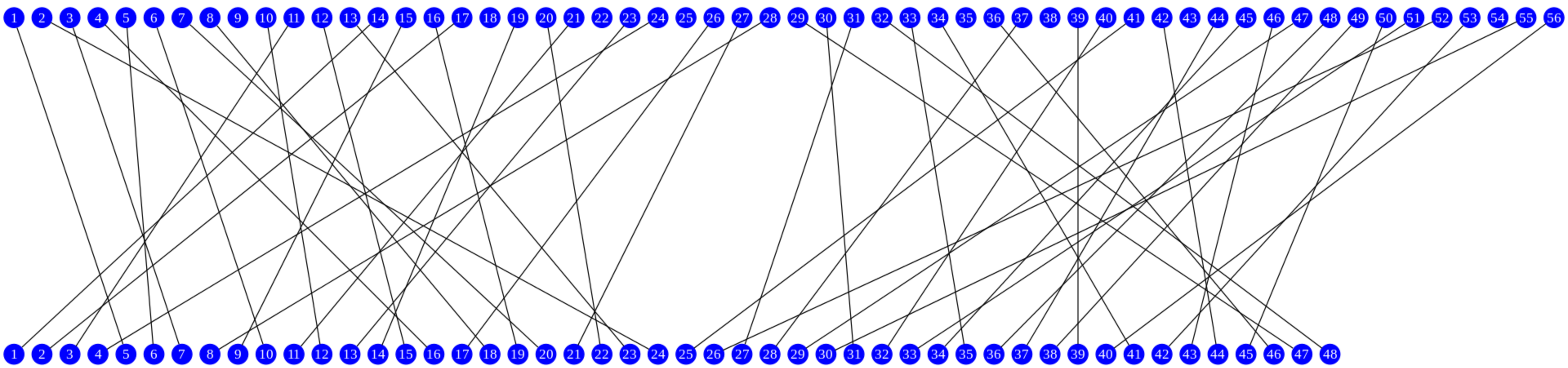
- **1.4 Form the keys K_n, for each $1<=n<=16$, by applying the following permutation table to each of the concatenated pairs C_nD_n.**
- Each pair has 56 bits, but PC-2 only uses 48 of these.

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

The first bit of the output is taken from the 14th bit of the input…

Form the keys K_n

- **1.4 Form the keys K_n, for each $1<=n<=16$, by applying the following permutation table to each of the concatenated pairs C_nD_n.**
- Each pair has 56 bits, but PC-2 only uses 48 of these.



Example (Cont.)

For the first key we have

C_1D_1 = 1110000 1100110 0101010 1011111
1010101 0110011 0011110 0011110

which, after we apply the permutation **PC-2**, becomes

K_1 = 000110 110000 001011 101111
111111 000111 000001 110010

Example (Cont.)

K_2 = 011110 011010 111011 011001 110110 111100 100111 100101

K_3 = 010101 011111 110010 001010 010000 101100 111110 011001

K_4 = 011100 101010 110111 010110 110110 110011 010100 011101

K_5 = 011111 001110 110000 000111 111010 110101 001110 101000

K_6 = 011000 111010 010100 111110 010100 000111 101100 101111

K_7 = 111011 001000 010010 110111 111101 100001 100010 111100

Example (Cont.)

K_8 = 111101 111000 101000 111010 110000 010011 101111 111011

K_9 = 111000 001101 101111 101011 111011 011110 011110 000001

K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111

K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110

K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001

Example (Cont.)

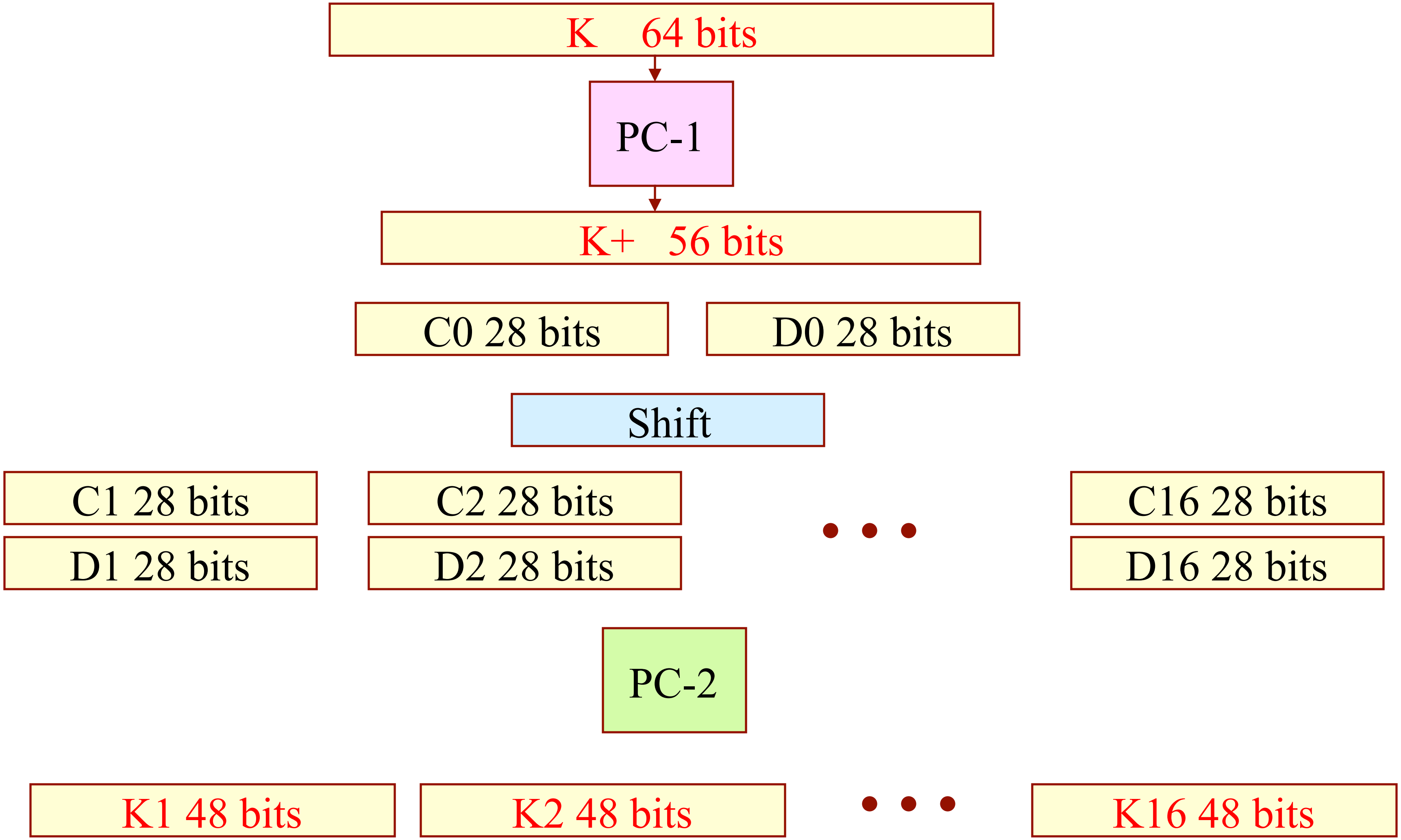
K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001

K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010

K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010

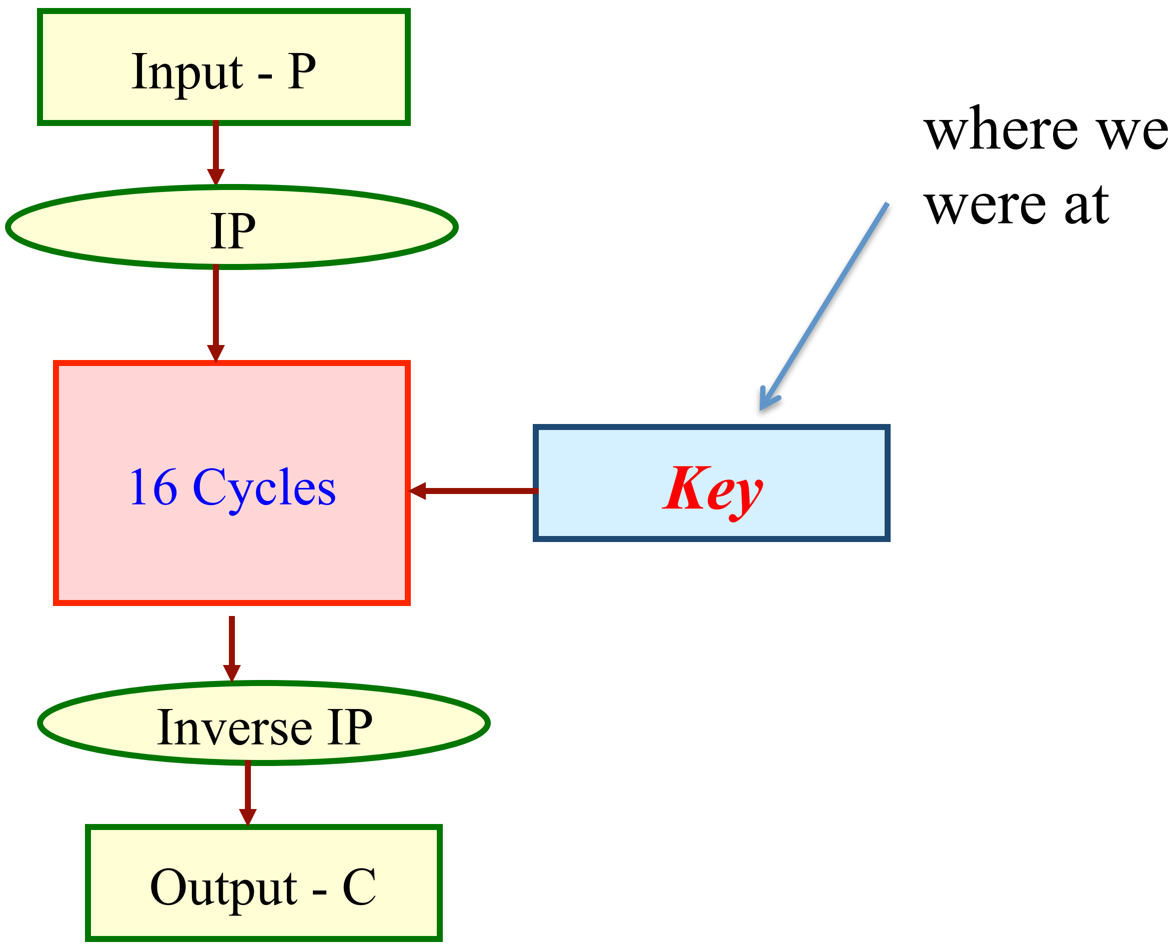
K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

Key Summary

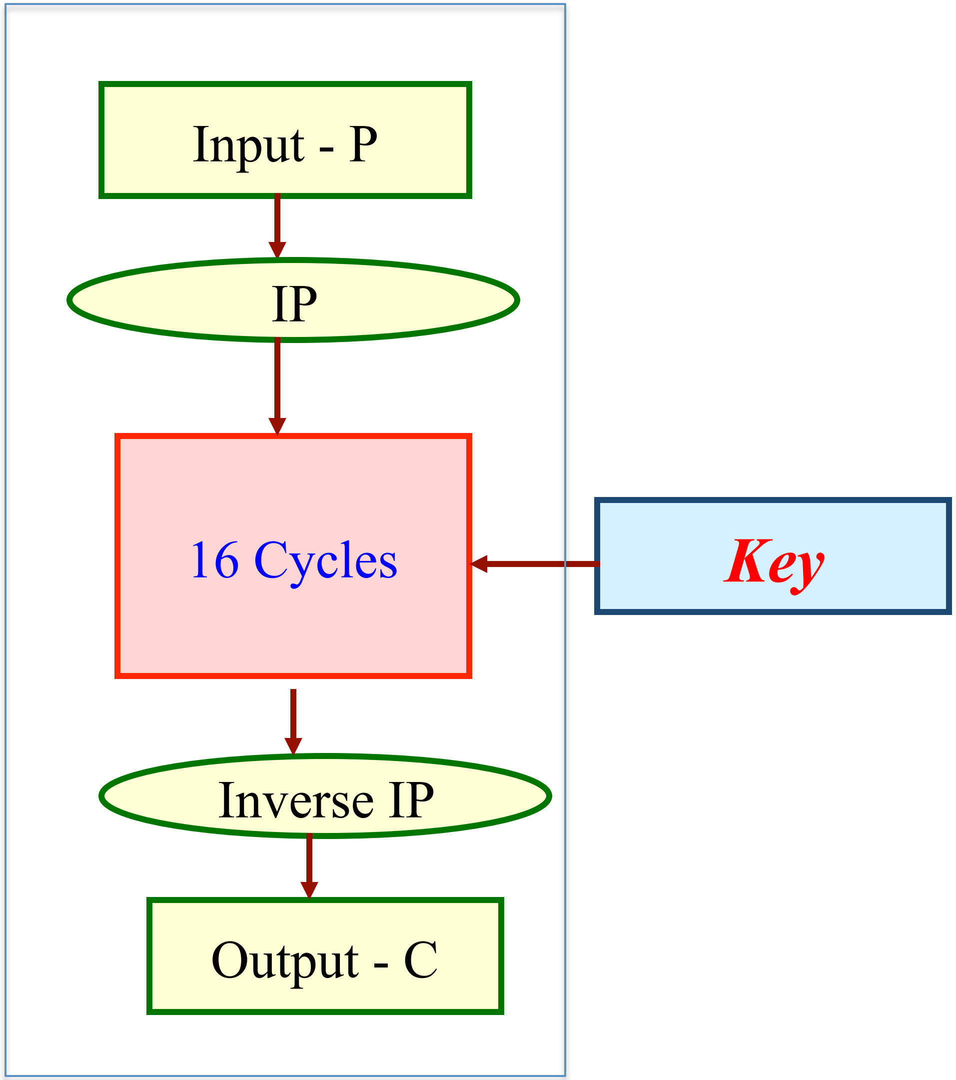


K 64 bits
PC-1
K+ 56 bits
C0 28 bits
D0 28 bits
Shift
C1 28 bits
D1 28 bits
C2 28 bits
D2 28 bits
• • •
C16 28 bits
D16 28 bits
PC-2
K1 48 bits
K2 48 bits
• • •
K16 48 bits

A High Level Description of DES

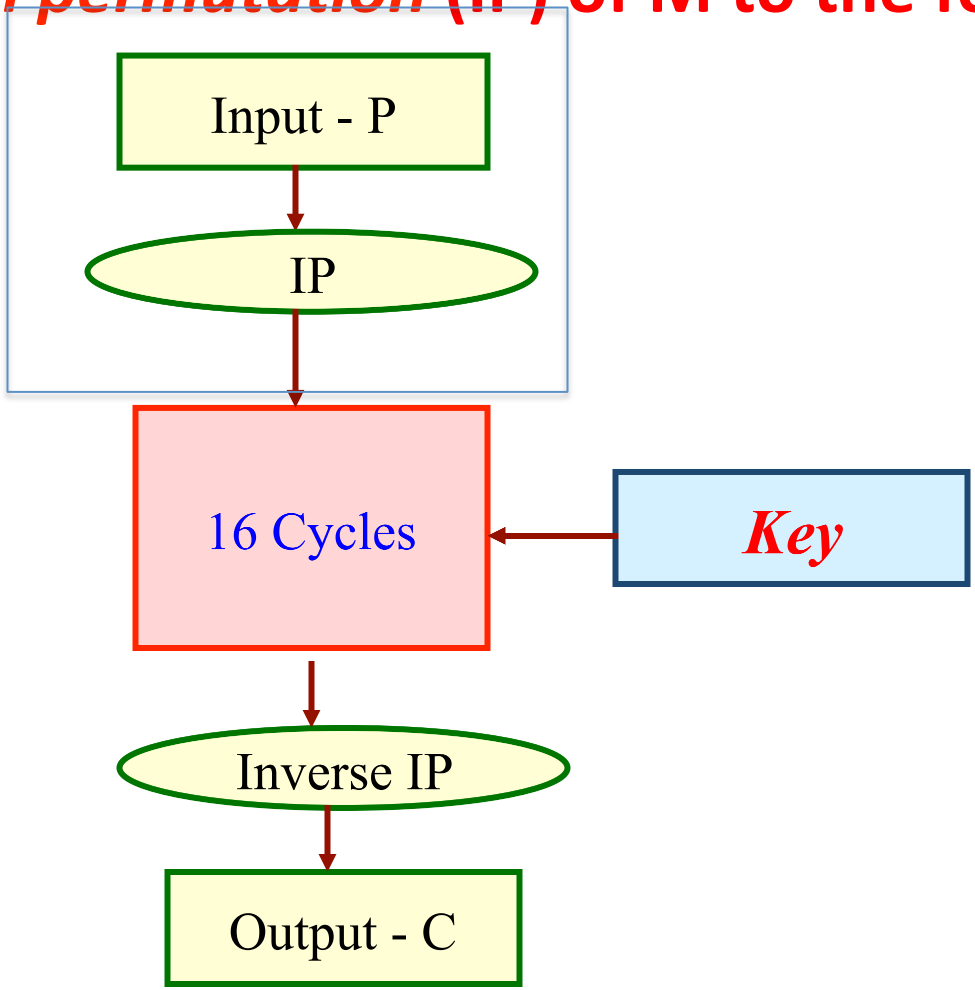


A High Level Description of DES



Step 2: Encode each 64-bit block of data

2.1 Do *initial permutation* (IP) of M to the following IP table.



Step 2: Encode each 64-bit block of data

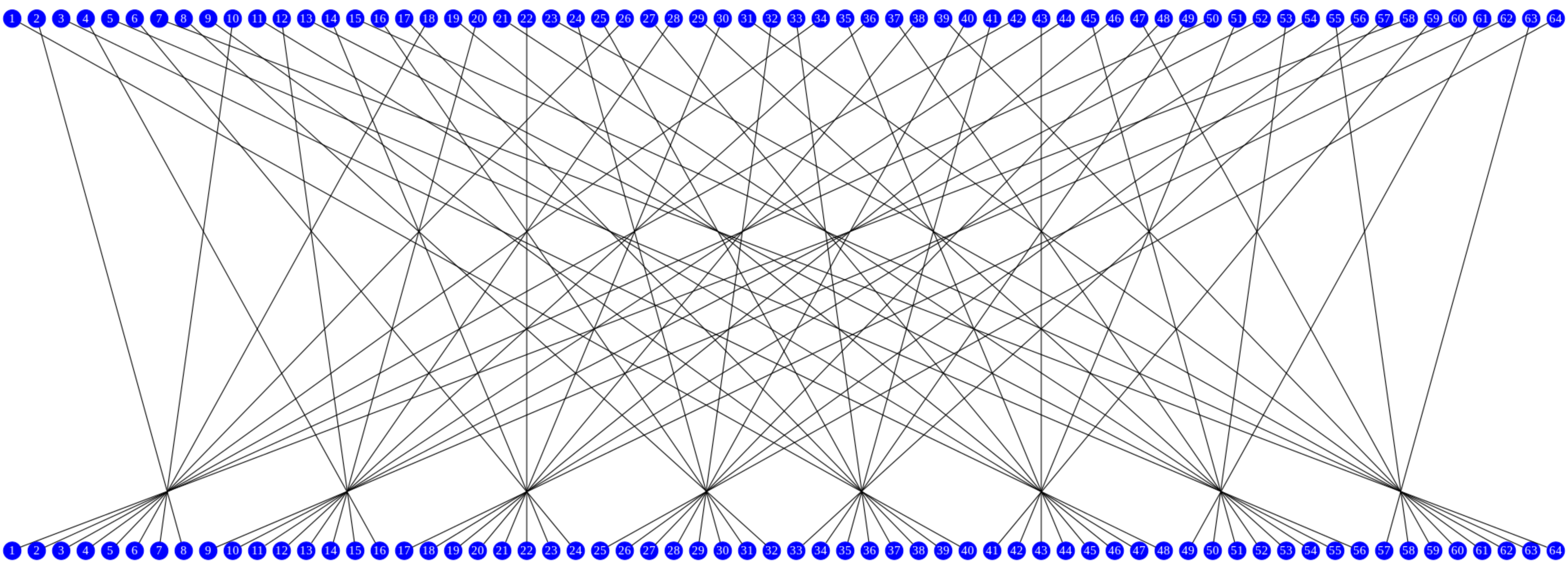
2.1 Do *initial permutation* IP of M to the following IP table.

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

The first bit of the output is taken from the 58th bit of the input…

Step 2: Encode each 64-bit block of data

2.1 Do *initial permutation* IP of M to the following IP table.



The first bit of the output is taken from the 58th bit of the input…

Example (Cont.)

Applying the initial permutation to the block of text M, we get

M =

0000 0001 0010 0011 0100 0101 0110 0111

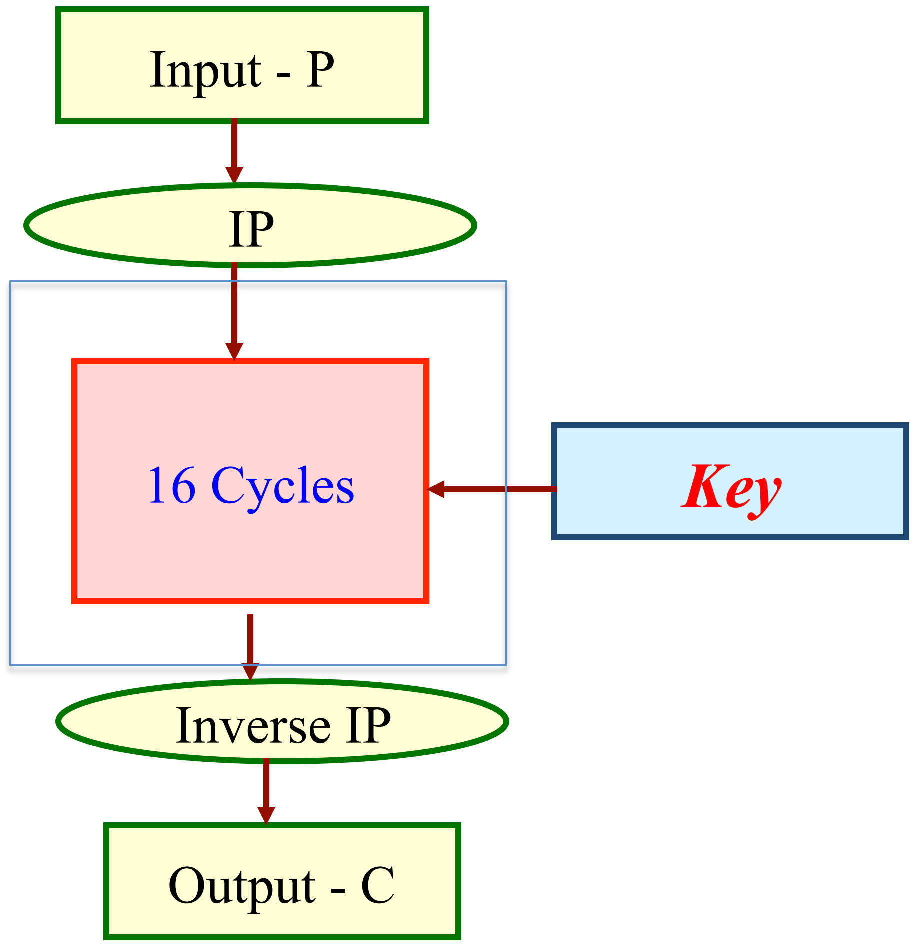
1000 1001 1010 1011 1100 1101 1110 1111

IP =

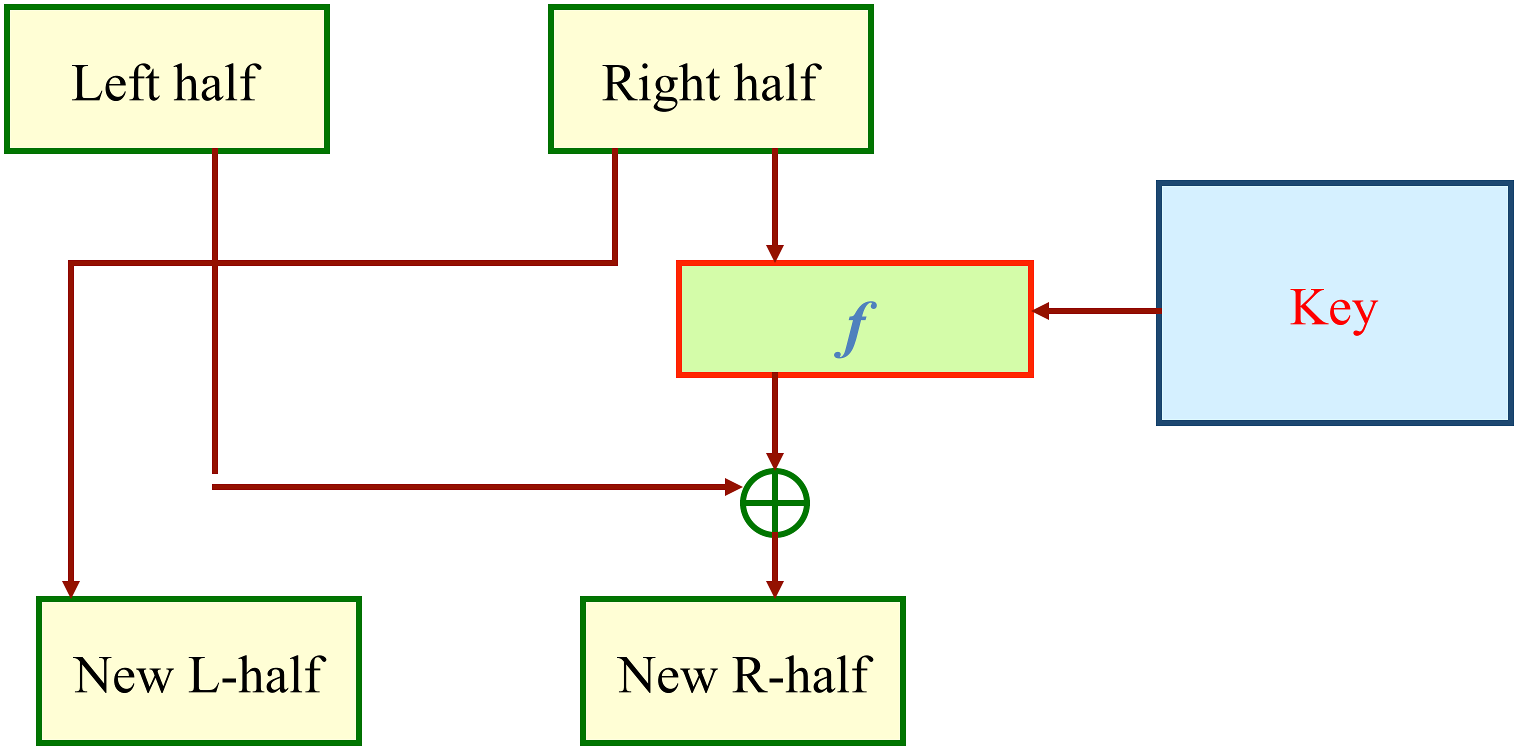
1100 1100 0000 0000 1100 1100 1111 1111

1111 0000 1010 1010 1111 0000 1010 1010

Proceed through 16 iterations of *f*



A Cycle in DES



Divide the permuted block IP

2.2 Divide the permuted block IP into a left half L_0 of 32 bits, and a right half R_0 of 32 bits

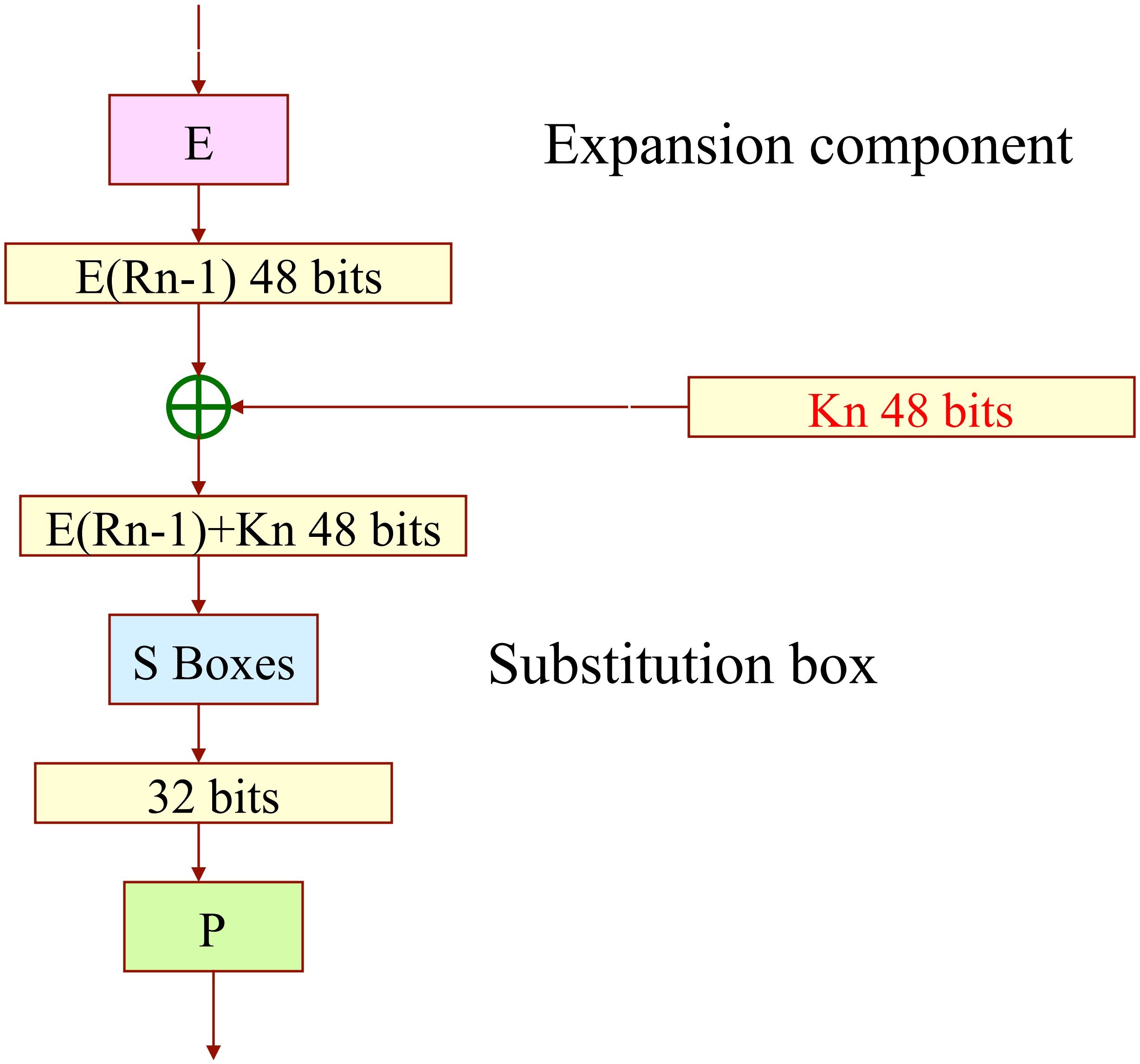
IP = 1100 1100 0000 0000 1100 1100 1111 1111
1111 0000 1010 1010 1111 0000 1010 1010

From IP we get

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

f



Expansion component

Substitution box

Expand each block R_{n-1}

- **2.4 Expand each block R_{n-1} from 32 bits to 48 bits using a permuation table that repeats some of the bits in R_{n-1} .**

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

Example (Cont.)

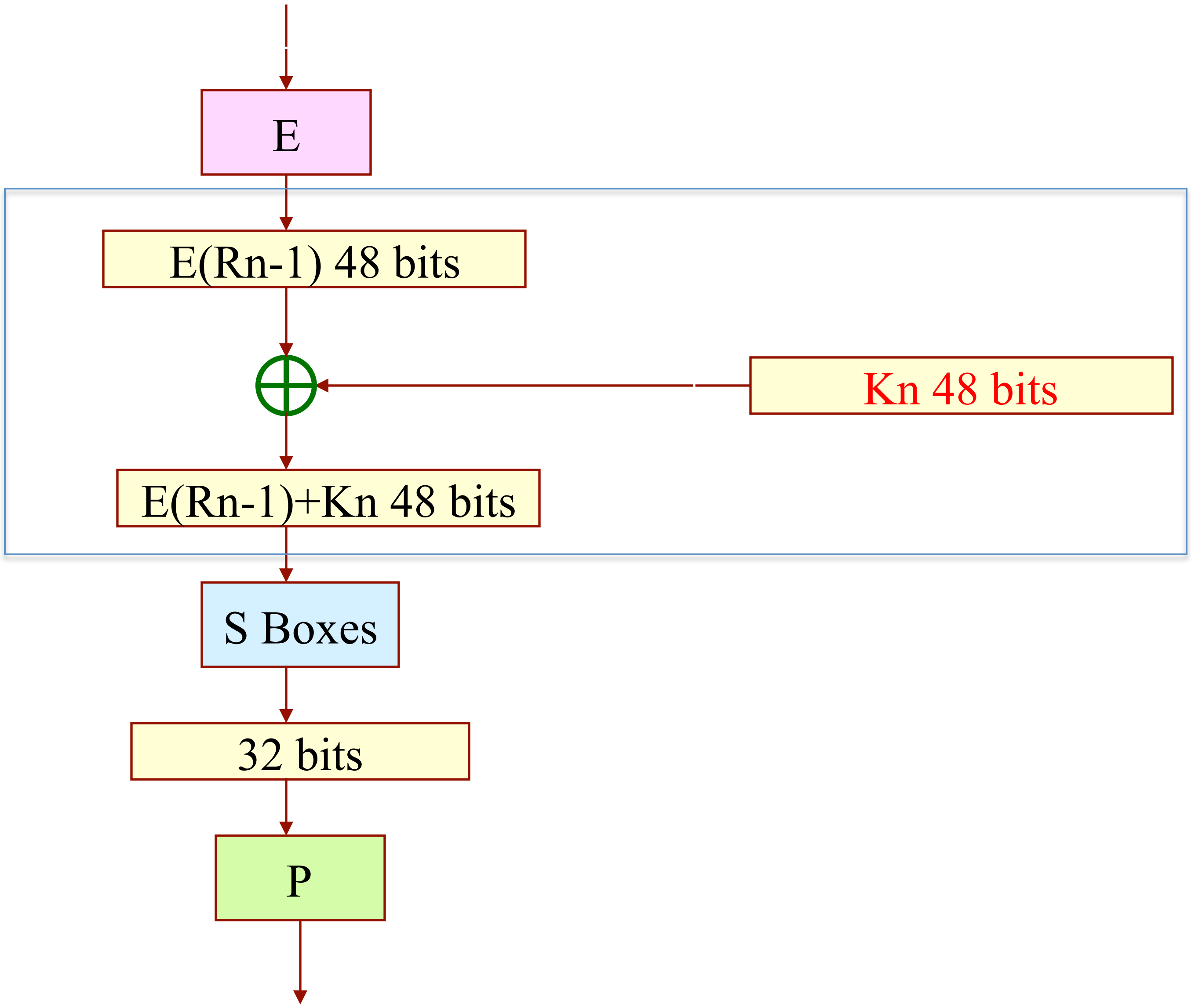
We calculate $E(R_0)$ from R_0 as follows:

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

$E(R_0)$ = 011110 100001 010101 010101
011110 100001 010101 010101

Note that each block of 4 original bits has been expanded to a block of 6 output bits.

f



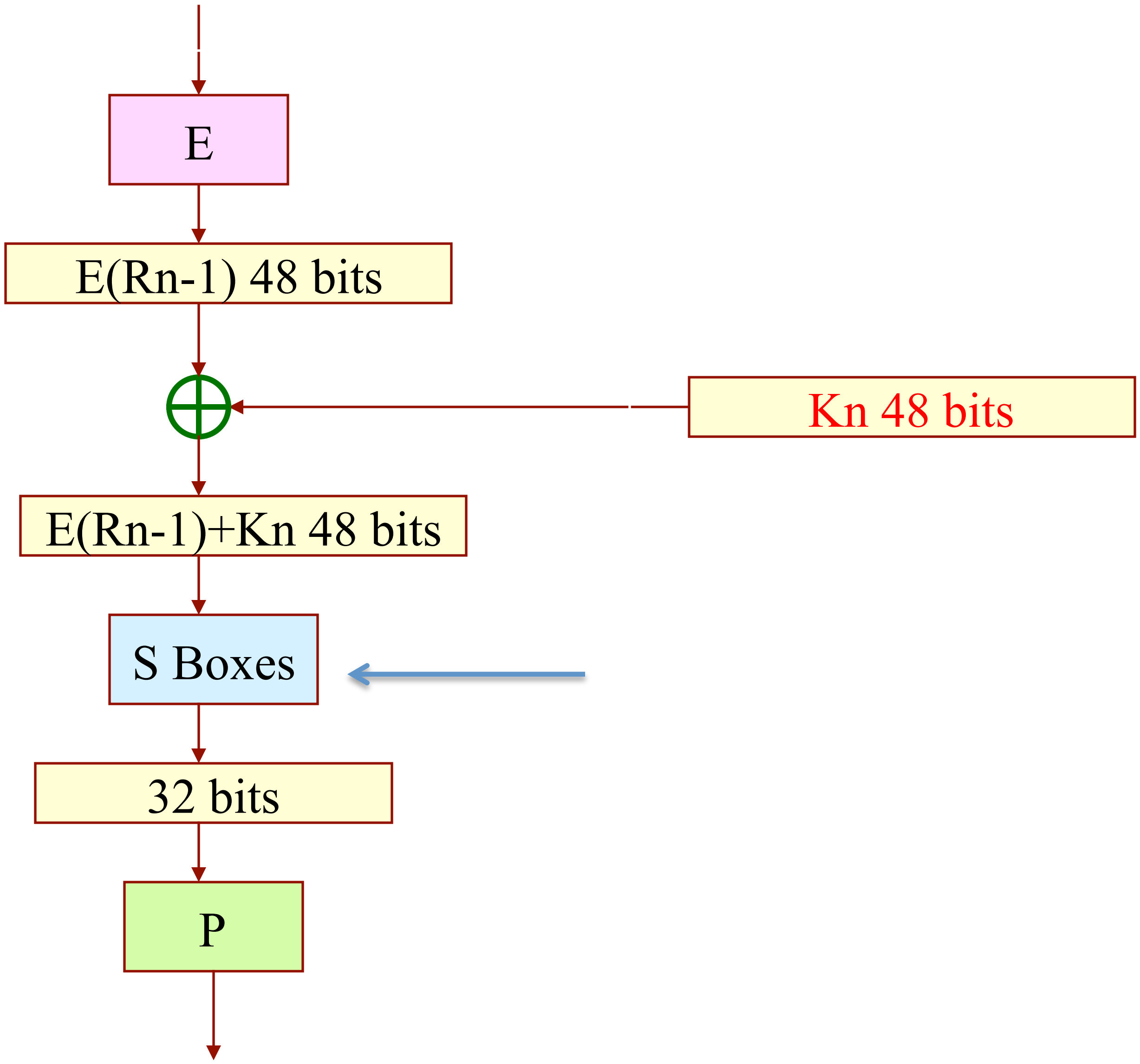
XOR Operation

- In the ***f*** calculation, we XOR the output $\mathbf{E}(\boldsymbol{R_{n-1}})$ with the key $\boldsymbol{K_n}$:

$$\boldsymbol{K_n} + \mathbf{E}(\boldsymbol{R_{n-1}})$$

$$\begin{aligned}
\boldsymbol{K_1} &= 000110\ 110000\ 001011\ 101111 \\
&\quad\ 111111\ 000111\ 000001\ 110010 \\
\mathbf{E}(\boldsymbol{R_0}) &= 011110\ 100001\ 010101\ 010101 \\
&\quad\ 011110\ 100001\ 010101\ 010101 \\
\boldsymbol{K_1}+\mathbf{E}(\boldsymbol{R_0}) &= 011000\ 010001\ 011110\ 111010 \\
&\quad\ 100001\ 100110\ 010100\ 100111
\end{aligned}$$

f



Substitution – S-Boxes (Cont.)

K_1+E(R_0) = 011000 010001 011110 111010
100001 100110 010100 100111

- K_n + E(R_{n-1}) = $B_1B_2B_3B_4B_5B_6B_7B_8$

where each B_i is a group of six bits.

We now calculate

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$

where $S_i(B_i)$ referrers to the output of the ***i***-th **S** box.

Substitution – S-Boxes (Cont.)

Box S1

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

Finding S1(B1)

- The first and last bits of B represent in base 2 a number in the decimal range 0 to 3.
  - Let that number be i.
- The middle 4 bits of B represent in base 2 a number in the decimal range 0 to 15.
  - Let that number be j.
- Look up in the table the number in the i-th row and j-th column.

Example (Cont.)

- For input block B = 011011 the first bit is "0" and the last bit "1" giving 01 as the row.
  - This is row 1.
- The middle four bits are "1101".
  - This is the binary equivalent of decimal 13, so the column is column number 13.
- In row 1, column 13 appears 5. This determines the output;
  - 5 is binary 0101, so that the output is 0101.
- Hence S1(011011) = 0101.

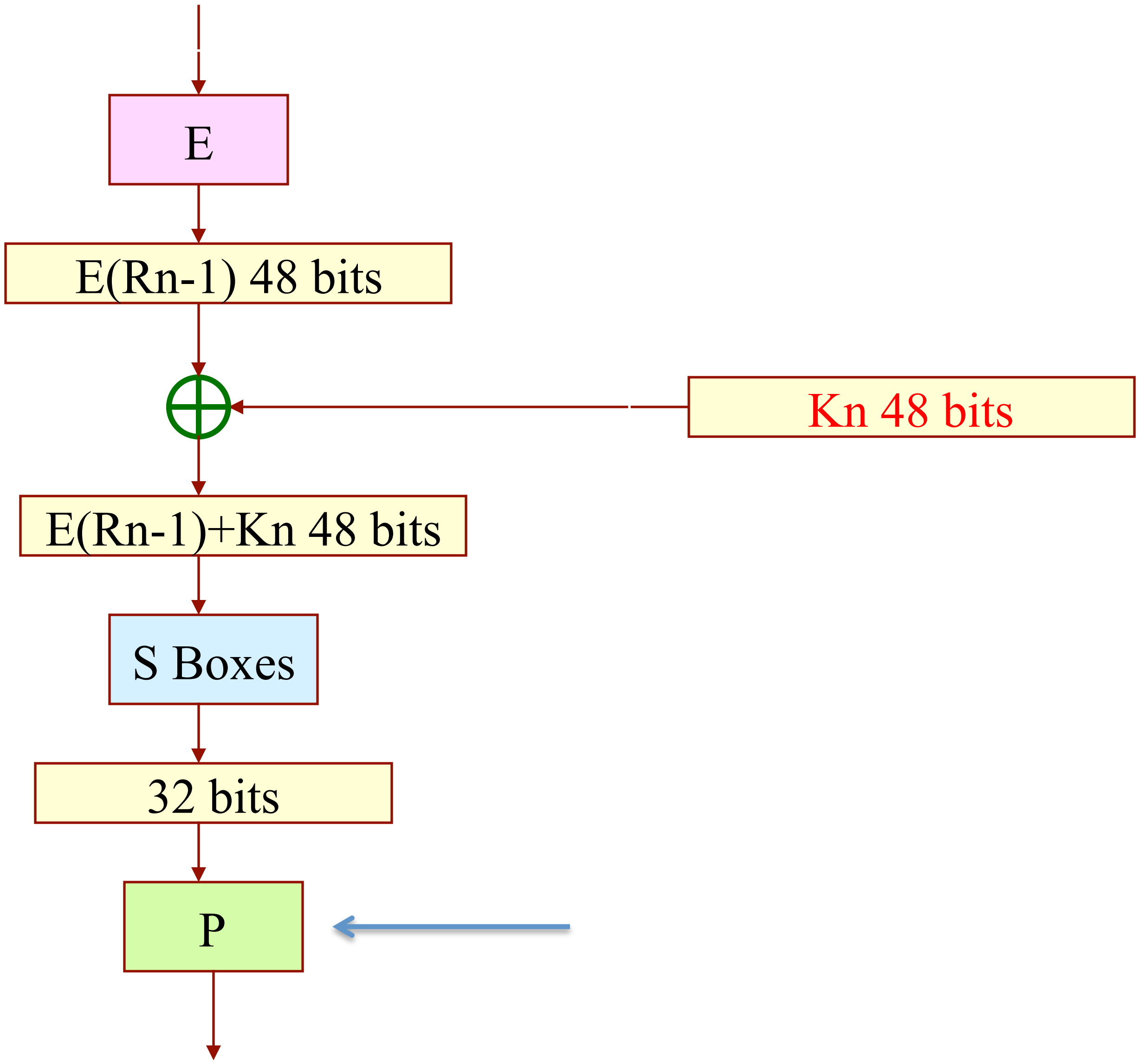
Example (Cont.)

For the first round, we obtain as the output of the eight **S** boxes:

$\boldsymbol{K_1} + \mathbf{E}(\boldsymbol{R_0})$ = 011000 010001 011110 111010 100001 100110 010100 100111

$\boldsymbol{S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)}$ =
0101 1100 1000 0010 1011 0101 1001 0111

Permutation P of the S-box output



Permutation P of the S-box output

- $f = \mathrm{P}(S_1(B_1)S_2(B_2)\ldots S_8(B_8))$

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

Example (Cont.)

From the output of the eight **S** boxes:

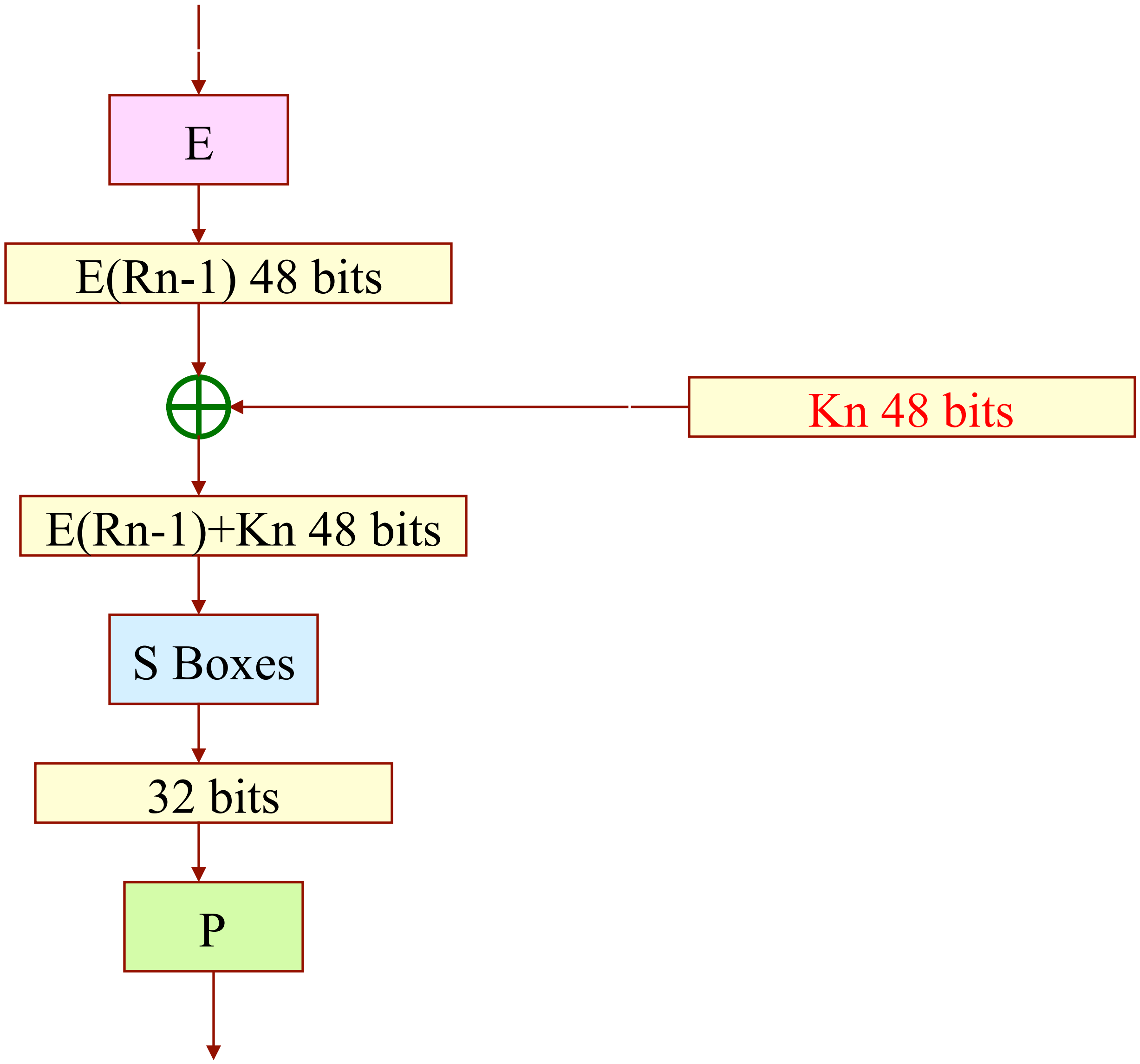
$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ =

0101 1100 1000 0010 1011 0101 1001 0111

we get

f = 0010 0011 0100 1010 1010 1001 1011 1011

f



Final Phase

- At the end of the sixteenth round we have $\boldsymbol{L_{16}}$ and $\boldsymbol{R_{16}}$. We then ***reverse*** the order of the two blocks into $\boldsymbol{R_{16}L_{16}}$ and apply a final permutation $\mathbf{IP^{-1}}$

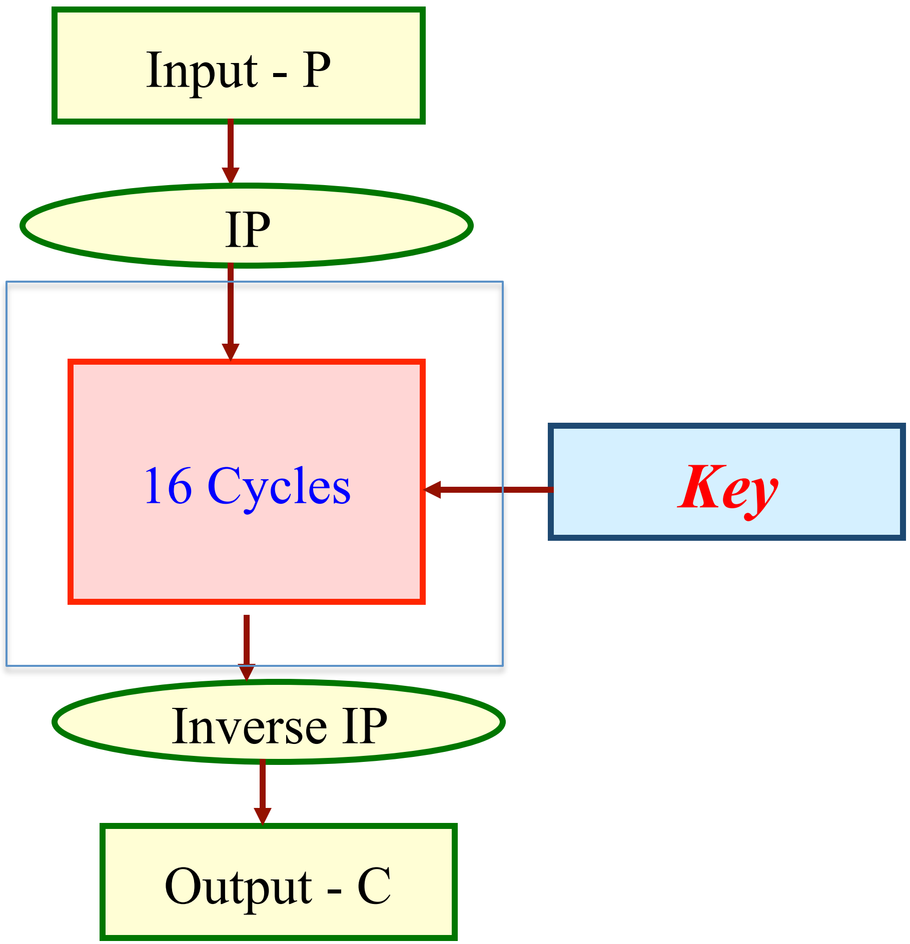
Example (cont.)

- If we process all 16 blocks using the method defined previously, we get, on the 16th round,

L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100

R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101

$R_{16}L_{16}$ = 00001010 01001100 11011001 100101


Input - P
IP
16 Cycles
Key
Inverse IP
Output - C

Final Phase

- At the end of the sixteenth round we have L_{16} and R_{16}. We then ***reverse*** the order of the two blocks into $R_{16}L_{16}$ and apply a final permutation **IP^{-1}** as defined by the following table

40	8	48	16	56	24	64	31
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

Example (cont.)

- We reverse the order of these two blocks and apply the final permutation to

$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

which in hexadecimal format is

85E813540F0AB405

The End

M = 0123456789ABCDEF

C = 85E813540F0AB405

- Decryption is simply the inverse of encryption, following the same steps as above, but reversing the order in which the sub-keys are applied

Overview