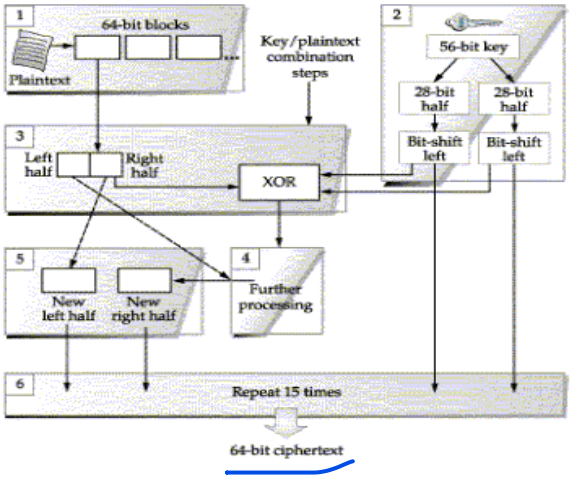
Data Encryption Standard - DES



DES was developed as a standard for communications and data protection by an IBM research team, in response to a public request for proposals by the NBS - the National Bureau of Standards (which is now known as NIST).

DES - History

- The Data Encryption Standard (DES) was developed in the 1970s by the National Bureau of Standards with the help of the National Security Agency.
- Its purpose is to provide a standard method for protecting sensitive commercial and unclassified data.
- IBM created the first draft of the algorithm, calling it LUCIFER with a 128-bit key.
- DES officially became a federal standard in November of 1976.

DES - History

- In May 1973, and again in Aug 1974 the NBS (now NIST) called for possible encryption algorithms for use in unclassified government applications.
- Response was mostly disappointing, however, IBM submitted their LUCIFER design.
- Following a period of redesign and comment it became the Data Encryption Standard (DES).

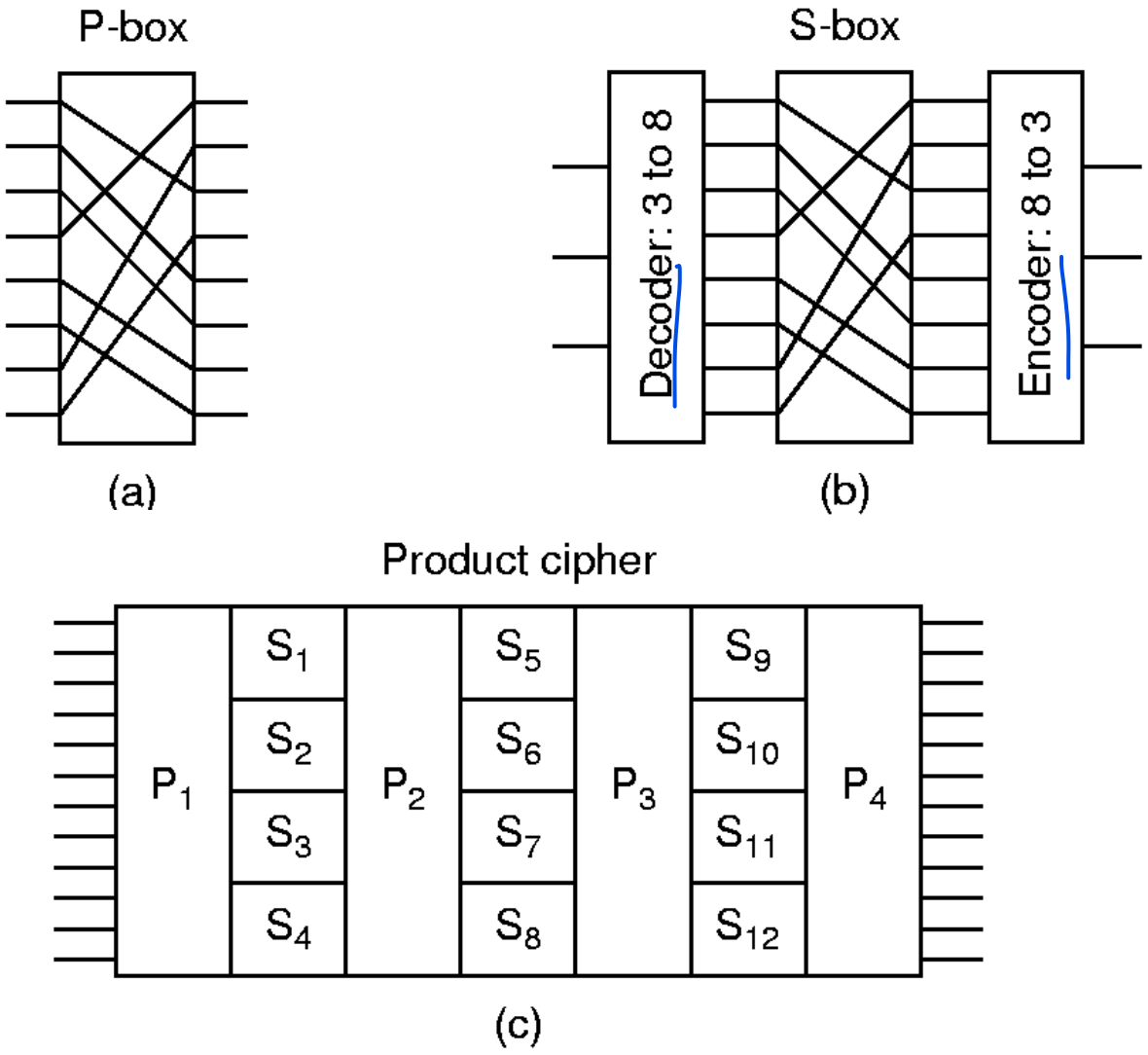
DES - As a Federal Standard

- DES was adopted as a (US) federal standard in November 1976, published by NBS as a hardware only scheme in January 1977 and by ANSI for both hardware and software standards in ANSI X3.92-1981 (also X3.106-1983 modes of use) .
- Subsequently DES has been widely adopted and is now published in many standards around the world.

DES - Basics

- DES uses the two basic techniques of cryptography - confusion and diffusion.
- At the simplest level, diffusion is achieved through numerous permutations and confusions is achieved through the XOR operation.
- This is also called an S-P network.

The S-P Network



DES in a Nutshell

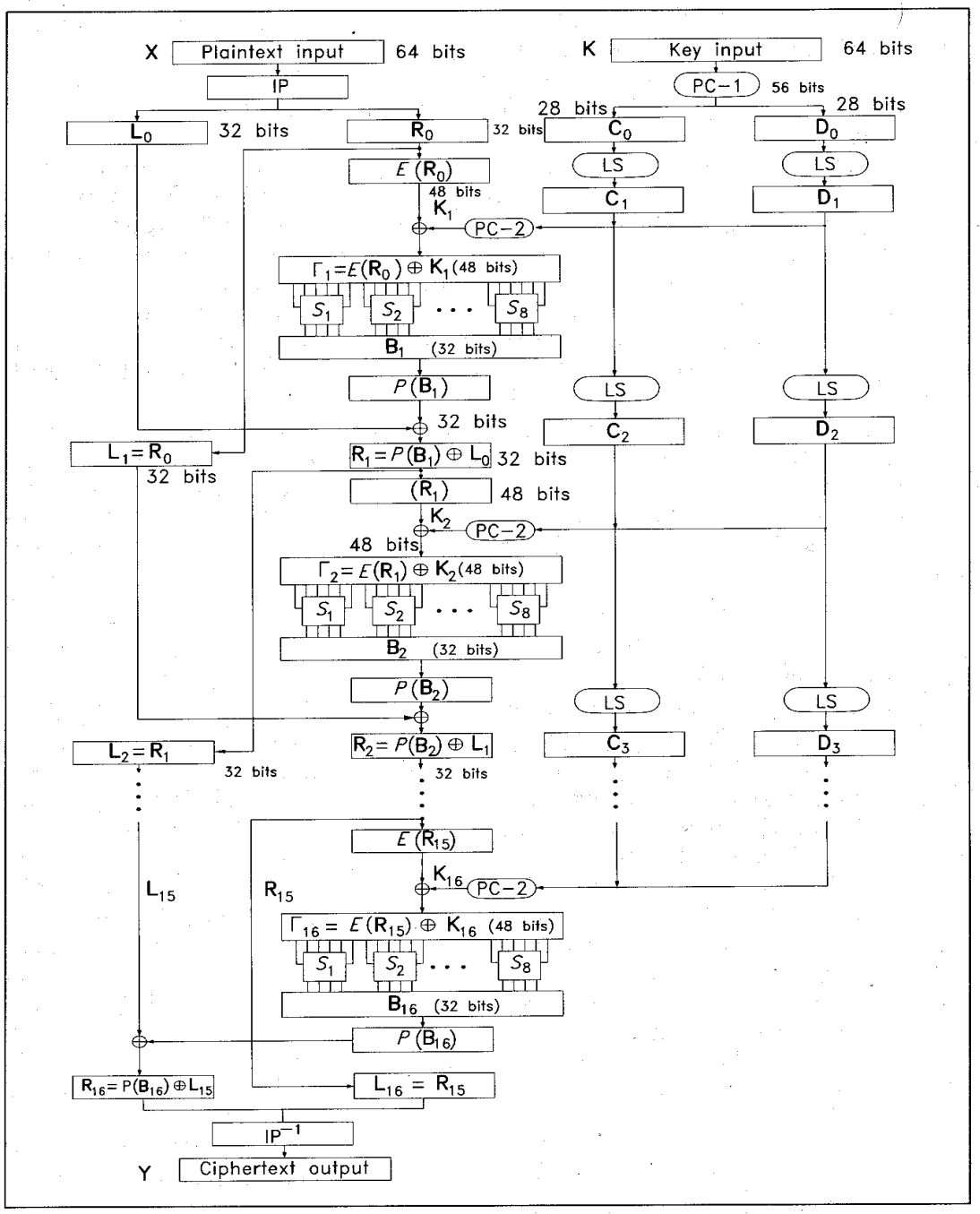
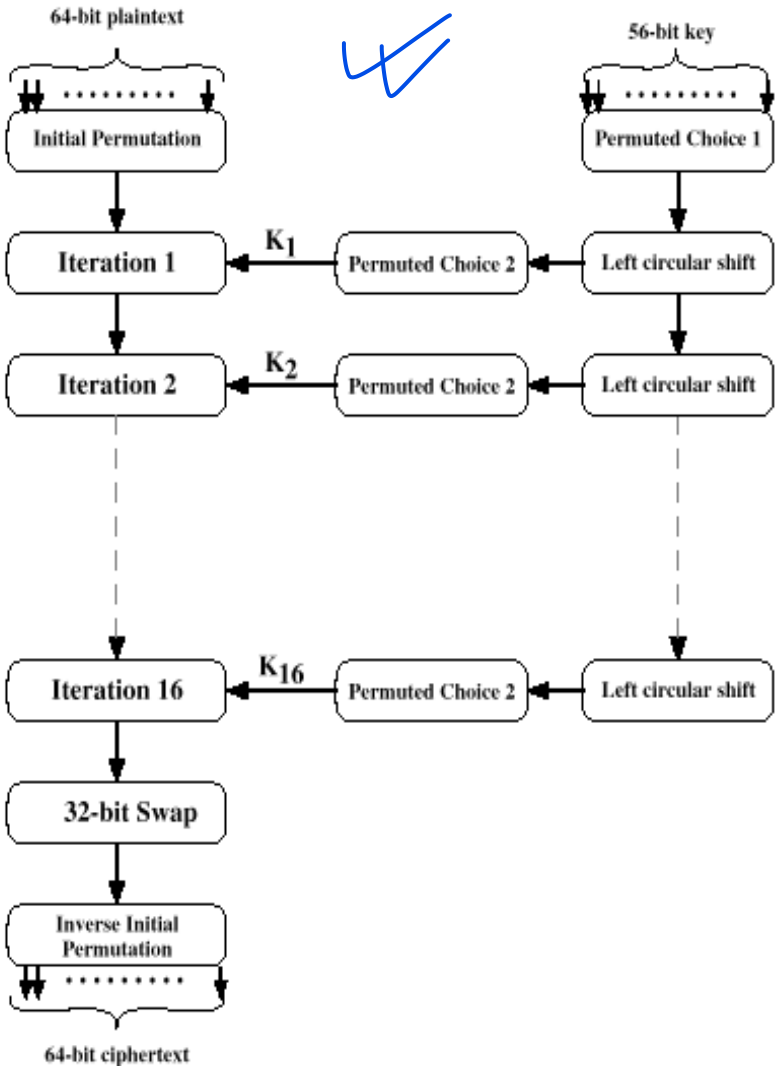


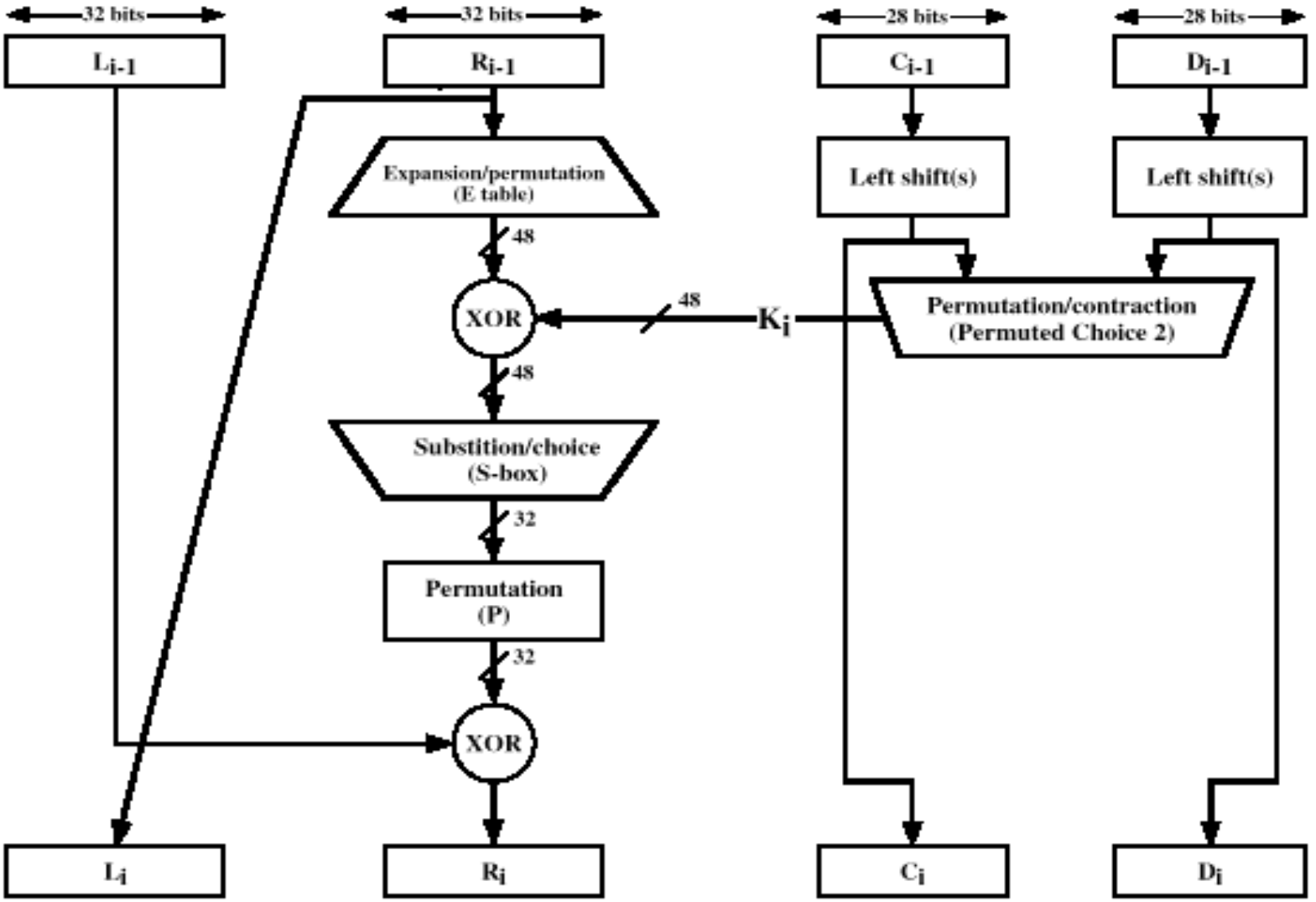
Fig. 3.1 Block cipher design used by DES

DES - The 16 Rounds

- The basic process in enciphering a 64-bit data block and a 56-bit key using the DES consists of:
  - An initial permutation (IP)
  - 16 rounds of a complex key dependent calculation f
  - A final permutation, being the inverse of IP



The 16 Rounds of F Consist Of:



DES - Swapping of Left and Right Halves

- The 64-bit block being enciphered is broken into two halves.
- The right half goes through one DES round, and the result becomes the new left half.
- The old left half becomes the new right half, and will go through one round in the next round.
- This goes on for 16 rounds, but after the last round the left and right halves are not swapped, so that the result of the 16th round becomes the final right half, and the result of the 15th round (which became the left half of the 16th round) is the final left half.

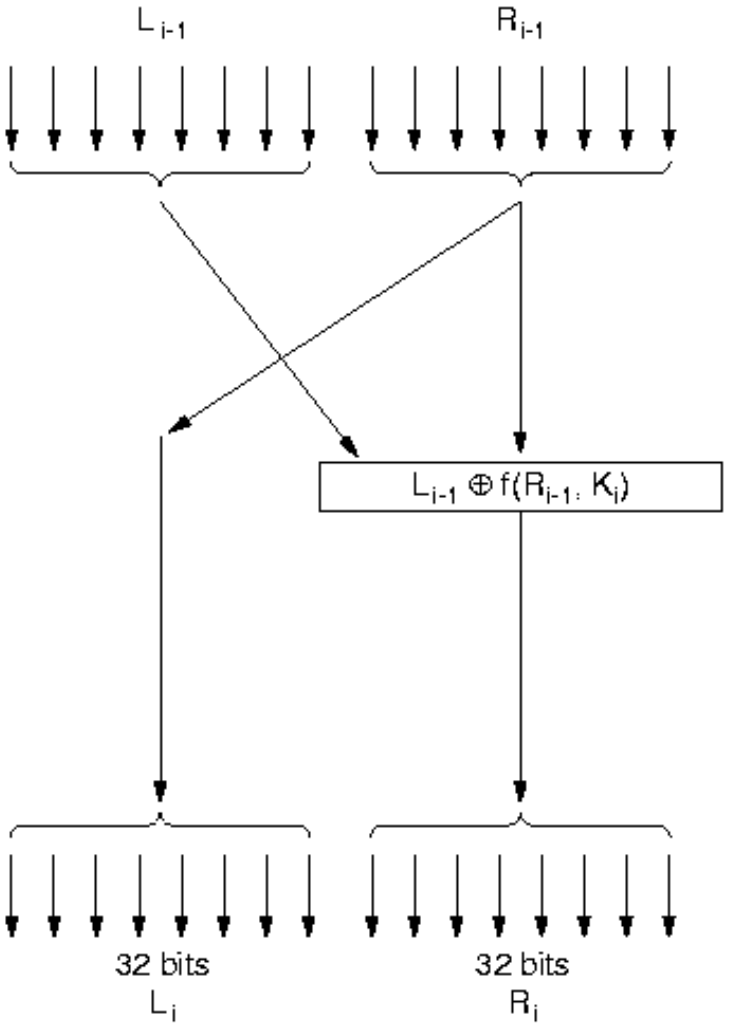
DES - Swapping of Left and Right Halves

- This can be described functionally as

$$L(i) = R(i-1)$$

$$R(i) = L(i-1) \oplus P(S(E(R(i-1)) \oplus K(i)))$$

⌘ This forms one round in an S-P network



DES - Basics

- Fundamentally DES performs only two operations on its input, bit shifting (permutation), and bit substitution.
- The key controls exactly how this process works.
- By doing these operations repeatedly and in a non-linear manner you end up with a result which can not be used to retrieve the original without the key.
- Those familiar with chaos theory should see a great deal of similarity to what DES does. By applying relatively simple operations repeatedly a system can achieve a state of near total randomness.

Each Iteration Uses a Different Sub-key

- DES works on 64 bits of data at a time. Each 64 bits of data is iterated on from 1 to 16 times (16 is the DES standard).
- For each iteration a 48 bit subset of the 56 bit key is fed into the encryption block
- Decryption is the inverse of the encryption process.

DES Key Processing

- The key is usually stored as a 64-bit number, where every eighth bit is a parity bit.
- The parity bits are pitched during the algorithm, and the 56-bit key is used to create 16 different 48-bit subkeys - one for each round.
- DES Subkeys: K_1, K_2, K_3, ... K_{16}

DES Key Processing - Subkeys Generation

- In order to generate 16 48-bit subkeys from the 56-bit key, the following process is used.
- First, the key is loaded according to the PC-1 and then halved.
- Then each half is rotated by 2 bits in every round except the first, second, 9th and last rounds.
- The reason for this is that it makes it secure against related-key cryptanalysis.
- Then 48 of the 56 bits are chosen according to a compression permutation - PC-2.

The Key Schedule

- The **subkeys** used by the 16 rounds are formed by the **key schedule** which consists of:
  - An initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
  - 16 stages consisting of
    - selecting 24-bits from each half and permuting them by PC2 for use in function f,
    - rotating each half either 1 or 2 places depending on the **key rotation schedule** KS
    - this can be described functionally as:

```
K(i) = PC2(KS(PC1(K),i))
```

Permuted Choice 1 -- PC-1

Table 3.1 Permuted Choice 1 (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

Permuted Choice 2 -- PC-2

Table 3.3 Permuted Choice 2 (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

12

Key Rotation Schedule

- The **key rotation schedule** KS is specified as:

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
KS	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Total Rot	1	2	4	6	8	10	12	14	15	17	19	21	23	25	27	28

Table 3.2 Shifted Schedule for Encipherment

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number of Left Shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Operation - Plaintext

- The block to be encrypted is halved - the right half goes through several steps before being XOR-ed with the left half and, except after the last round, trading places with the left half.

DES - Expansion Permutation

- First the right half goes through an expansion permutation which expands it from 32 to 48 bits.
- This makes it the same length as the subkey to allow the XOR, but it also demonstrates an important concept in cryptography. In expanding to 1.5 times its size, several bits are repeated (no new bits are introduced - all the existing bits are shifted around, and some are used twice).
- Because of this some of the input bits affect two output bits instead of one, the goal being to have every output bit in DES depend upon every input bit as quickly as possible. This is known as the avalanche effect.

Expansion Permutation Table

Table 3.5 E Bit-Selection Table

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

DES Operation - $E(R_i) \oplus K_i$

- The result of the expansion permutation is XOR-ed with the subkey, and then goes through the S-boxes.
- There are 8 S-boxes, each of which takes a 6-bit input an spits out a 4-bit output.
- This step is non-linear. For a given input i1, i2 ... i6, the output is determined by using the concatenation of i1 and i6, and the concatenation of i2… i5, and using these as the indices to the table which is the S-box.

S-box Permutations

- The S-boxes are somewhat different from the other permutations. While all the others are set up according to "bit x goes to bit y", the input bits can be viewed differently for the S-boxes.
- If the input is {d1,d2,d3,d4,d5,d6} then the two-bit number {d1,d6} and the the four-bit number {d2,d3,d4,d5} are used as indices to the table.
- For the 48-bit word {d1,d2..d48}, the word {d1… d6} is sent to S-box 1, the word {d7… d12} to S-box 2, etc. The output of S-box 1, {o1… o4}, that of S-box 2, {o5..o8} etc. are concatenated to form the output.

S-box Permutations

Table 3.6 Primitive S-Box Functions

						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_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_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

S1 Box Truth Table

Table 3.9 Truth Table for S_1-box Function

x_6	x_5	x_4	x_3	x_2	x_1	y_4	y_3	y_2	y_1
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	0	1	0	0
0	0	0	0	1	0	1	1	0	1
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	0	0	0	1	0
0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	1	0	1	1
0	0	0	1	1	1	1	0	0	0
0	0	1	0	0	0	0	0	1	1
0	0	1	0	0	1	1	0	1	0

The 8 DES S Boxes

Table 3.6 Primitive S-Box Functions

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_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_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_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_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_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_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_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

DES Operation - P Box

- The output of each of the 8 S-boxes is concatenated to form a 32-bit number, which is then permutated with a P-box.
- This P-box is a straight permutation, and the resulting number is XOR-ed with the left half of the input block with which we started at the beginning of this round.
- Finally, if this is not the last round, we swap the left and right halves and start again.

P Box

Table 3.7 Permutation Function P

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

DES Permutations

- The initial and final permutations in DES serve no cryptographic function. They were originally added in order to make it easier to load the 64-bit blocks into hardware - this algorithm after all predates 16-bit busses - and is now often omitted from implementations.
- However the permutations are a part of the standard, and therefore any implementation not using the permutations is not truly DES.

DES Permutations

- Using the Initial Permutation a DES chip loads a 64-bit block one bit at a time (this gets to be very slow in software).
- The order in which it loads the bits is shown below.
- The final permutation is the inverse of the initial (for example, in the final permutation bit 40 goes to bit 1, whereas in the initial permutation bit 1 goes to bit 40).

Initial Permutation

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

DES Initial and Final Permutations

Table 3.8 Inverse of Initial Permutation, IP^{-1}

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

Weak Keys

- There are a few keys which are considered weak for the DES algorithm. They are so few, however, that it is trivial to check for them during key generation.