Data Encryption Standard (DES)

Dr. B C Dhara
Department of Information Technology
Jadavpur University

Objectives

- Define the basic structure of DES
- Describe the details of building elements of DES
- Describe the round keys generation process
- Analyze DES

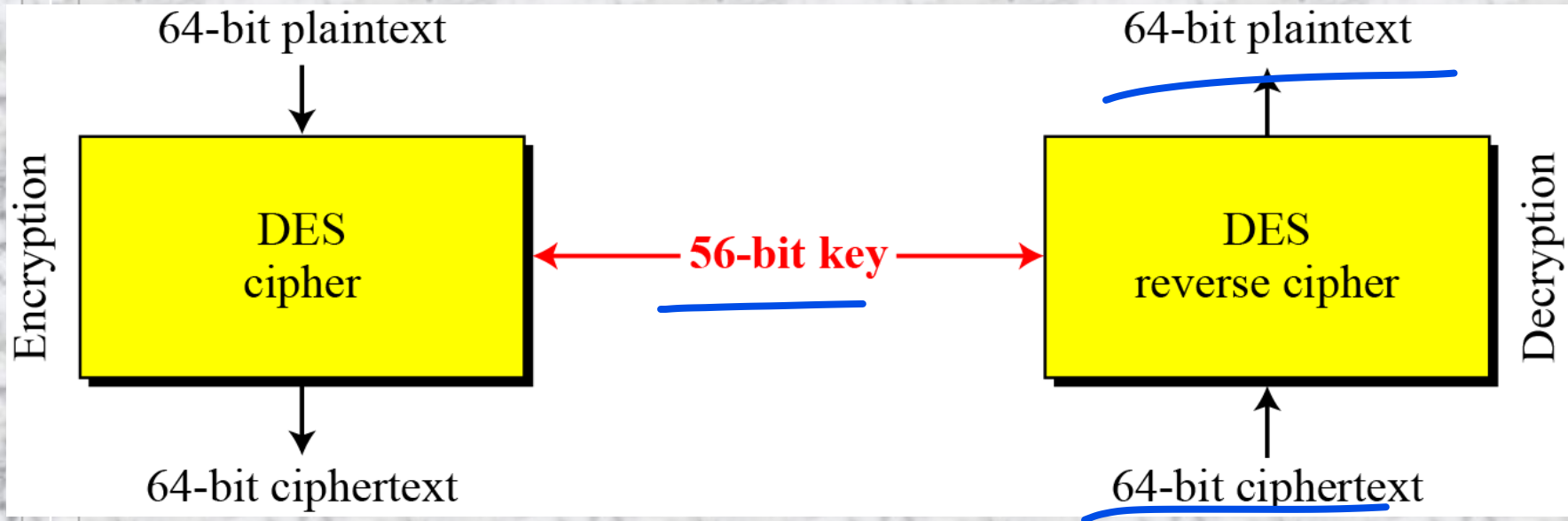
Introduction

- *The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).*
- *In 1973, NIST published a request for proposals for a national symmetric-key cryptosystem.*
  - *A proposal from IBM, a modification of a project called Lucifer, was accepted as DES.*
  - *DES was published in the Federal Register in March 1975 as a draft of the Federal Information Processing Standard (FIPS).*

Introduction (contd…)

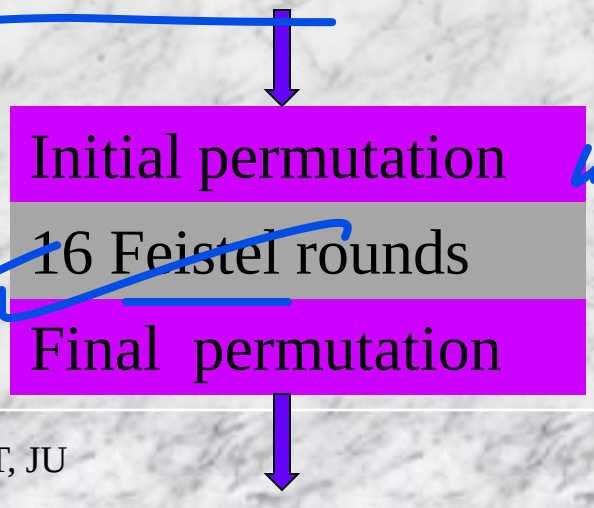
- The draft was criticized for:
  - Small key length (56 bits), vulnerable to brute-force attack
  - Hidden design structure behind the internal structure of DES
- DES was finally published as FIPS 46 in the Federal Register in January 1977
  - DES as standard for use in unclassified applications
- Later a new standard FIPS-3 recommended the use of triple DES

Encryption and decryption with DES

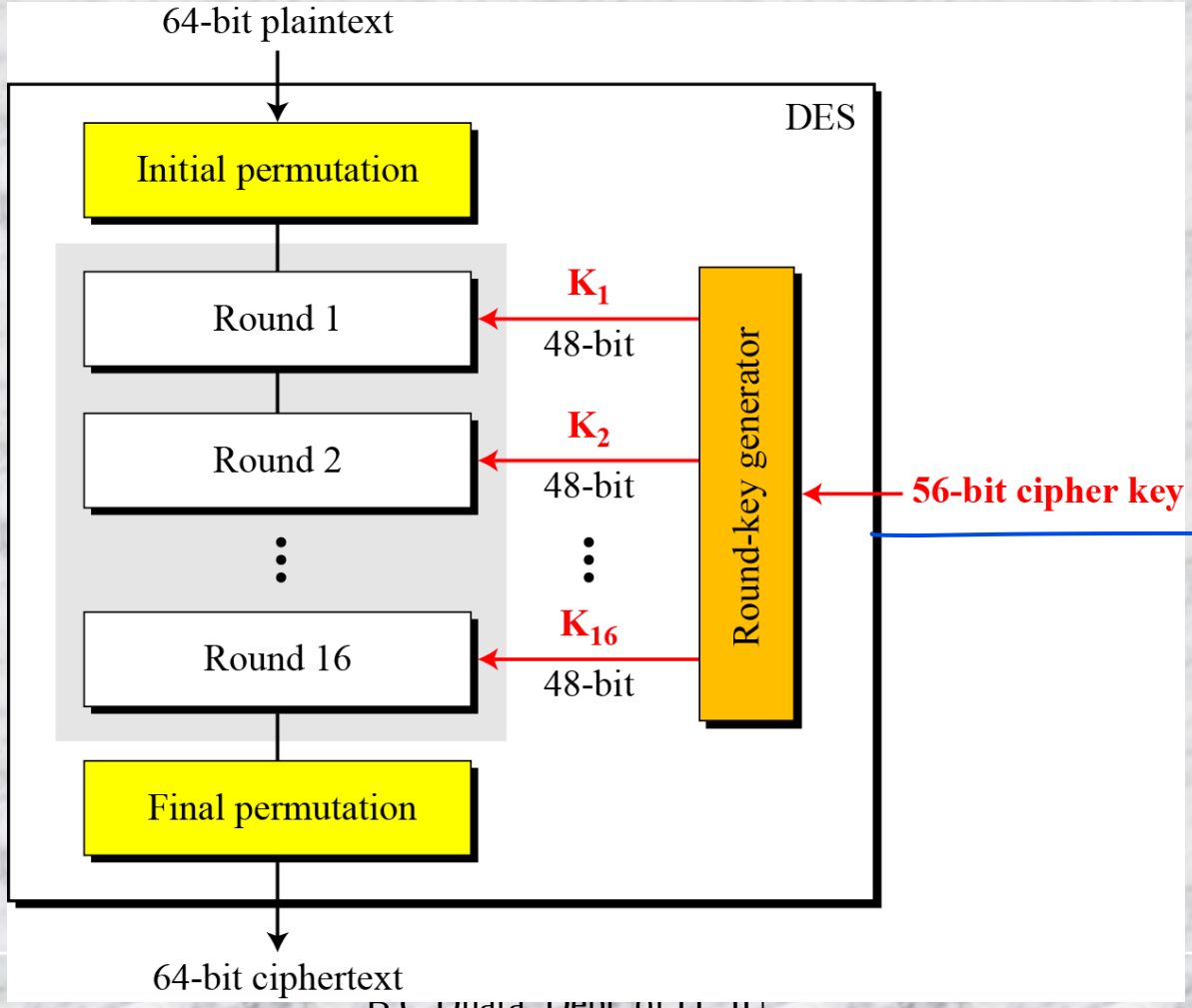


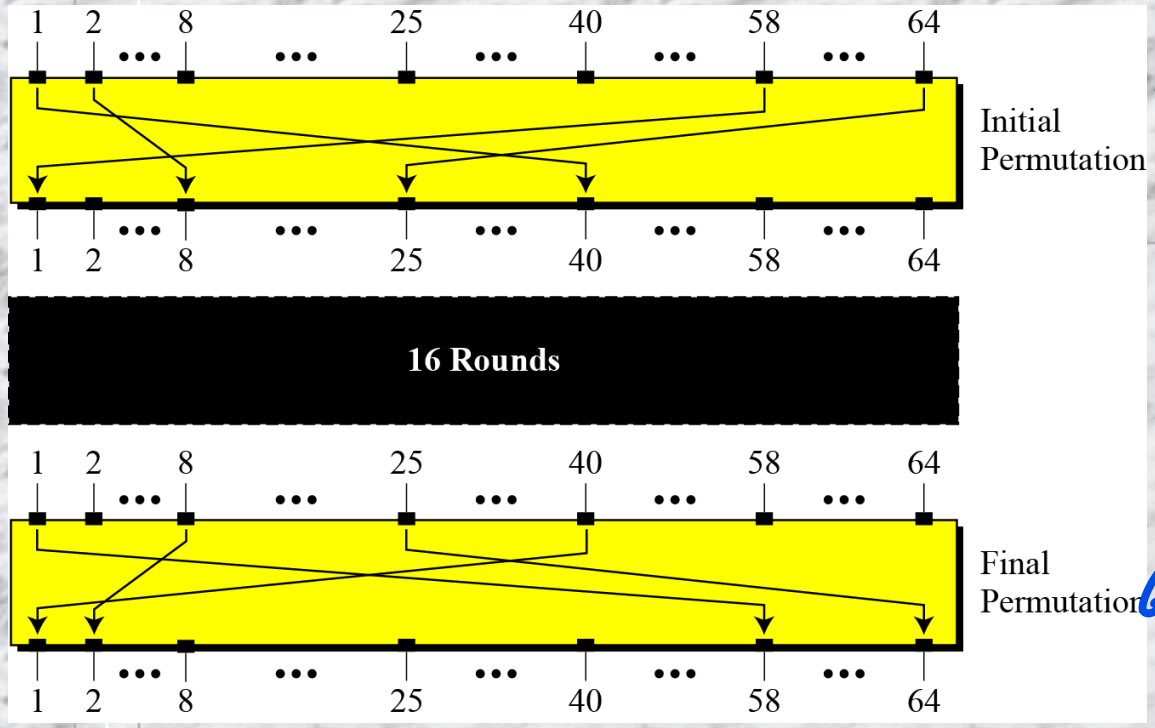
DES structure

- Discuss encryption process
  - Decryption is in reverse order
- *The encryption process is made of two permutations (P-boxes), which we call initial and final permutations, and sixteen Feistel rounds*



General structure of DES





Initial and Final Permutations

64 x 1 = 64-bit

Initial Permutation							
58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

Final Permutation							
40	08	48	16	56	24	64	32
39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25

Initial and Final Permutations (contd…)

- These two permutations (straight P-boxes) have no cryptography significance in DES
- Both are keyless, predetermined and inverse of each other
- The purpose to include these is not clear

Example: initial permutation

- Find the output of the final permutation box when the input is given in hexadecimal as:

  0x0000 0080 0000 0002

- Only bit 25 and bit 63 are 1s; the other bits are 0s. In the final permutation, bit 25 becomes bit 64 and bit 63 becomes bit 15. The result is

  0x0002 0000 0000 0001

Example: final permutation

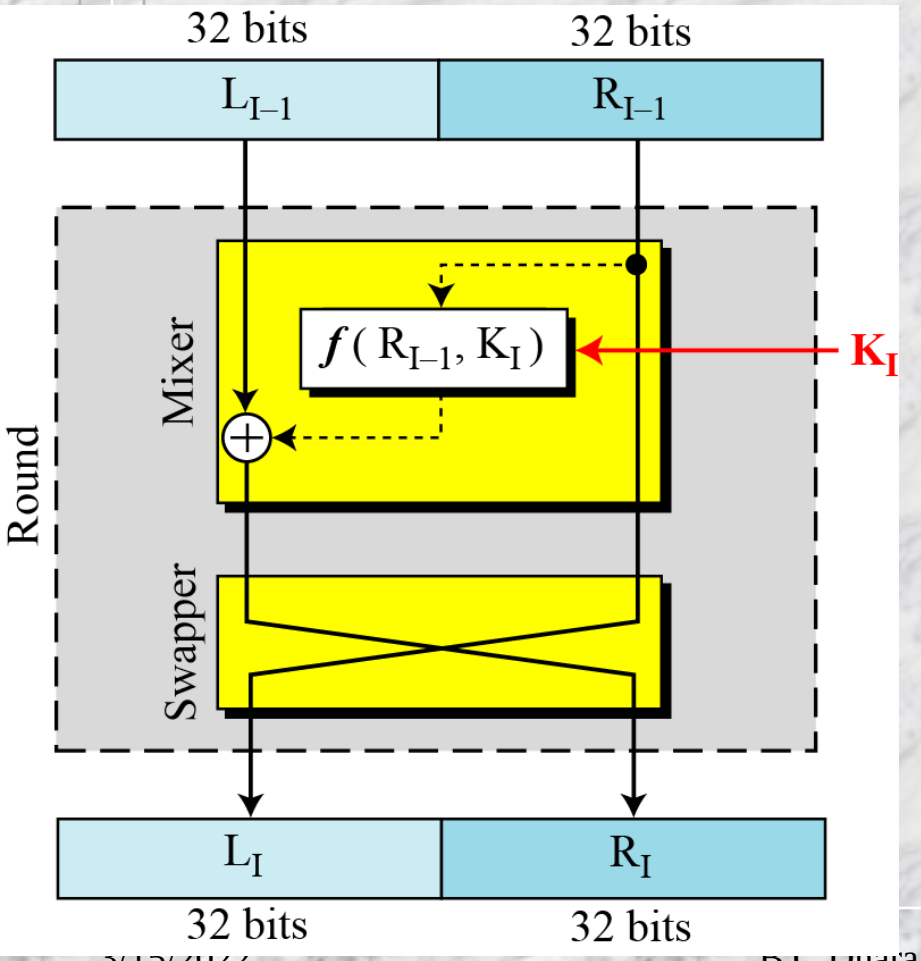
0x0002 0000 0000 0001

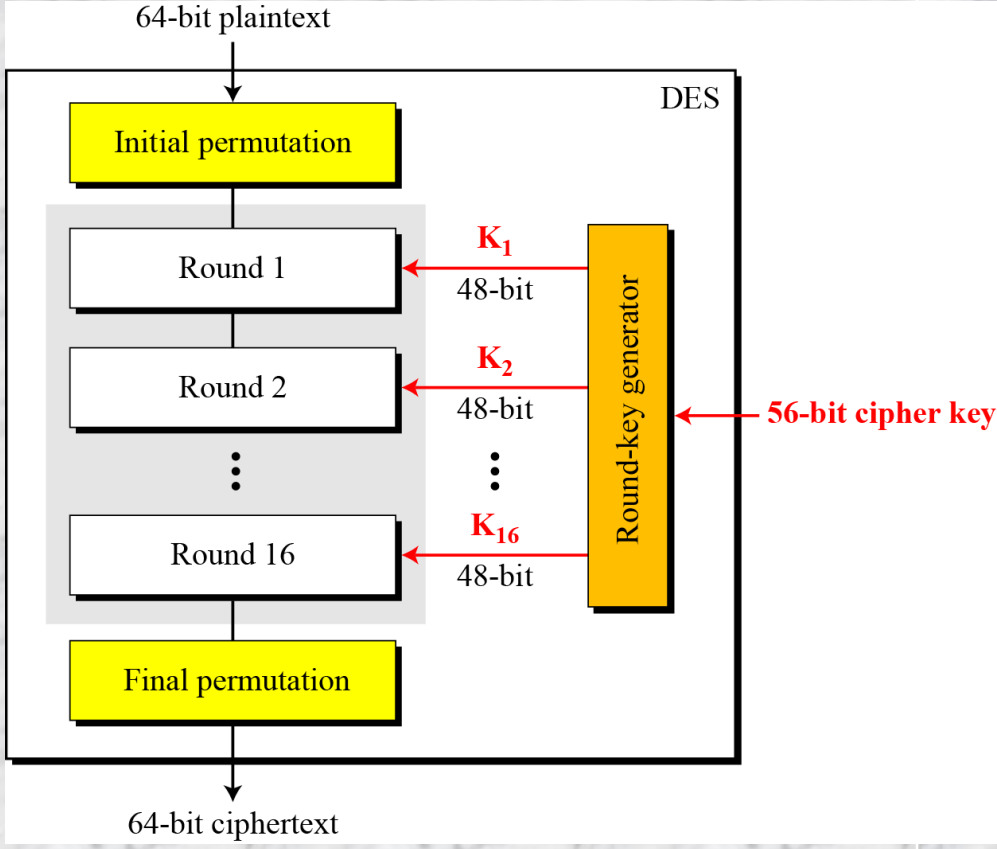
- The input has only two 1s; the output must also have only two 1s.
- Bit 15 in the input becomes bit 63 in the output.
- Bit 64 in the input becomes bit 25 in the output.

0x0000 0080 0000 0002

Rounds

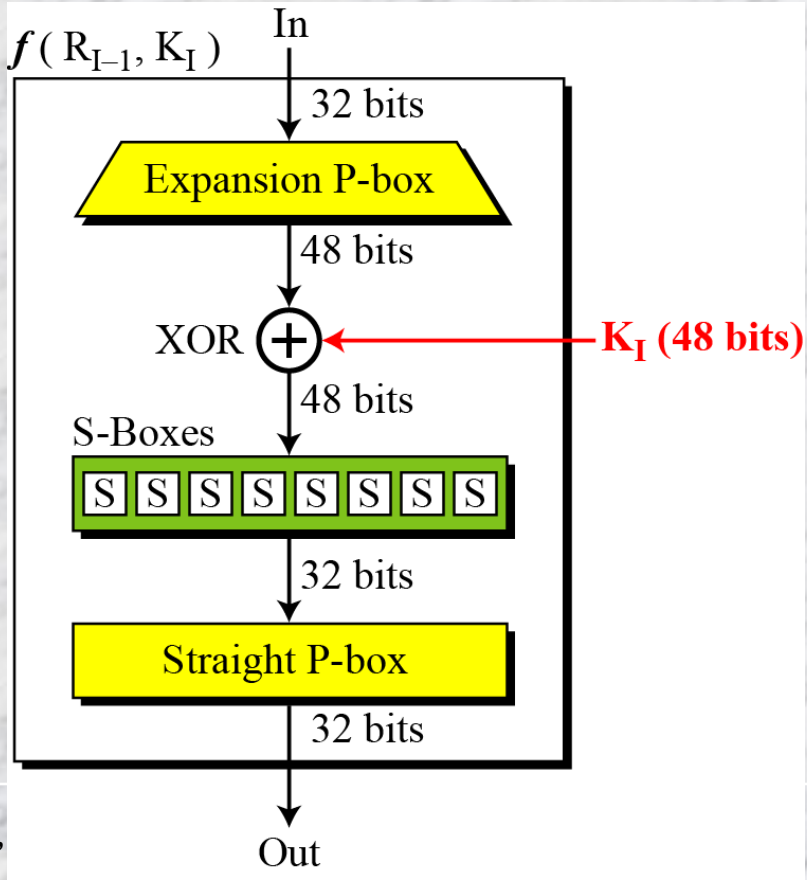
- *DES uses 16 rounds. Each round of DES is a Feistel cipher.*





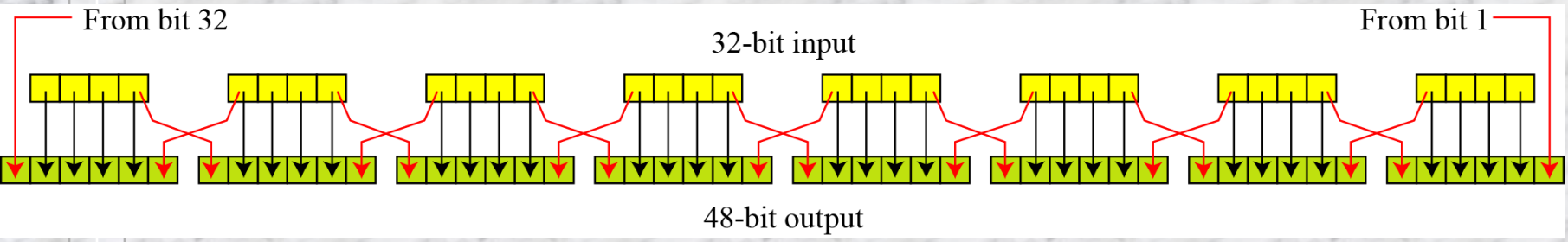
DES Function

- *The heart of DES is the DES function.*
- *The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.*



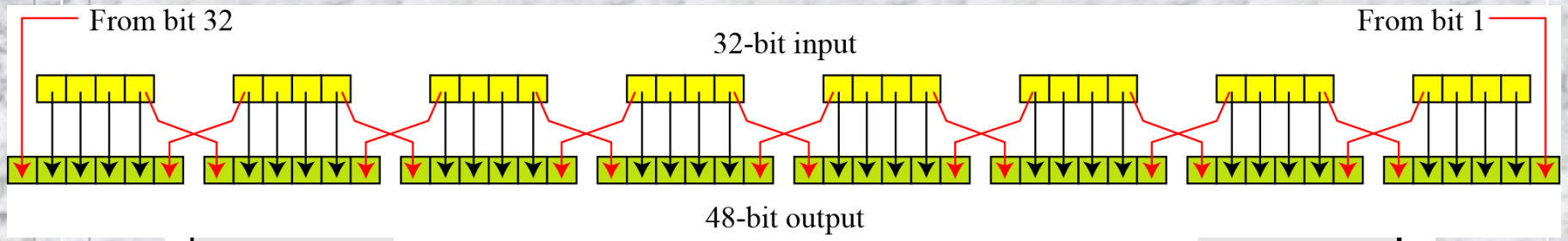
Expansion P-box

- *Since R_{I-1} is a 32-bit input and*
- *K_I is a 48-bit key*
  - *we first need to expand R_{I-1} to 48 bits.*



Expansion P-box

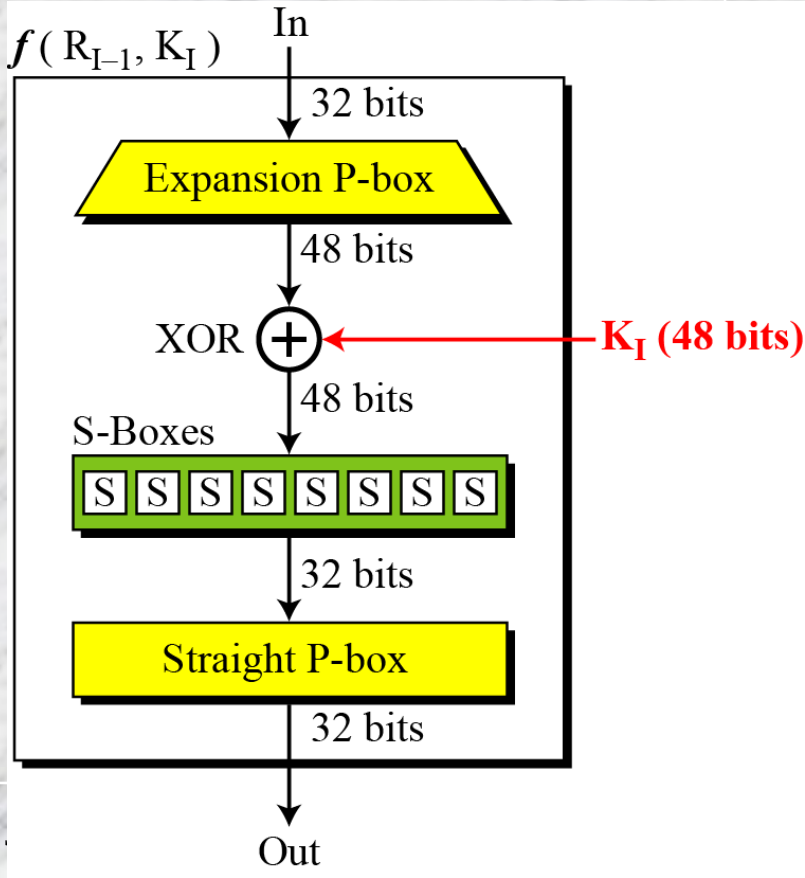
- This P-box can be defined using a table also



32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	31	31	32	01

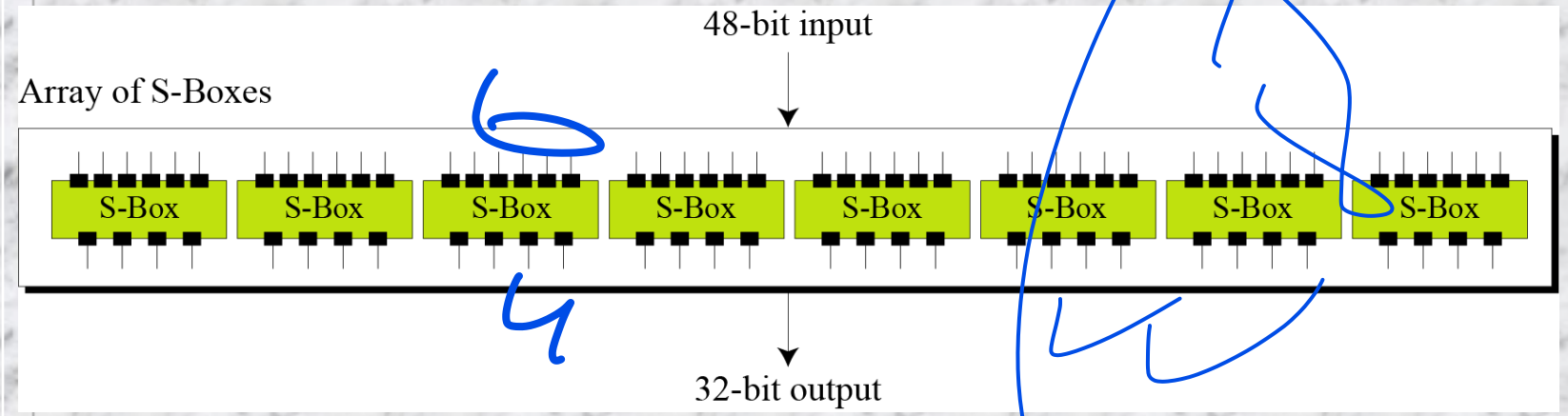
Whitener (XOR)

- *After the expansion permutation, DES uses the XOR operation on the expanded right section and the round key.*



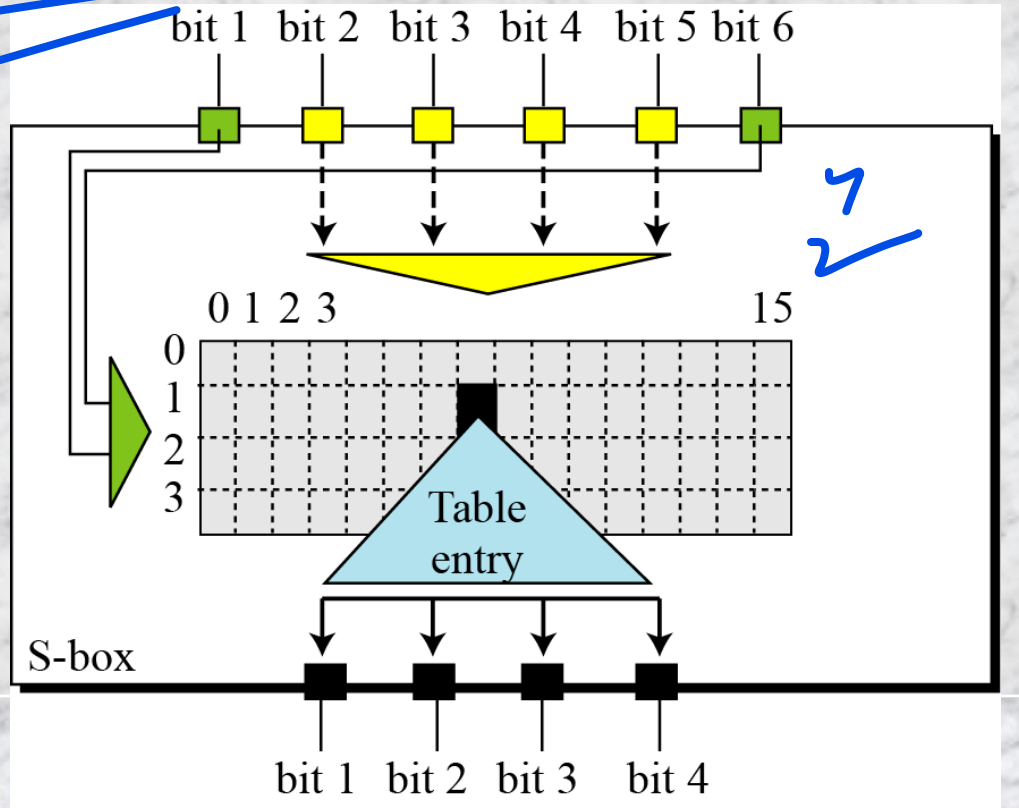
S-boxes

- *The S-boxes do the real mixing (confusion)*
- *DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output*



S-boxes (contd…)

- A table with size 4 × 16 with 4-bits value (0-15)
- Separate table for different S-boxes



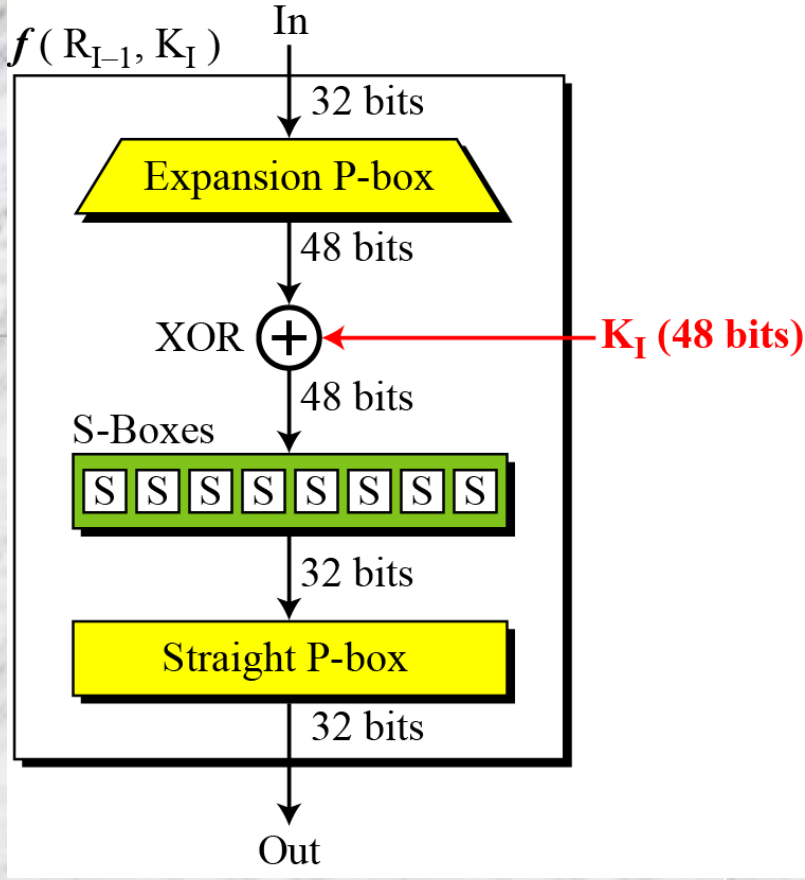
S-boxes (contd…) : S1-box

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Example

- The input to S-box 1 is 100011. What is the output?
- The first and the sixth bits together, we get 11 in binary, which is 3 in decimal.
- The remaining bits are 0001 in binary, which is 1 in decimal.
  - We look for the value in row 3, column 1, in S-box1
  - The result is 12 in decimal, which in binary is 1100. So the input 100011 yields the output 1100.

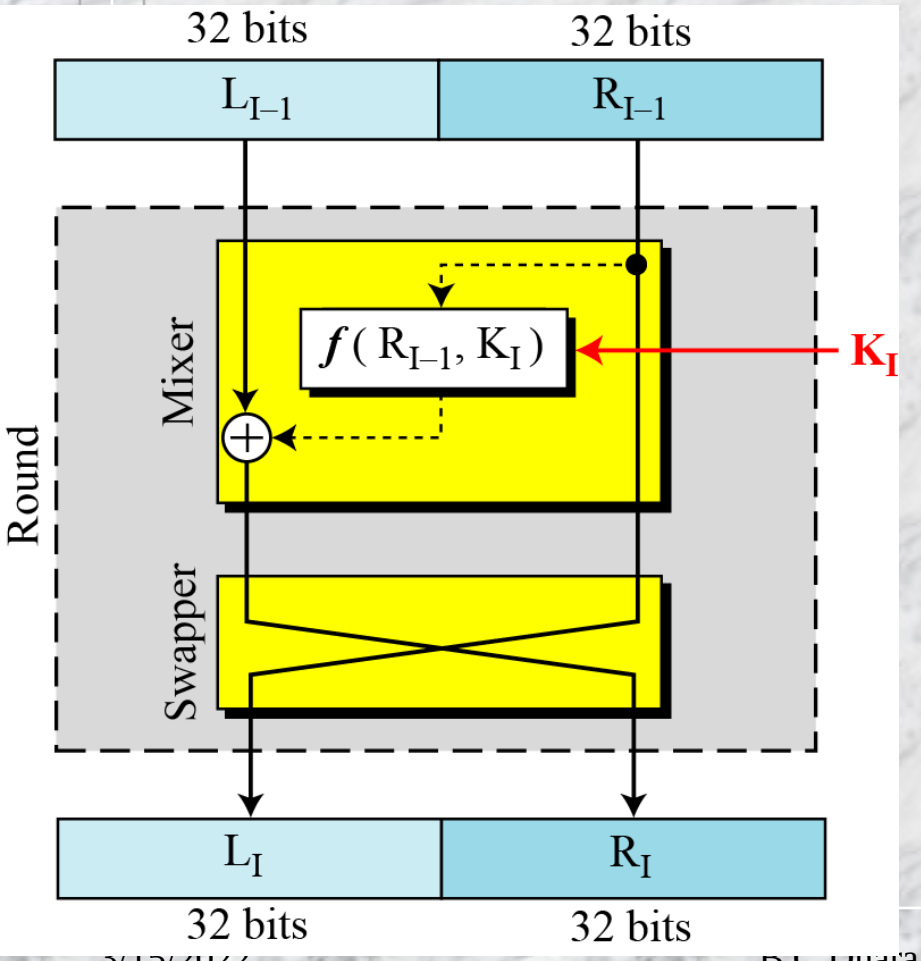
Straight permutation

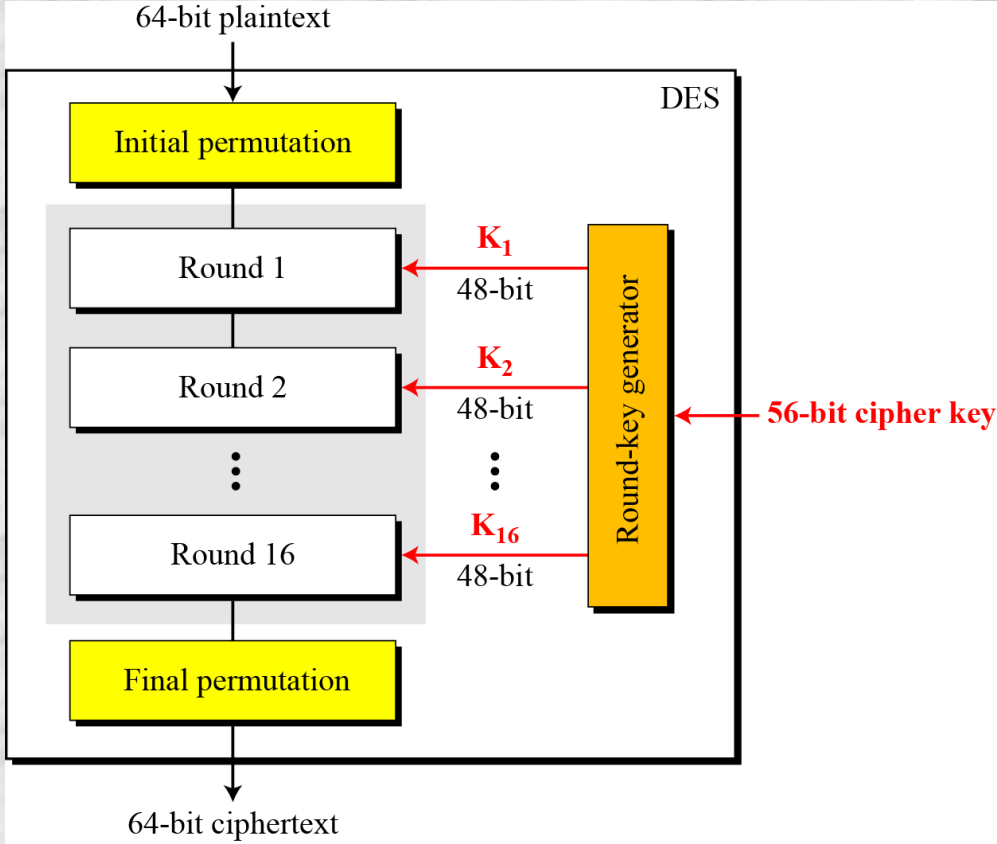


16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Rounds

- *DES uses 16 rounds. Each round of DES is a Feistel cipher.*





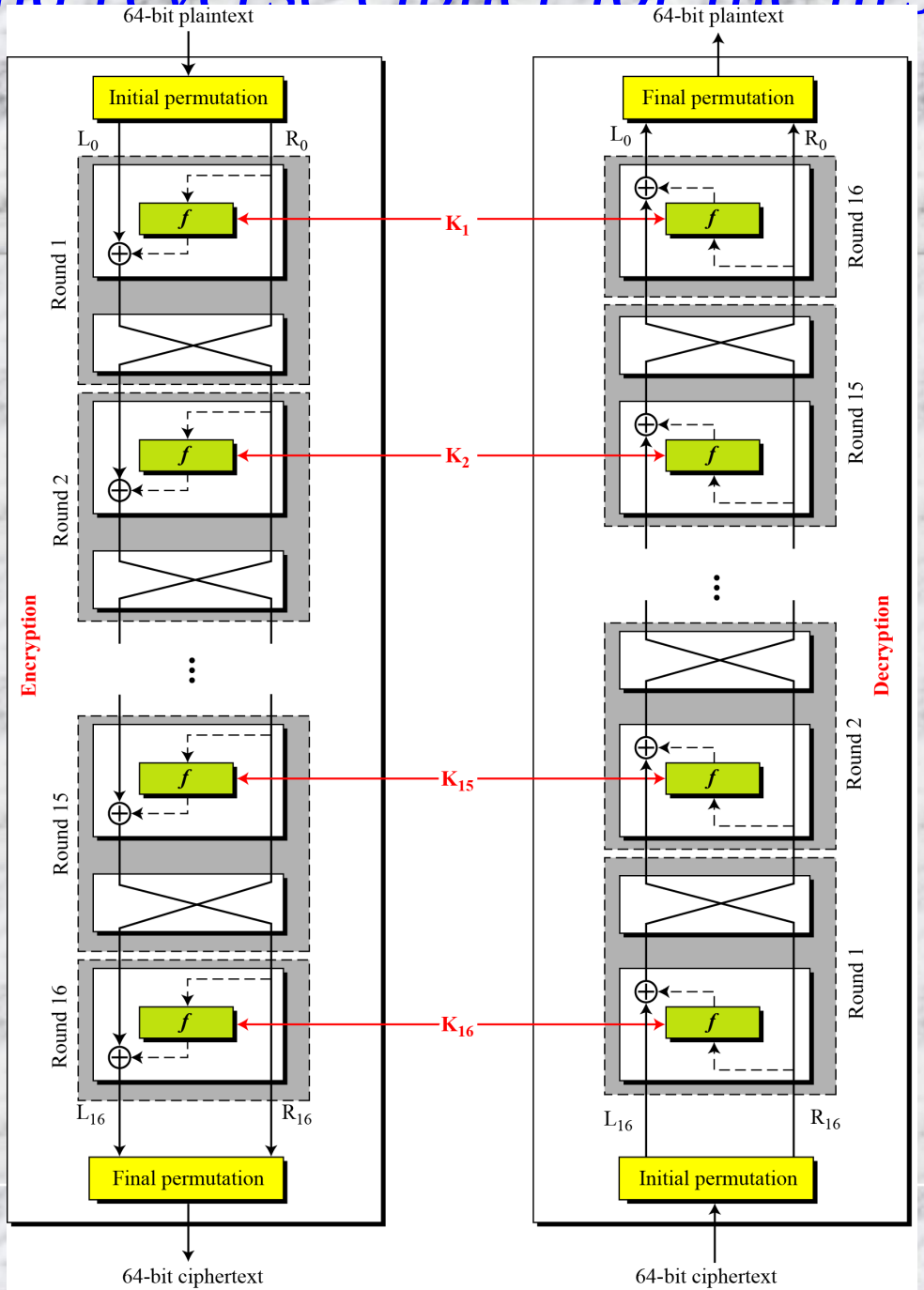
DES implementation

- *Using mixers and swappers, the cipher and reverse cipher is created, each having 16 rounds*

DES implementation: First approach

- *This approach is to make the last round (round 16) different from the others; it has only a mixer and no swapper*

DES cipher and reverse cipher for the first approach

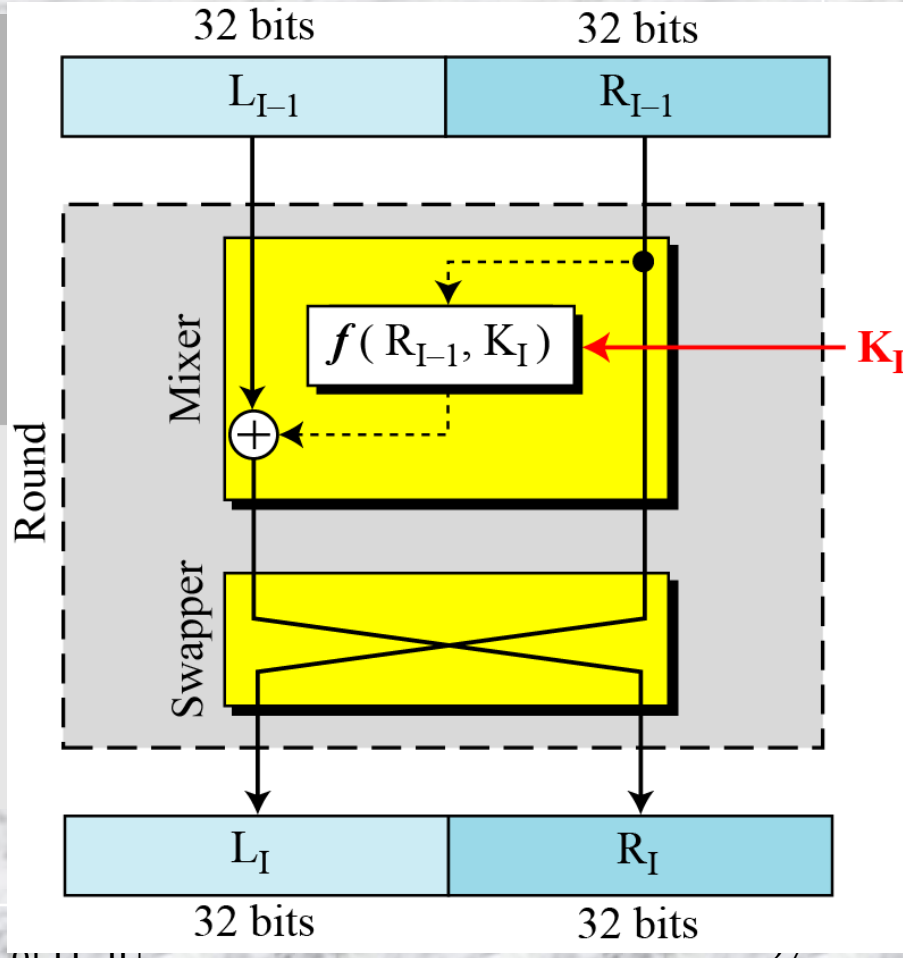


Pseudo code for DES

```
Cipher (plainBlock[64], RoundKeys[16, 48], cipherBlock[64])
{
    permute (64, 64, plainBlock, inBlock, InitialPermutationTable)
    split (64, 32, inBlock, leftBlock, rightBlock)
    for (round = 1 to 16)
    {
        mixer (leftBlock, rightBlock, RoundKeys[round])
        if (round!=16) swapper (leftBlock, rightBlock)
    }
    combine (32, 64, leftBlock, rightBlock, outBlock)
    permute (64, 64, outBlock, cipherBlock, FinalPermutationTable)
}
```

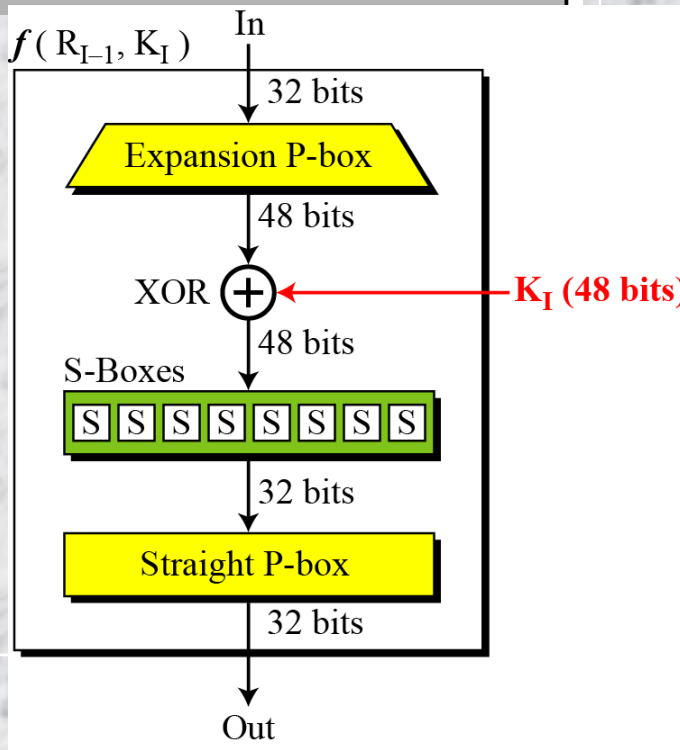
Pseudo code for DES (contd…)

```
mixer (leftBlock[48], rightBlock[48], RoundKey[48])
{
    copy (32, rightBlock, T1)
    function (T1, RoundKey, T2)
    exclusiveOr (32, leftBlock, T2, T3)
    copy (32, T3, leftBlock )
}
swapper (leftBlock[32], rigthBlock[32])
{
    copy (32, leftBlock, T)
    copy (32, rightBlock, leftBlock)
    copy (32, T, rightBlock)
}
```



Pseudo code for DES (contd…)

```
function (inBlock[32], RoundKey[48], outBlock[32])
{
    permute (32, 48, inBlock, T1, ExpansionPermutationTable)
    exclusiveOr (48, T1, RoundKey, T2)
    substitute (T2, T3, SubstituteTables)
    permute (32, 32, T3, outBlock, StraightPermutationTable)
}
```



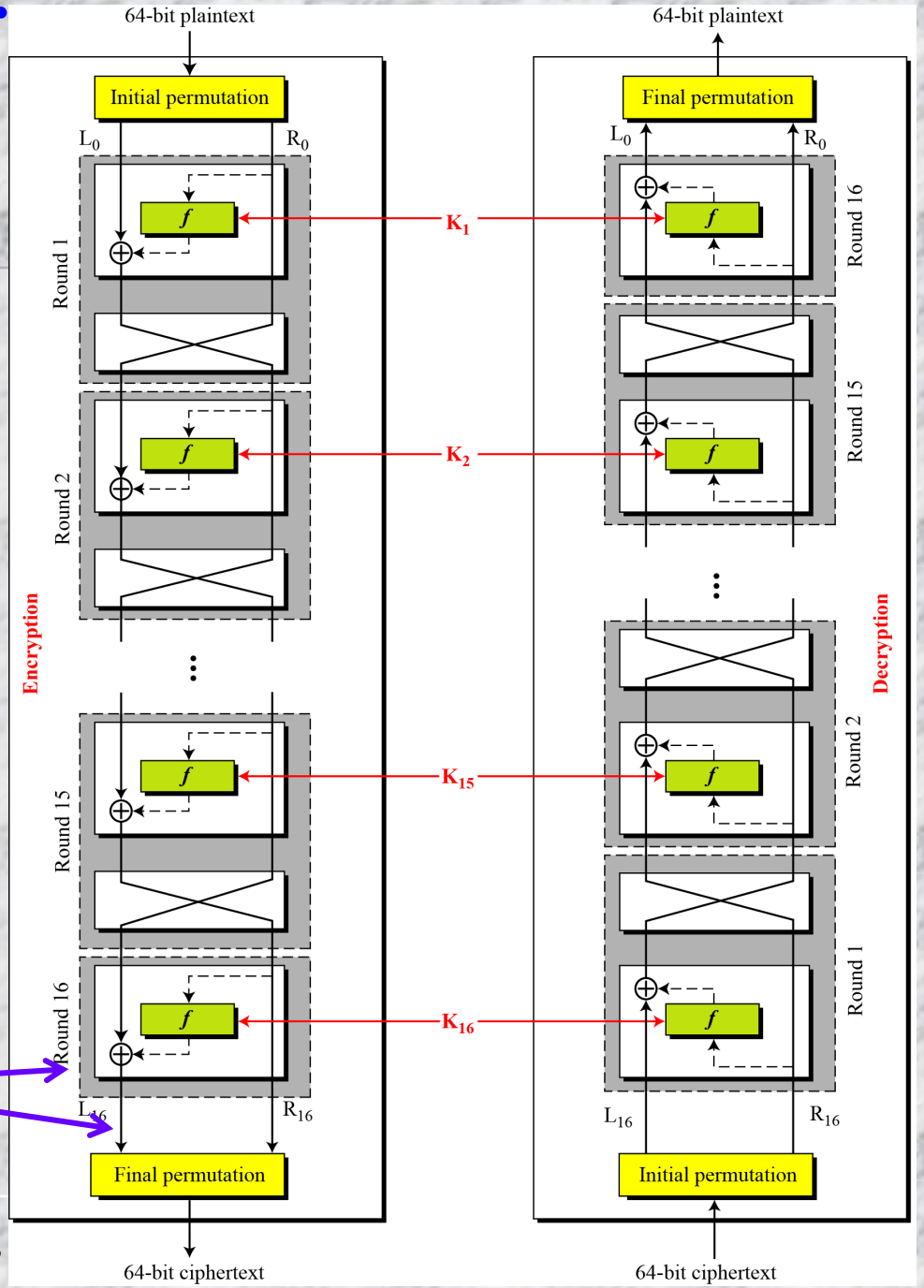
Pseudo code for DES (contd…)

```
substitute (inBlock[48], outBlock[32], SubstitutionTables[8, 4, 16]
{
    for (i = 1 to 8)
     {
        row ← 2 × inBlock[i × 6 + 1] + inBlock [i × 6 + 6]
        col  ← 8 × inBlock[i × 6 + 2] + 4 × inBlock[i × 6 + 3] +
                2 × inBlock[i × 6 + 4] + inBlock[i × 6 + 5]

        value = SubstitutionTables [i][row][col]


        outBlock[[i × 4 + 1] ← value / 8;        value ← value mod 8
        outBlock[[i × 4 + 2] ← value / 4;        value ← value mod 4
        outBlock[[i × 4 + 3] ← value / 2;        value ← value mod 2
        outBlock[[i × 4 + 4] ← value
     }
}
```

DES implementation: Second approach

- We can make all 16 rounds the same by including one swapper to the 16th round and
- Add an extra swapper after the 16^{th} round and before the combination
  - this will cancel out the effect of the 16^{th} round swapper

Add swapper



Key Generation

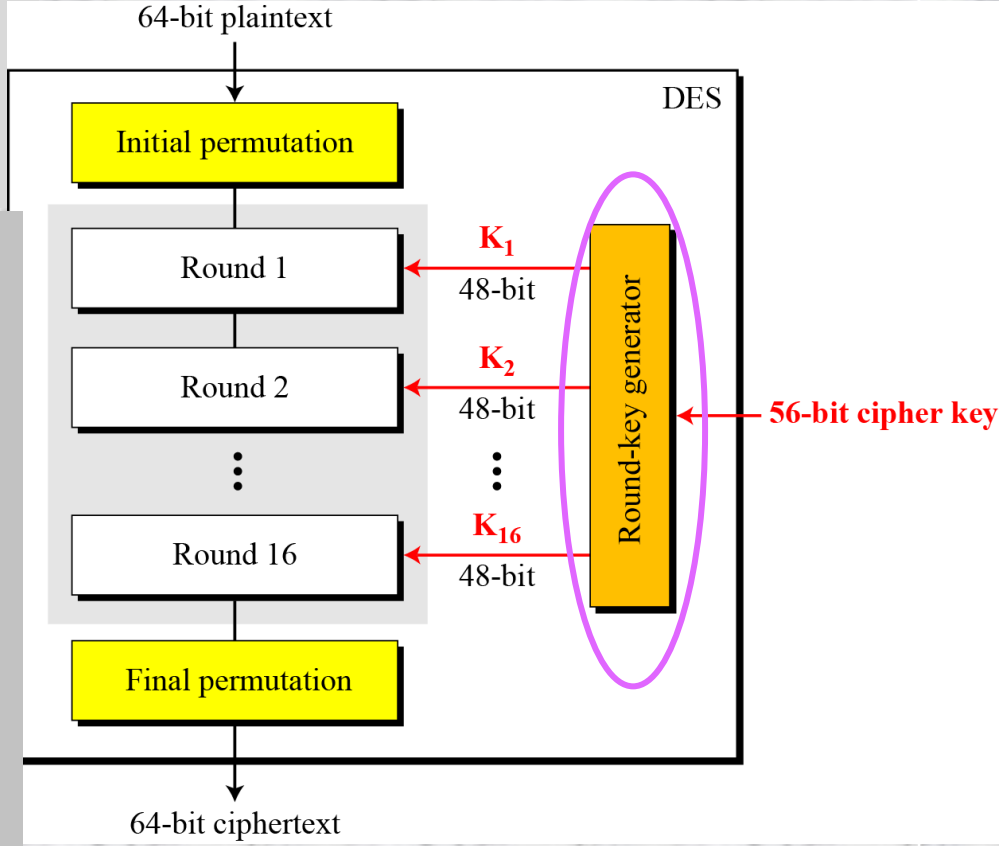
The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key

Normally Cipher key is 64-bit length

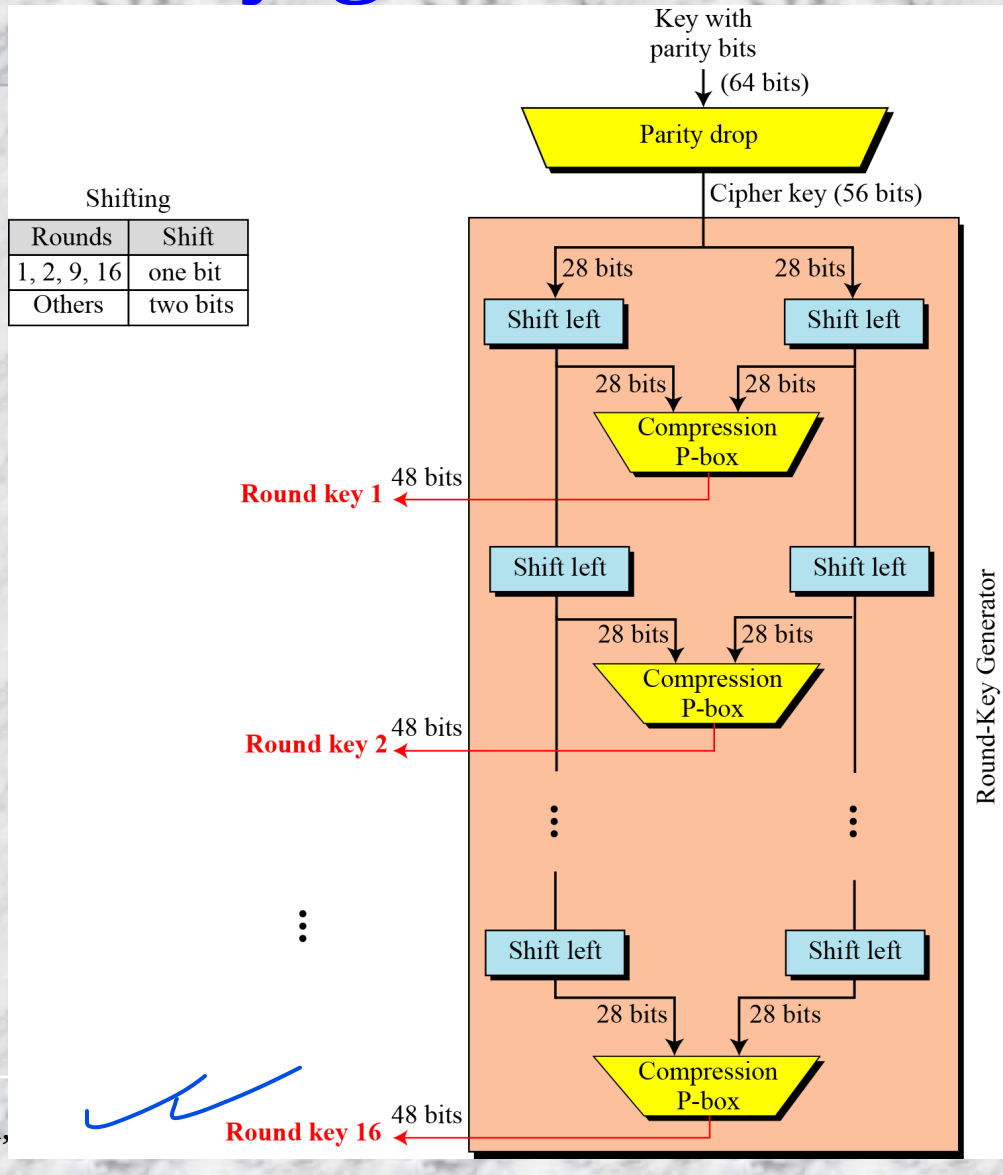
8 parity bits [8, 16, 24, 32, …, 64]

Parity bits are dropped and make it 56 bits

Remaining 56 bits are permuted according to the next table



Block diagram of key generation



Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

Key Generation (contd…)

Parity-bit drop table

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

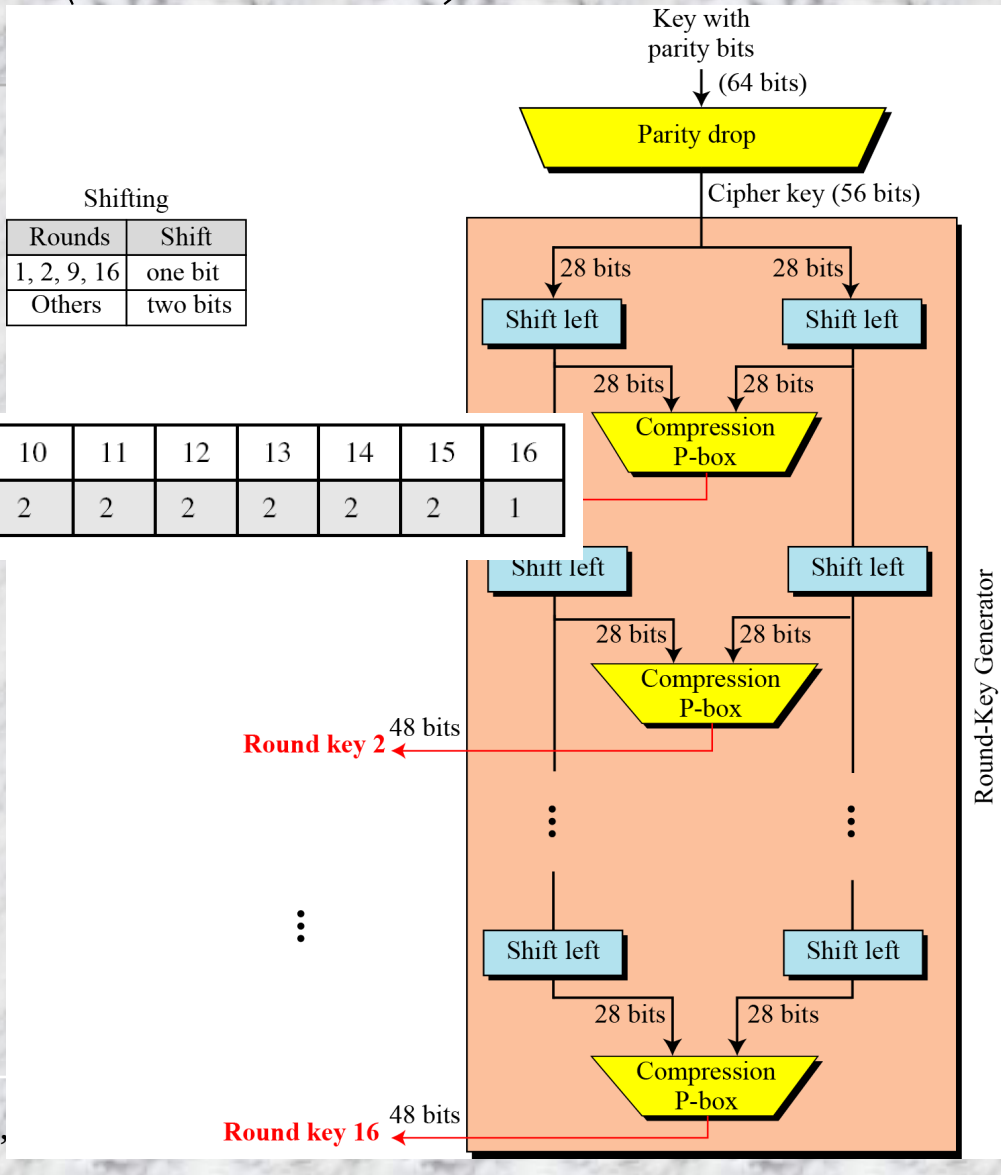
Key Generation (contd…)

Number of bits shifts

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

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



Key Generation (contd…)

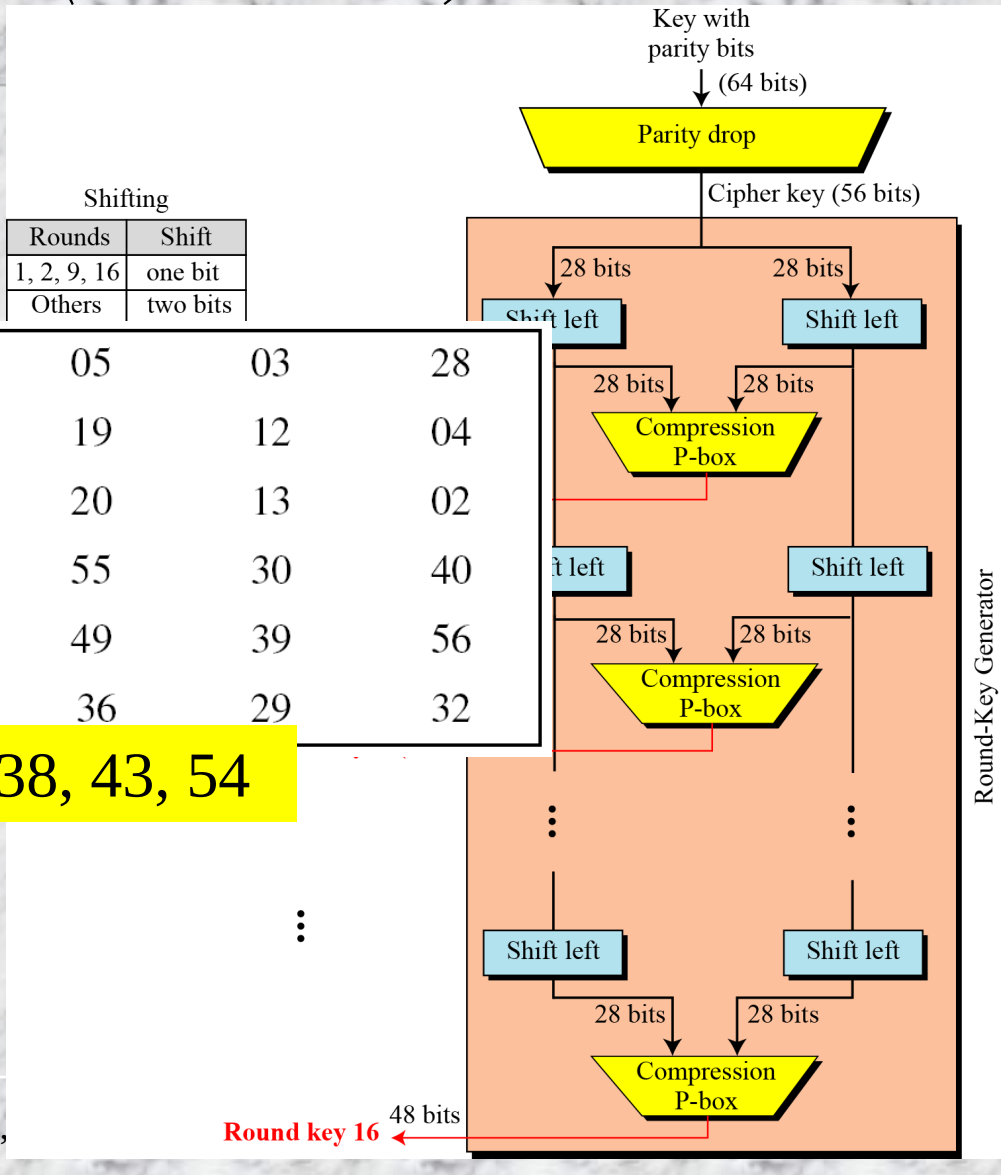
Key-compression table

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Dropped bits: 7, 9, 18, 22, 25, 38, 43, 54



Algorithm for round-key generation

```
Key_Generator (keyWithParities[64], RoundKeys[16, 48], ShiftTable[16])
{
    permute (64, 56, keyWithParities, cipherKey, ParityDropTable)
    split (56, 28, cipherKey, leftKey, rightKey)
     for (round = 1 to 16)
     {
          shiftLeft (leftKey, ShiftTable[round])
          shiftLeft (rightKey, ShiftTable[round])
          combine (28, 56, leftKey, rightKey, preRoundKey)
          permute (56, 48, preRoundKey, RoundKeys[round], KeyCompressionTable)
     }
}
```

```
shiftLeft (block[28], numOfShifts)
{
    for (i = 1 to numOfShifts)
    {
        T ← block[1]
        for (j = 2 to 28)
         {
            block [j−1] ← block [j]
        }
        block[28] ← T
    }
}
```

Example DES encipherment

- We choose a random plaintext block and a random key, and determine what the ciphertext block would be (all in hexadecimal):

Plaintext: 123456ABCD132536 Key: AABB09182736CCDD
CipherText: C0B7A8D05F3A829C

Plaintext: 123456ABCD132536			
After initial permutation: 14A7D67818CA18AD After splitting: L_0=14A7D678 R_0=18CA18AD			
Round	*Left*	*Right*	*Round Key*
Round 1	18CA18AD	5A78E394	194CD072DE8C
Round 2	5A78E394	4A1210F6	4568581ABCCE
Round 3	4A1210F6	B8089591	06EDA4ACF5B5
Round 4	B8089591	236779C2	DA2D032B6EE3

Example DES encipherment (contd…)

Round 5	236779C2	A15A4B87	69A629FEC913
Round 6	A15A4B87	2E8F9C65	C1948E87475E
Round 7	2E8F9C65	A9FC20A3	708AD2DDB3C0
Round 8	A9FC20A3	308BEE97	34F822F0C66D
Round 9	308BEE97	10AF9D37	84BB4473DCCC
Round 10	10AF9D37	6CA6CB20	02765708B5BF
Round 11	6CA6CB20	FF3C485F	6D5560AF7CA5
Round 12	FF3C485F	22A5963B	C2C1E96A4BF3
Round 13	22A5963B	387CCDAA	99C31397C91F
Round 14	387CCDAA	BD2DD2AB	251B8BC717D0
Round 15	BD2DD2AB	CF26B472	3330C5D9A36D
Round 16	19BA9212	CF26B472	181C5D75C66D
After combination: 19BA9212CF26B472			
Ciphertext: C0B7A8D05F3A829C			*(after final permutation)*

Example DES decipherment (contd…)

Ciphertext: C0B7A8D05F3A829C			
After initial permutation: 19BA9212CF26B472 After splitting: L_0=19BA9212 R_0=CF26B472			
Round	*Left*	*Right*	*Round Key*
Round 1	CF26B472	BD2DD2AB	181C5D75C66D
Round 2	BD2DD2AB	387CCDAA	3330C5D9A36D
. . .	. . .	. . .	. . .
Round 15	5A78E394	18CA18AD	4568581ABCCE
Round 16	14A7D678	18CA18AD	194CD072DE8C
After combination: 14A7D67818CA18AD			
Plaintext:123456ABCD132536 (after final permutation)			

DES Analysis

- *Critics have used a strong magnifier to analyze DES*
- *Tests have been done to measure the strength of some desired properties in a block cipher*
  - Properties
  - Design Criteria
  - DES Weaknesses

Properties

- *Two desired properties of a block cipher are*
  - *the avalanche effect and*
  - *the completeness*

Avalanche effect

- A small change in plaintext or key should create a significant change in ciphertext
- In following example, we check the avalanche effect in DES
- Two plaintext blocks (with the same key) that differ only in one bit and observe the result

Plaintext: 0000000000000000 Key: 22234512987ABB23
Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000001 Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3

Rounds	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Bit differences	1	6	20	29	30	33	32	29	32	39	33	28	30	31	30	29

Completeness effect

- *Completeness effect means that each bit of the ciphertext needs to depend on many bits on the plaintext*
- *S-Boxes*
  - *The design provides confusion and diffusion of bits from each round to the next*
- *P-Boxes*
  - *They provide diffusion of bits*
- *Number of Rounds*
  - *DES uses sixteen rounds of Feistel ciphers. the ciphertext is thoroughly a random function of plaintext and ciphertext.*

DES Weaknesses

- *Weaknesses in S-boxes*
  - *not covered as all the S-boxes are not included in this slide*
- *Weaknesses in P-boxes*
  - *Purpose of initial and final permutation is not clear, they also not provide any security benefit*
  - *In expansion permutation, the first and fourth bits of every 4-bit series are repeated*
- *Weaknesses in Key*

Weaknesses in Key

- Several weakness in the cipher key
  - The size of the 56 bits
    - With checking one million keys per second in a computer with single processor takes 2000 years for brute-force attack
    - A computer with one million chips (parallel processing) takes 20 hours to test the whole key space
    - …
  - DES cipher with 56 bits key is not safe enough

Weak keys

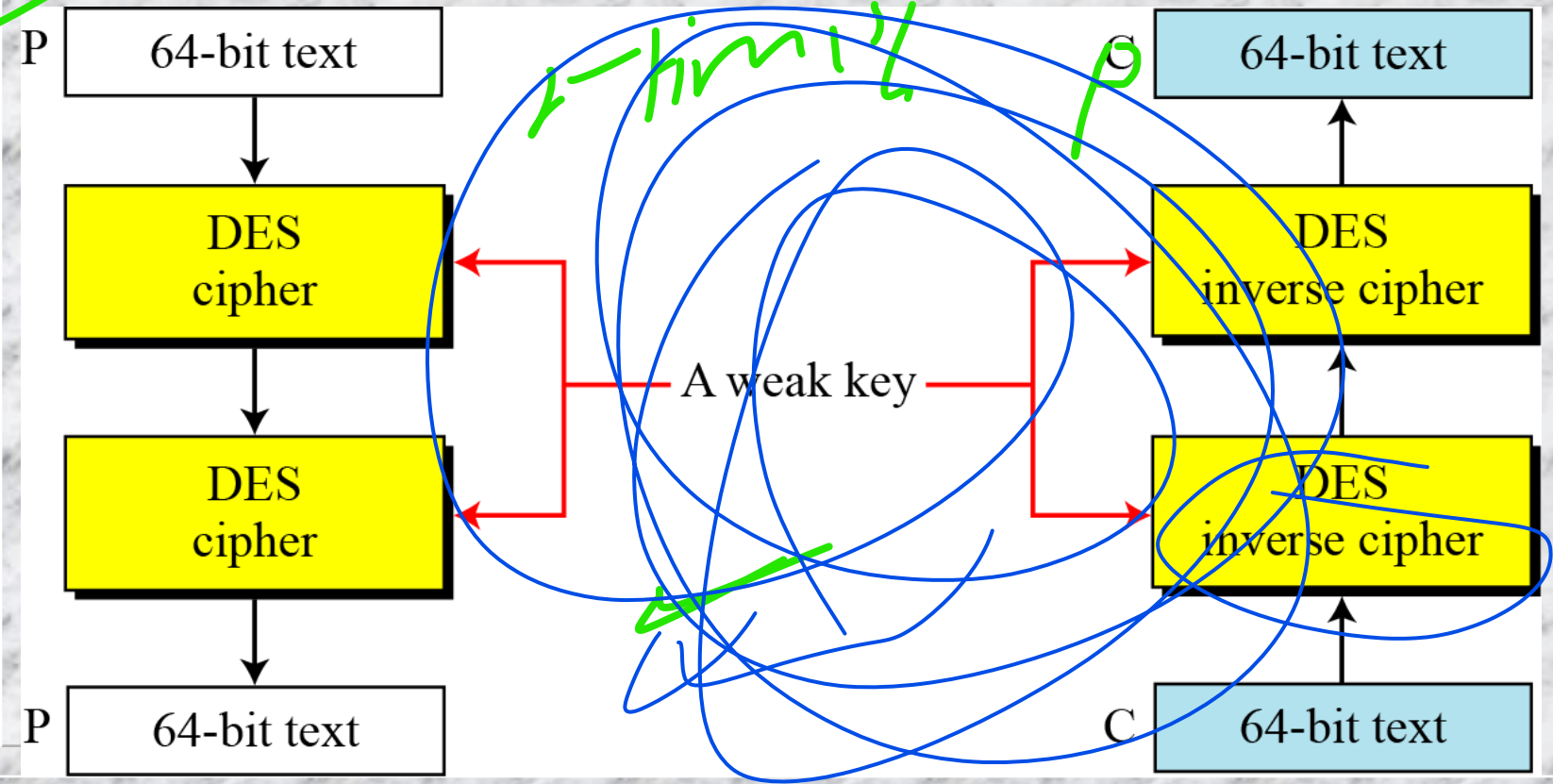
- Four keys out of 2^{56} keys are called weak keys
- The round keys generated from the weak keys are same and have same pattern as the cipher key
  - E.g., all sixteen round keys generated from first weak key is all mode of 0s
    - As key-generation algorithm first divides the cipher key into two halves and shifting and permutation does not change the block at all

Table 6.18 *Weak keys*

Keys before parities drop (64 bits)	*Actual key (56 bits)*
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

Weak keys (contd…)

- Weak key is the inverse itself



Weak keys (contd…)

- Let us try the first weak key to encrypt a block two times
  - After two encryptions with the same key the original plaintext block is created
  - Note that we have used the encryption algorithm two times, not one encryption followed by another decryption.

Key: 0x0101010101010101
Plaintext: *0x1234567887654321* Ciphertext: 0x814FE938589154F7

Key: 0x0101010101010101
Plaintext: 0x814FE938589154F7 Ciphertext: *0x1234567887654321*

Semi-weak keys

- There are six key pairs that are called semi-weak keys
  - A semi-weak key creates two different round keys and each of them repeats eight times
  - Round keys generated from each pair are the same with different orders

Table 6.19 *Semi-weak keys*

First key in the pair	*Second key in the pair*
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E1 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

Round keys generated from semi-weak keys pair

Round key 1	9153E54319BD	6EAC1ABCE642
Round key 2	6EAC1ABCE642	9153E54319BD
Round key 3	6EAC1ABCE642	9153E54319BD
Round key 4	6EAC1ABCE642	9153E54319BD
Round key 5	6EAC1ABCE642	9153E54319BD

Round key 1 in first set is the same as round key 16 in the second

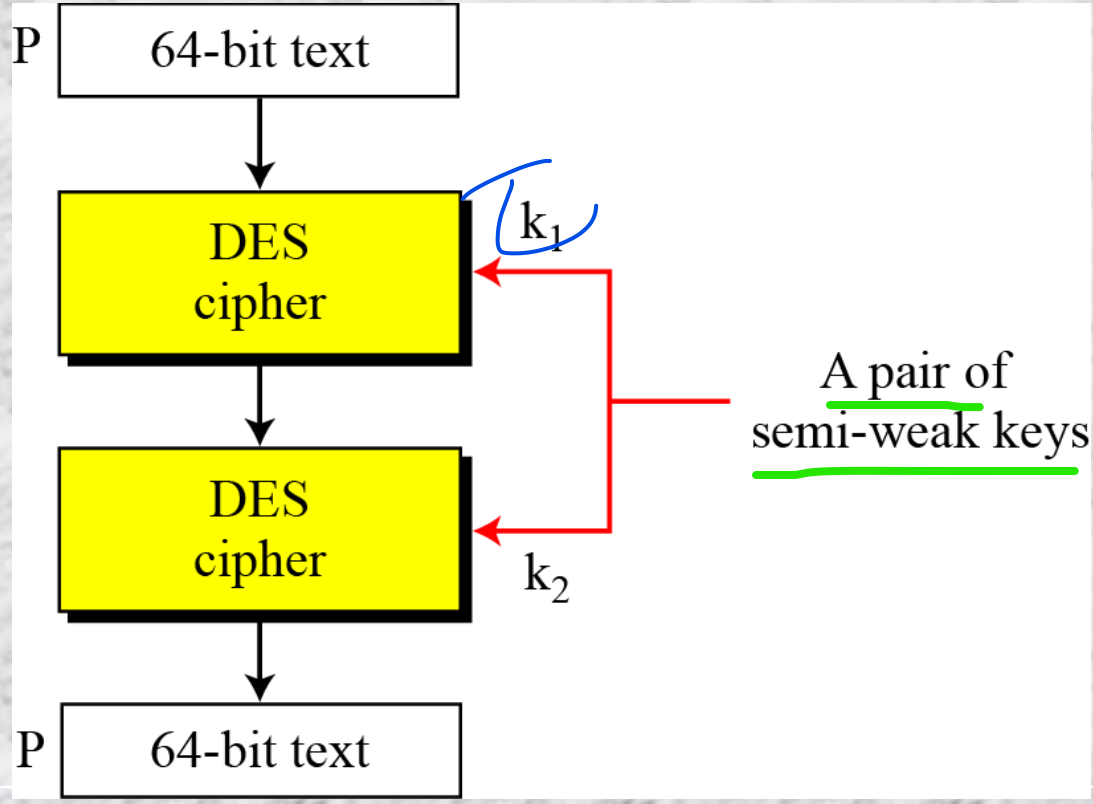
Round key 2 in first set is the same as round key 15 in the second

and so on …

Round key 13	9153E54319BD	6EAC1ABCE642
Round key 14	9153E54319BD	6EAC1ABCE642
Round key 15	9153E54319BD	6EAC1ABCE642
Round key 16	6EAC1ABCE642	9153E54319BD

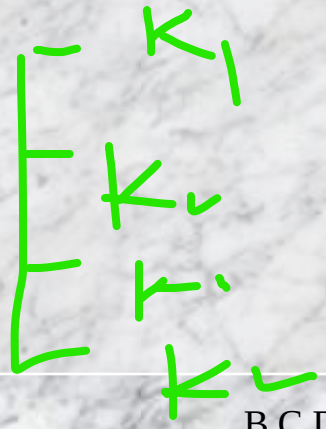
Semi-weak keys (contd…)

A pair of semi-weak keys in encryption and decryption



Possible weak keys

- There are also 48 keys that are called possible weak keys
  - A possible weak key generates only four distinct round keys



Key complement

- A key complement is made by inverting each bit
- It can be proved that C= E(K, P) → C' = E(K', P')
  - If the complement of the plaintext with the complement of the key is used to encrypt
    - We get the complement of the ciphertext
  - Attacker does not need to test all possible 2^{56} possible keys
    - She can test only half of them

Key complement

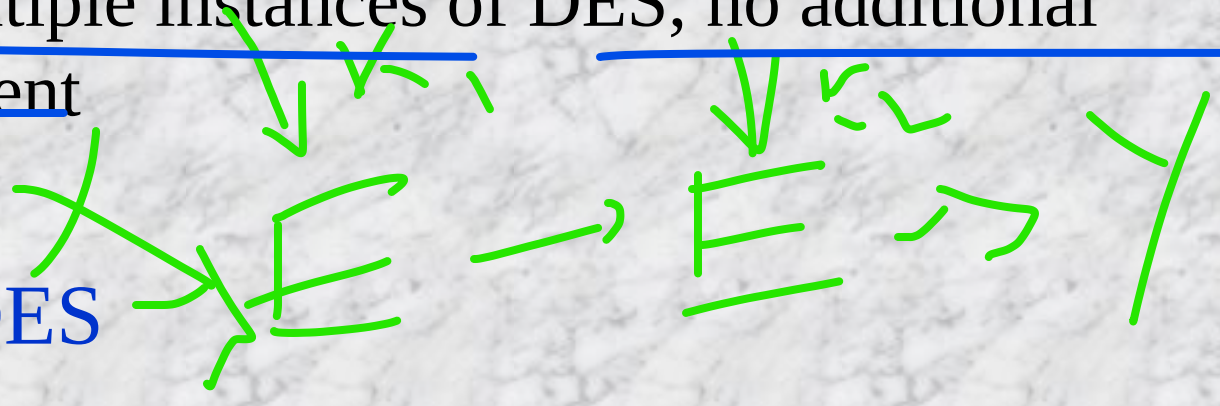
- Here one arbitrary key and plaintext are used to find the corresponding ciphertext
  - If the key complement and the complement of plaintext, we can obtain the complement of the ciphertext

Table 6.20 *Results for Example 6.10*

	Original	*Complement*
Key	1234123412341234	EDCBEDCBEDCBEDCB
Plaintext	12345678ABCDEF12	EDCBA987543210ED
Ciphertext	E112BE1DEFC7A367	1EED41E210385C98

Multiple DES

- The major drawback of DES is its key length
  - With available technology, parallel processing a brute-force attack is feasible
  - Abandon the DES and design a new cipher (AES)
  - Use multiple instances of DES, no additional investment



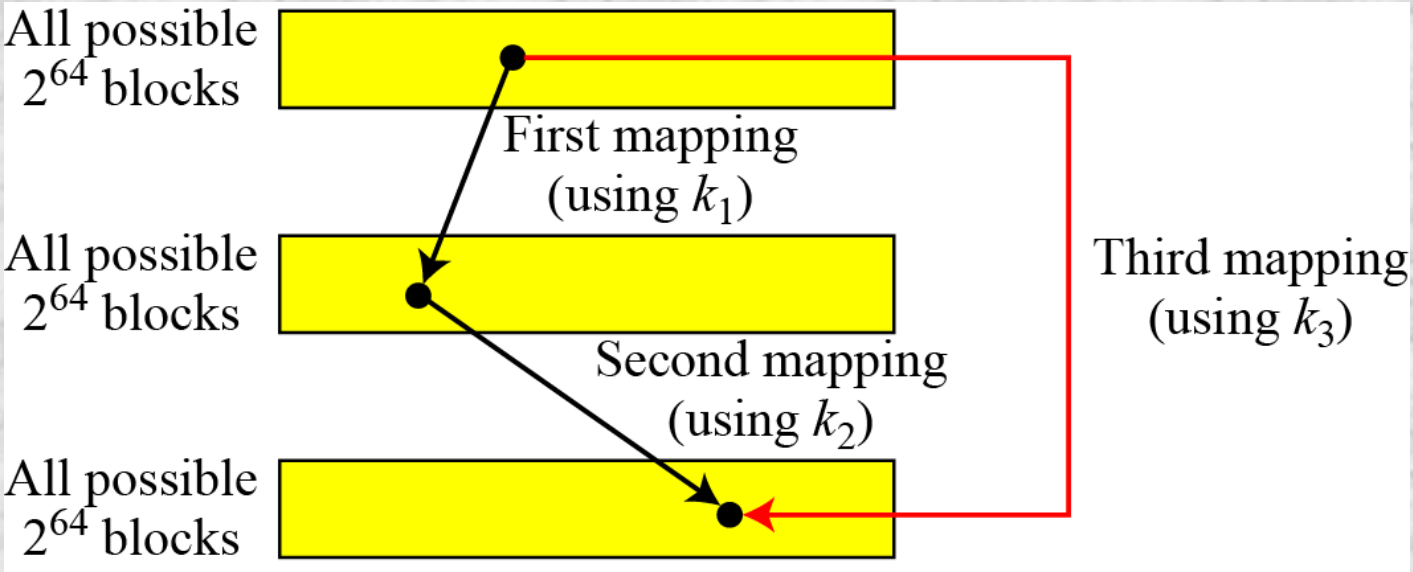
- Double DES
- Triple DES

Multiple DES (contd…)

- *A substitution that maps every possible input to every possible output is a group.*
- *If group: using two consecutive mappings is useless because we can always find third mapping that is equivalent to composition of the two*
- *Using double DES with two keys k_1 and k_2 is useless as a single DES with k_3 does same thing*

Composition mappings

using closure property of a group



Multiple DES (contd…)

- *Number of possible inpute or outputs in DES is* $N=2^{64}$
- *Possible mappings is* $N! = 10^{347,380,000,000,000,000}$
- *To support all of these mappings, key size should be* $\log_2(2^{64}!)= 2^{70}$ *bits*
- *DES key size is 56 bits*
- *Non of the subset defined by 56 bits can form a group, DES is not a group*
- *This means that we can use double or triple DES to increase the key size*

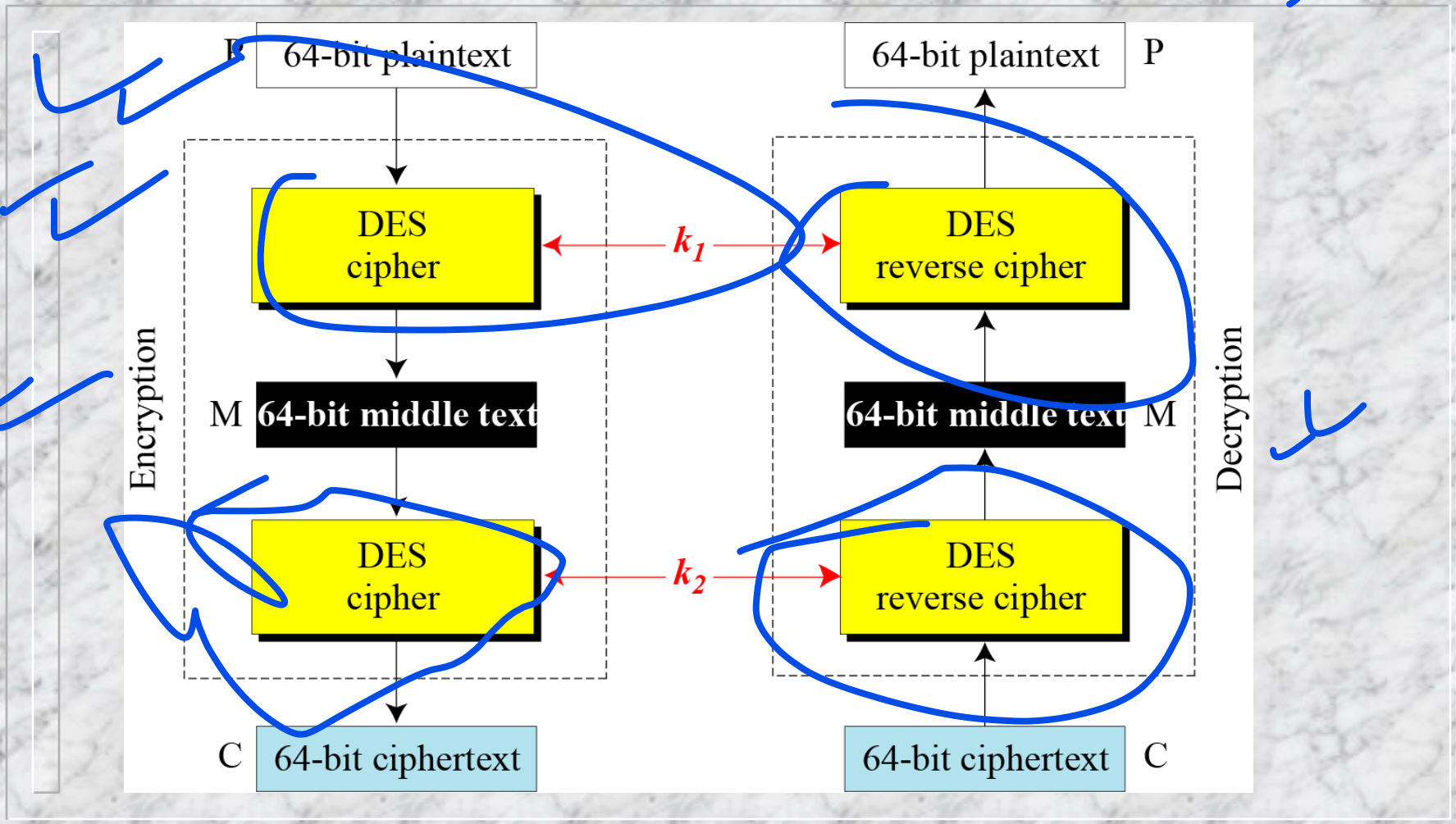
Double DES (2DES)

- Use two instances of DES ciphers for encryption with different keys
  - Key size is doubled (i.e., 112 bits)
  - 2DES is vulnerable to known-plaintext attack

Meet-in-the-Middle Attack

- It looks that 2DES increases the number of tests for key search from 2^{56} to 2^{112}
- A known-plaintext attack, called, meet-in-the-middle attack proves that 2DES improves this vulnerability slightly (to 2^{57} tests), but not tremendously (to 2^{112})

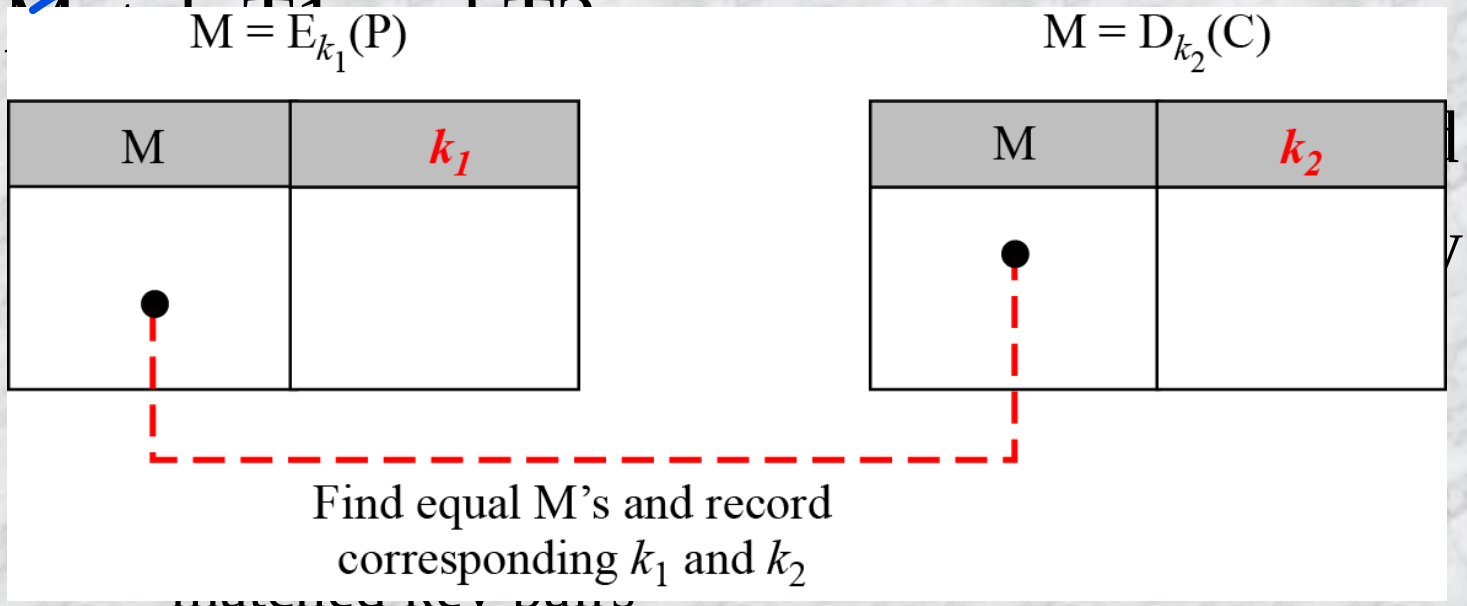
Meet-in-the-middle attack



Meet-in-the-middle attack (contd…)

- For middle text M:
  - $M = E(k_1, P) = D(k_2, C)$
- Attacker intercepted P and C
  - Attacker use all possible keys and compute corresponding M1=E(P, k) and store in a table T1 (2^{56})
  - Attacker use all possible keys and compute corresponding M2=D(C, k') and store in a table T2 (2^{56})

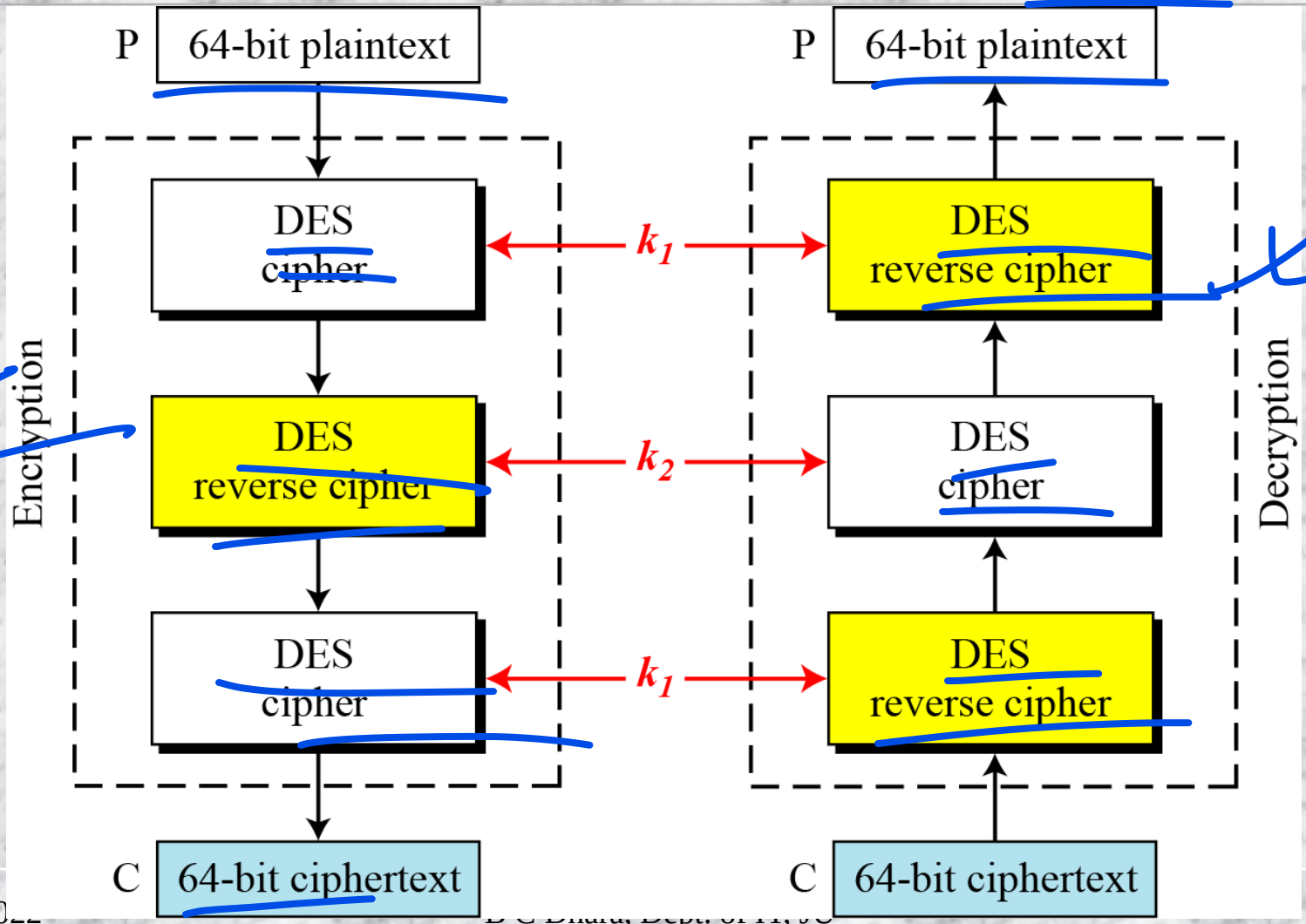
Meet-in-the-middle attack (contd…)

- 

- Instead of 2^{112} searches number of search will be limited by 2^{57}

Triple DES

- Two versions: With two keys, with three keys



Triple DES with Three Keys

- *The possibility of known-plaintext attacks on triple DES with two keys has enticed some applications to use triple DES with three keys*
- *Triple DES with three keys is used by many applications such as PGP (See Chapter 16).*

Security of DES

- *DES, as the first important block cipher, has gone through much scrutiny*
- *Among the attempted attacks, three are of interest:*
  - *brute-force,*
  - *differential cryptanalysis, and*
  - *linear cryptanalysis*

Brute-Force Attack

- *We have discussed the weakness of short cipher key in DES. Combining this weakness with the key complement weakness, it is clear that DES can be broken using 2^{55} encryptions.*
- *3DES:*
  - *with two keys: key size 2^{112}*
  - *with three keys: key size 2^{168}*
  - Make brute-force quite difficult

Differential Cryptanalysis

- *It has been revealed that the designers of DES already knew about this type of attack and designed S-boxes and chose 16 as the number of rounds to make DES specifically resistant to this type of attack*
- *Using 2^{47} chosen ciphertext or 2^{55} known plaintext, DES can be broken*
  - *Impractical to find such texts*

Linear Cryptanalysis

- *Linear cryptanalysis is newer than differential cryptanalysis*
- *DES is more vulnerable to linear cryptanalysis than to differential cryptanalysis*
- *S-boxes are not very resistant to linear cryptanalysis*
- *It has been shown that DES can be broken using 2^{43} pairs of known plaintexts*
  - *However, from the practical point of view, finding so many pairs is very unlikely*