Data Encryption Standard (DES)

DES Background

The DES algorithm based on **LUCIFER,** designed by Horst Feistel, was developed at IBM in 1972. This algorithm was approved by the National Bureau of Standards (now NIST) after assessment of DES strength and modifications by the National Security Agency (NSA), and became a Federal standard in 1977.

In 2000, NIST selected a new algorithm (**Rijndael**) to be the Advanced Encryption Standard (AES). This will eventually replace DES.

Following Kerckhoffs' principle, all aspects of DES and AES are public knowledge.

The Algorithm

Overview

Initial Perm / Expander

DES function

S-boxes

Key generation

Exit

Keylength Weakness

The key length (effectively 56 bits) is now considered to be too short.

In 1977, Diffie and Hellman claimed that an appropriate machine consisting of a million LSI chips could try all $2^{56} \sim 10^{17}$ keys in one day for the entire search. The cost would be about $20 million ('77 dollars) for such a machine.

In 1993, Michael Wiener gave a detailed design of a key search machine based on a chip that could test 5×10^7 keys per second, and could be built with current technology for $10.50 per chip. A frame consisting of 5760 chips can be built for $100,000 and would allow a DES key to be found in about 1.5 days on average. A machine using 10 frames would cost a million dollars but would reduce the search time to about 3.5 hours.

Keylength Weakness

Starting in 1996, in an attempt to prove the inadequacy of this key length, Ron Rivest (through his company RSALabs) conducted four contests, offering cash rewards ($10K) for any individual or group who could break a DES encrypted message. The first contest was won by a group (*Deschall - coordinated by Rake Verser*) using a distributed network approach, taking 96 days. The second by another group (*Distributed Net*) in 41 days. In 1998 the Electronic Frontier Foundation (EFF) using a specially built computer (**Deep Crack**) costing less than $250,000 won the third contest in 56 hours. The last contest was won in January 1999 by a combination of distributed network (*Distributed Net*) and the EFF machine in 22 hours 15 min.

S-box Construction

The complete specifications of the S-boxes have remained secret. This has lead some to believe that NSA has a backdoor into the DES algorithm.

To allay these suspicions in the early 1990's IBM published its design criteria for the S-boxes. These indicated that the S-boxes were designed to thwart certain sophisticated attacks (especially differential cryptanalysis), and that the actual construction was based on computer searches. However, this does not address the modifications made by NSA.

Other Developments

In 1986, NIT in Japan developed the Fast Data Encipherment Algorithm (FEAL-8). It was designed to be a high-speed software cipher and is used in FAX terminals, modems and telephone cards due to its compactness. Further development has given FEAL-N and FEAL-NX (which uses a 128 bit key). (The N refers to the number of rounds and is a power of 2).

In 1990, Brown, Piepzyk and Seberry at UNSW (Univ. of New South Wales - Australia) proposed a DES-like cipher LOKI which uses a full 64-bit key.

In the 1991, Biham and Shamir introduced a method called differential cryptanalysis and demonstrated that many symmetric cryptosystems can be broken by their method. This has been one of the most effective attacks on DES type systems.

Future of DES

DES was up for a 5 year review by NIST in 1992 and the decision was made to keep it as a standard (to the surprise of many), but at its next review in 1997 it was clear that it was going to be replaced. Some expected it not to remain a standard after this review, but due to NIST's activities concerning the new AES (Advanced Encryption Standard) the decision was made to keep DES as the standard (but only **triple DES** was to be considered secure). On Dec. 4, 2001, Secretary of Commerce Don Evans announced the approval of AES as the new standard, replacing DES. Products implementing the AES are now available in the marketplace.

Modes of Operation

The standard procedure of blocking the message into blocks of length 64 bits and enciphering each block (using the same key) is known as the *electronic codebook mode* (ECB).

It can also be used to produce a key stream cipher, this is known as the *output feedback mode* (OFB). In this mode of operation, an initialization string of 64 bits is encrypted with DES and then the output is again encrypted, and again, and again ... This produces a bit stream (the original string and each of its encryptions) which is then xor'ed (addition mod 2) with the message to produce the encrypted message as in the one-time pad.

In *cipher block chaining mode* (CBC) the enciphered output of a message block is xor'ed with the next message block before it is run through DES. In this mode of operation, any altered message block will affect all the ciphertext blocks that follow it. This is a useful property in certain applications, in particular, in the construction of *message authentication codes* (MAC's).

Message Authentication

In some applications, it is more important to know that a message has not been altered rather than keeping it secret. This can be achieved by tacking on to a message a special code, some type of encrypted ciphertext that depends on the message, called a *message authentication code* (**MAC**). If the plaintext message is altered in some way, the MAC would not be correct, thus alerting the message receiver of the tampering. DES can be used to produce MAC's in the following way. The sender uses DES in CBC mode to encipher the message. The very last block of the enciphered message is the MAC. The plaintext message and the MAC are then sent to the receiver. The receiver then runs the plaintext that was received through DES (in CBC mode with the same key) and compares the MAC that was just calculated with the MAC that was transmitted.

Of course, if the entire transaction is to be done in secret, the plaintext and MAC can be run through DES (in any mode, but with a different key) before transmission.

Advanced Encryption Standard
AES

Background

In 1997 the National Institute of Standards and Technology (NIST) put out a call for candidates to replace DES. The requirements included the ability to allow key sizes of 128, 192 and 256 bits; the algorithm should work on blocks of 128 bits; and it should be highly portable, working on a variety of hardware platforms including 8-bit processors used in smart cards and 32-bit processors used in most personal computers. Speed and cryptographic strength were also considerations. NIST selected 15 algorithms and asked the cryptographic community to comment on them in a series of forums and workshops. In 2000 the list had been reduced to five finalists: **MARS** (the IBM entry), **RC6** (from RSA Laboratories), **Rijndael** (from Joan Daemen and Vincent Rijmen), **Serpent** and **Twofish**. Eventually Rijndael was selected to be the AES and the official announcement that it was the new standard was made on Dec. 4, 2001 (to be effective March 26, 2002).

Rijndael

The algorithm is designed to use keys of length 128, 192 or 256. It works on one block of 128 bits at a time, producing 128 bits of ciphertext. There are 10 rounds, after an initial XOR'ing (bitwise addition mod 2) with the original key (assuming a key length of 128). These rounds, except for the last, consist of 4 steps (layers), called **ByteSub**, **ShiftRow**, **MixColumn** and **AddRoundKey**. In the 10th round the MixColumn step is omitted.

The 128 bit input is divided into 16 bytes of 8 bits apiece. These are arranged in a 4×4 matrix. The ShiftRow and MixColumn steps operate on this matrix while the ByteSub and AddRoundKey steps just operate on the bytes.

GF(256)

Some of the operations use the finite field GF($2^8 = 256$). The important features of this field are that each of its elements is represented by a single byte (8 bits), and one can add and multiply these bytes to get another byte. The construction of this field (and more importantly the multiplication rule) depends on a fixed irreducible polynomial of degree 8. The polynomial used for Rijndael is $x^8 + x^4 + x^3 + x + 1$.

ByteSub

In this step each byte of the matrix is replaced by another byte. The original byte is split into two 4 bit nibbles. The first (most significant) nibble is used as a row index and the second (least significant) nibble is used as a column index. These indices pick an entry in a 16×16 matrix whose entries are all the distinct bytes, and this byte replaces the original byte. This matrix is called an **S-box**. Unlike the S-boxes used in DES, the construction of this S-box is public (and one does not have to use the look-up table to find the value ... but this would require a larger program). To calculate the replacement byte, view the original byte as an element of GF(256) and first find its multiplicative inverse (if the original byte is the 0 byte, it is unchanged). Consider this inverse as a binary vector of length 8 and multiply it by a fixed 8×8 right circulant matrix (C (10001111)) and then add (XOR) a fixed vector (byte form of 99) to the result.

ShiftRow

The rows of the 4×4 matrix of bytes are cyclically shifted to the left. The first row is not shifted, the second shifted by one place, the third by two places and the fourth by three places.

$$\begin{matrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{matrix} \quad \rightarrow \quad \begin{matrix} A & B & C & D \\ F & G & H & E \\ K & L & I & J \\ P & M & N & O \end{matrix}$$

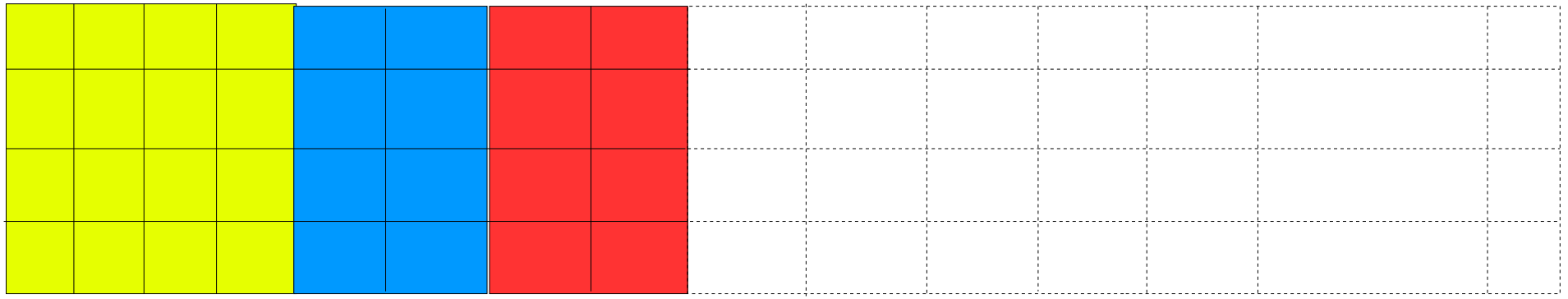
MixColumn

The 4×4 matrix is multiplied on the left by the matrix

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} A & B & C & D \\ F & G & H & E \\ K & L & I & J \\ P & M & N & O \end{pmatrix}$$

where the entries are written as bytes and the multiplication is done in GF(256).

Key Schedule



If $i \neq 4k$, $\mathbf{W(i) = W(i-4) \oplus W(i-1)}$ else $\mathbf{W(i) = W(i-4) \oplus T(W(i-1))}$

Cyclically shift the elements of the column up by one.

Using the S-box , replace the bytes in the column.

Compute the round constant, $00000010^{(i-4)/4}$ in GF(256) and add the result to the first byte of the column.

For the i^{th} round of Rijndael use the key which consists of the columns **W(4i), W(4i+1), W(4i+2)** and **W(4i+3)**.

Decrypting Rijndael

Rijndael is not symmetric (like DES), but the structure of the decryption algorithm is similar to the encryption algorithm.

The inverse of ByteSub is another lookup table, called **InvByteSub**. The inverse of ShiftRow is just shifting rows to the right instead of the left, called **InvShiftRow**. The matrix used in MixColumn is invertible, so **InvMixColumn** uses the inverse matrix:

$$\begin{pmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{pmatrix}$$

Finally, AddRoundKey is its own inverse.

Decrypting Rijndael

Inverting Rijndael with the keys used in reverse order is:

ARK ISR IBS
ARK IMC ISR IBS (9 rounds)
ARK

Since IBS acts on bytes and ISR moves bytes these operations commute. ARK and IMC however do not commute. "ARK then IMC" is the inverse of "MC then ARK". In matrix notation, "MC then ARK" is simply $Y = MX \oplus K$, where K is the round key matrix, M is the MixColumn matrix, X the current state of the working matrix and Y the matrix that results. The inverse is given by $X = M^{-1}(Y \oplus K) = M^{-1}Y \oplus M^{-1}K$. But this is just IMC applied to Y followed by adding IMC applied to K. By defining ***InvAddRoundKey*** to be XORing with IMC of K, we see that "ARK then IMC" can be replaced by "IMC then IARK".

Decrypting Rijndael

Now apply the switches to get:

ARK IBS ISR

IMC IARK IBS ISR

.....

IMC IARK IBS ISR

ARK

which can be rewritten as:

ARK

IBS ISR IMC IARK (9 rounds of this)

IBS ISR ARK

Thus we have the decryption algorithm written in the same form as the encryption algorithm, only replacing the steps by their inverses. The round keys of course have to be supplied in the reverse order.

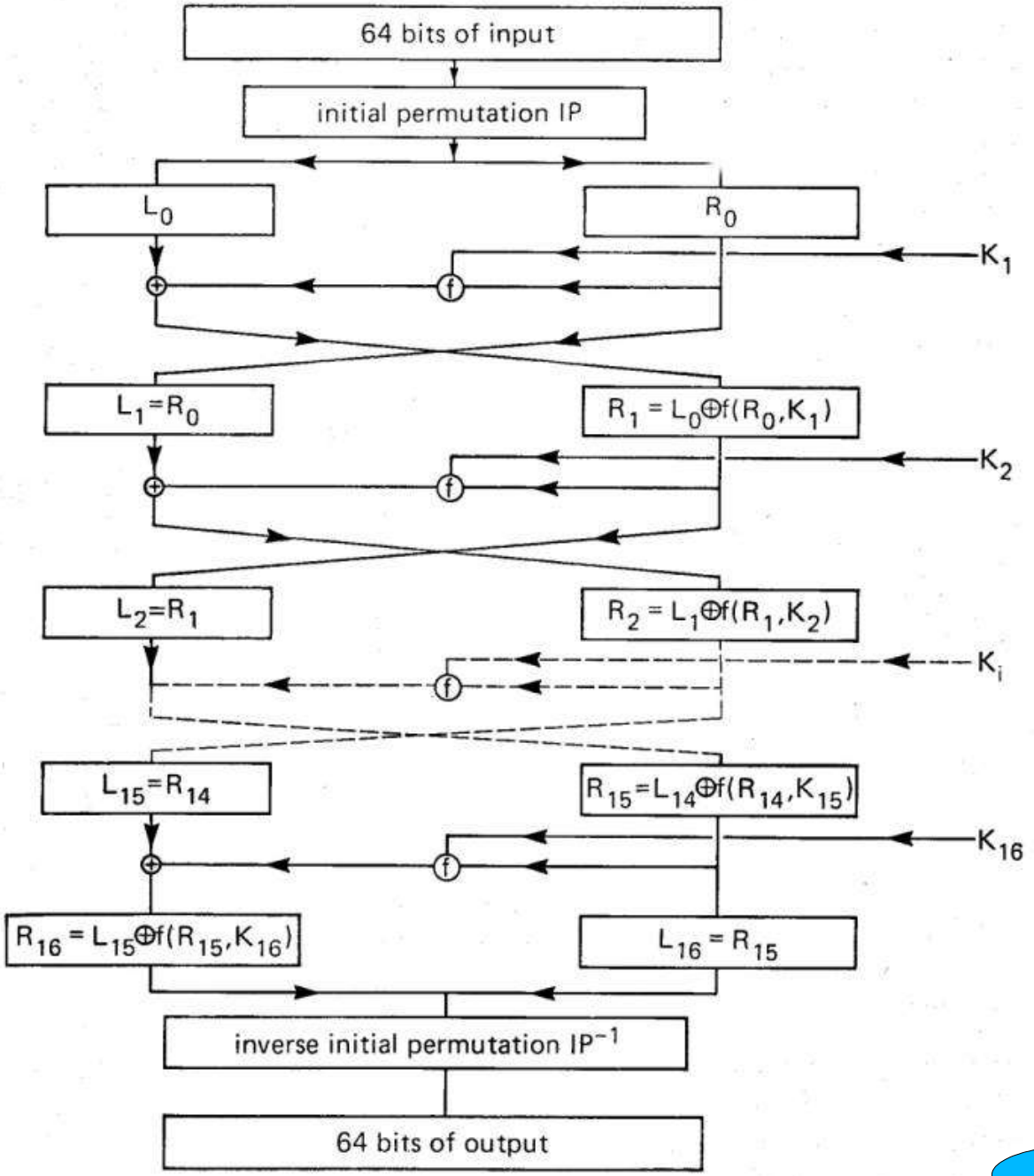


Figure 6.4 DES Encryption algorithm

Return

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

PC-1

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

PC-2

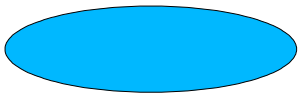
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

initial permutation IP

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

inverse initial permutation IP^{-1}

Table 6.3 IP and IP^{-1}



Return

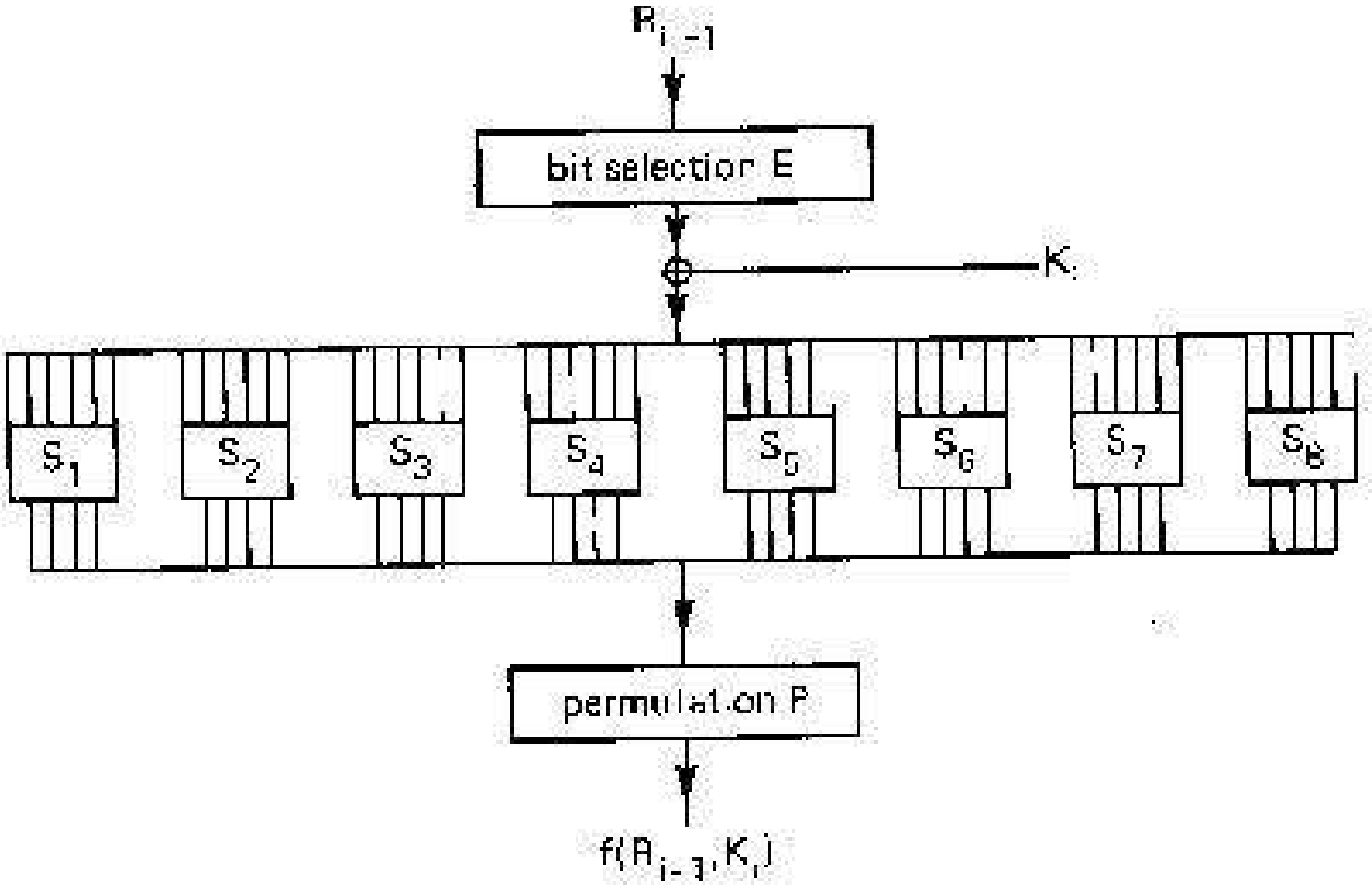


Figure 6.5 Calculation of $f(R_{i-1}, K_i)$

Return

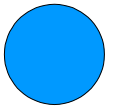
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Bit-selection table E

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

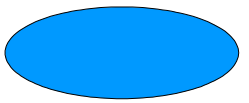
Permutation P

Table 6.6 Tables needed for the calculation of $f(R_{i-1}, K_i)$



		column															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
row	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	8
S_1	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
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
S_2	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
S_3	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
S_4	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
S_5	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
S_6	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
S_7	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
S_8	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 6.9 S-boxes (selection functions)



Return

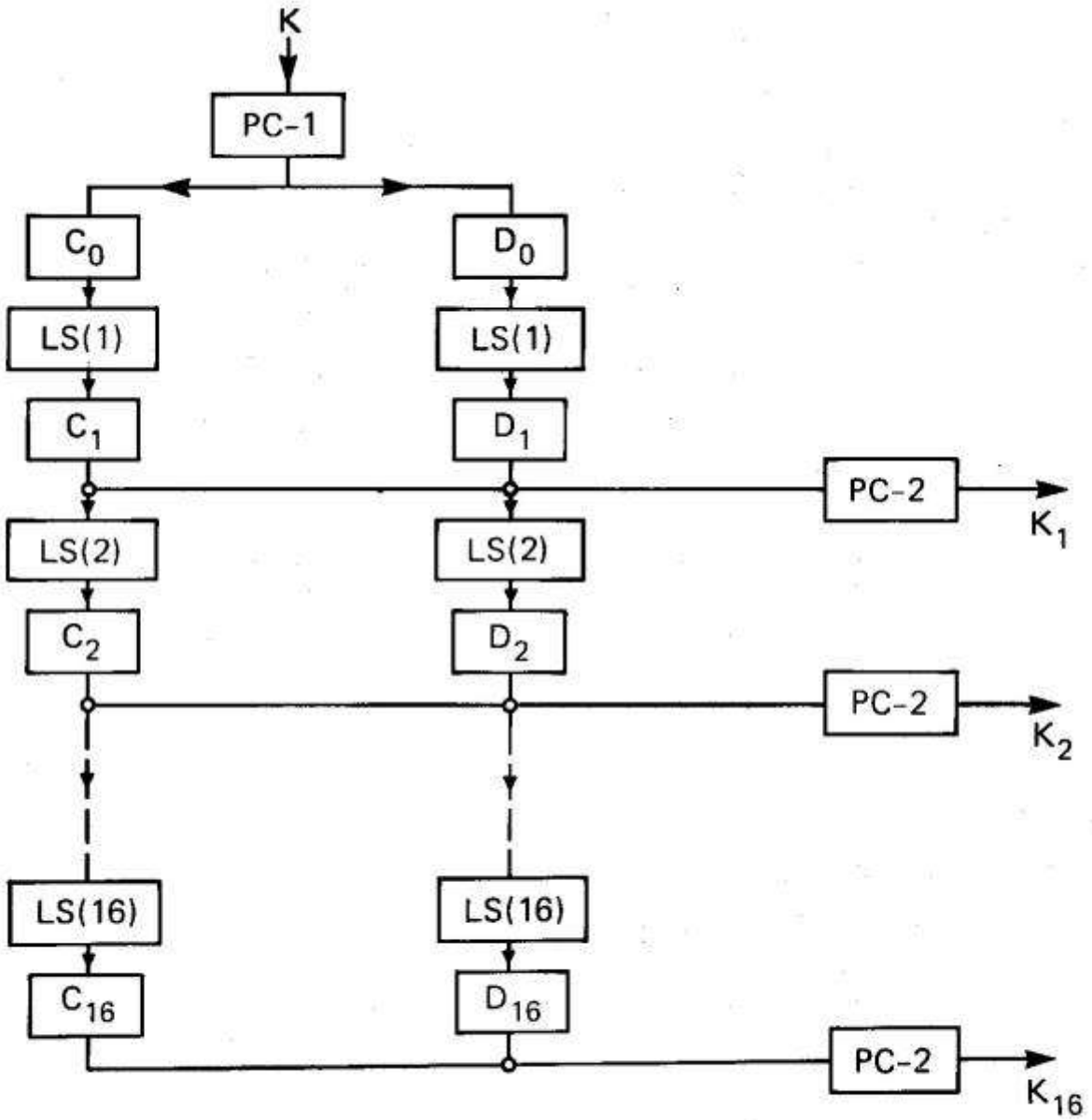


Figure 6.10 Key schedule calculation.

Return

round i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$LS(i)$	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1