CHAPTER 9

BLOCK-CIPHERS: LUCIFER, DES, AND AES

THE IBM Corporation decided to offer data security functionality using encryption for its customers in 1966. Horst Feistel (Fig. 9.1), who had previously worked in the cryptographic area, had developed a block cipher that was implemented in the IBM product for the Lloyd's bank. LUCIFER and its successor DES, had a profound effect on cryptography; it led to public-key cryptography, the active involvement of the university community, and changes in NSA. We review this development, the controversy surrounding DES, the replacement of DES by Rijndael, and the design of block ciphers.

9.1 LUCIFER

Horst Feistel's paper [Feistel, 1973] described the role cryptography might play in providing privacy in computer systems. The importance of this paper cannot be underestimated; first, it suggested a template for the design of cryptographic algorithms and second, it challenged the Government's undisputed role as master in the area of cryptology. It initiated a new era in cryptography that would lead to public-key cryptography. It was also of benefit to NSA, forcing it to re-examine its relationship with universities and business organizations.

Feistel's paper described LUCIFER, a *product block-cipher* enciphering plaintext data in blocks of *M* bits:

$$\underline{x} = (x_0, x_1, \ldots, x_{n-1}) \rightarrow \begin{pmatrix} x_0 & x_1 & \cdots & x_{M-1} \\ x_M & x_{M+1} & \cdots & x_{2M-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(n-1)M} & x_{(n-1)M+1} & \cdots & x_{nM-1} \end{pmatrix}.$$

Feistel used the APL programming language to experiment with and test LUCIFER. The program was stored in an APL-workspace, the analogue of a PC/MAC-folder and a UNIX-*directory*. The APL implementation, available at this time, imposed a limit on the number of letters in a workspace name. Feistel's original choice of DEMONSTRATION for the workspace name had to be shortened to DEMON; ultimately, someone suggested the sexier name LUCIFER.

A description of one version of LUCIFER may be found in Sorkin's paper [1984]. Outerbridge [1986] referred to LUCIFER as a Feistel-like block product cipher.

Figure 9.1 Horst Feistel (Courtesy of IBM).

In 1966, Lloyd's Banking contracted with IBM to design a remote-terminal-oriented banking system. The role of encipherment in ATM (Automated Teller Machine) transactions will be described in greater detail in Chapter 18, but it was clear that some sort of cryptographic capability would be needed. An algorithm proposed by another IBM division was rejected when it was recognized to be a variant of the Hill encipherment system (see Chapter 3). A group in the Mathematical Sciences Department at the IBM Yorktown Research Center including Roy Adler, Don Coppersmith, Horst Feistel, Edna Grossman, Alan Hoffman, Bryant Tuckerman, and myself had started in the 1960s to investigate encipherment. Feistel's `LUCIFER` was in the right place at the right time. Although IBM Research traditionally did *not* participate in product development, a good working relationship was established with a development group at the IBM division in Kingston, New York.

There are several versions of `LUCIFER`; for example Sorkin [1984] describes `LUCIFER` as it appears in a paper by Lynn Smith [1977]. I will describe the only commercial implementation of `LUCIFER`, contained in the IBM 2984 Cash Issuing Terminal.

Plaintext data of length $M = 32$ bits was enciphered following the paradigm proposed by Feistel, in which the plaintext $\underline{x}$ was viewed as consisting of equal length left (L) and right (R) blocks

$$\underline{x} = (x_0, x_1, \ldots, x_{M-1}) = (L, R).$$

`LUCIFER` enciphered plaintext $\underline{x}$ in 16 *rounds*, each round using a key-dependent transformation:

1. The two message halves (L, R) were transformed to $\mathcal{T}\colon (L, R) \to (F(R) + L, R)$, where $F(R)$ is a 16-bit to 16-bit mapping applied to the message right block R composed of

   L1. Modulo 16 addition of 16 bits of key to the block R,

   L2. Transformation then of the 16 bits by a nonlinear substitution *S-box*,

   L3. Transformation then of the 16 bits by a *P-box*,

   L4. Finally the addition of the result $F(R)$ to the 32-bit left block L.

2. The two halves $F(R) + L$ and R were interchanged $\vartheta\colon (F(R) + L, R) \to (R, F(R) + L)$.

$\mathcal{T}$ and ϑ are involutions

$$\mathcal{T}^{-1} = \mathcal{T}$$

$$(L, R) \xrightarrow{\mathcal{T}} (F(R) + L, R) \xrightarrow{\mathcal{T}} (F(R) + F(R) + L, R) = (L, R)$$

$$\vartheta^{-1} = \vartheta$$

$$(L, R) \xrightarrow{\vartheta} (L, R) \xrightarrow{\vartheta} (R, L)$$

where the *round transformation* $\mathcal{R} = \vartheta\mathcal{T}$ is invertible for every possible function F and

$$\mathcal{R}^{-1} = \mathcal{T}\vartheta.$$

F is a nonlinear transformation on 32-bit data strings in the IBM 2984 Cash Issuing Terminal.

The parameters of the 2984 implementation of LUCIFER [IBM, 1971] are:

2984-1: The block length $M = 32$ bits;

2984-2: Key length 64 bits; and

2984-3: 16 rounds in an encipherment.

A total of 36 bits are used on each round; each key bit is used $9 = \dfrac{16 \times 36}{64}$ times:

K1. 16 bits in Step L1;

K2. 4 bits in Step L2; and

K3. 16 bits in Step L3.

The 2984 LUCIFER-*schedule* specifies the 36 bits used in each round as follows:

KS0. The 64-bit LUCIFER-key is loaded into a *key register* and cyclically left-shifted 28 bit-positions;

KS1. The leftmost 36 bits used in a round are labeled as 4-bit *nibbles* `a, b, c, ..., i,`

KS2. The 4 bits in nibble `a` are used in the S-box transformation;

KS3. The 16 bits in nibbles `b, d, f,` and `h` are used in key-dependent L1 addition with carry;

KS4. The 16 bits in nibbles `c, e, g,` and `i` are used in the P-box transformation;

KS5. The key register is cyclically left-shifted 28 positions after each round.

The nibbles used in each round are shown in Table 9.1.

There are two different S-box mappings S_0 and S_1:

S_0 If $a_i = 0$, then S-box $\mathbf{S}_0$ transforms the input 4-bit data $\underline{d}$;

S_1 If $a_i = 1$, then S-box $\mathbf{S}_1$ transforms the input 4-bit data $\underline{d}$.

The nibble `a` $= (a_0, a_1, a_2, a_3)$ determines which of the $2^4 = 16$ possible S-box combinations is used to transform the 16 bits of data $\underline{r}$ in Step L2 as specified by the next equation and Table 9.2.

$$\underline{d} = (d_0, d_1, d_2, d_3) \rightarrow (\mathbf{S}_{a_0}(\underline{d}), \mathbf{S}_{a_1}(\underline{d}), \mathbf{S}_{a_2}(\underline{d}), \mathbf{S}_{a_3}(\underline{d}))$$

TABLE 9.1 IBM 2984 Key Register Schedule

Round r	Nibbles Used in Round r								
	a	b	c	d	e	f	g	h	i
1	7	8	9	10	11	12	13	14	15
2	14	15	0	1	2	3	4	5	6
3	5	6	7	8	9	10	11	12	13
4	12	13	14	15	0	1	2	3	4
5	3	4	5	6	7	8	9	10	11
6	10	11	12	13	14	15	0	1	2
7	1	2	3	4	5	6	7	8	9
8	8	9	10	11	12	13	14	15	0
9	15	0	1	2	3	4	5	6	7
10	6	7	8	9	10	11	12	13	14
11	13	14	15	0	1	2	3	4	5
12	4	5	6	7	8	9	10	11	12
13	11	12	13	14	15	0	1	2	3
14	2	3	4	5	6	7	8	9	10
15	9	10	11	12	13	14	15	0	1
15	0	1	2	3	4	5	6	7	8

TABLE 9.2 IBM 2984 S-Box Output

$\underline{d}$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_0(\underline{d})$	3	0	8	5	1	2	4	F	D	9	C	E	6	B	A	7
$S_1(\underline{d})$	8	D	1	6	c	4	F	B	3	2	5	4	9	0	7	A

The 2984 LUCIFER P-box is a key-dependent mapping of 32 bits to 32 bits;The 2984 LUCIFER P-box is a key-dependent mapping of 32 bits to 32 bits; Table 9.3 specifies how the nibbles c, e, g, and i are used in the P-box transformation where $'$ denotes the complement operation:

$$(\underline{T}_0, \underline{T}_1, \underline{T}_2, \underline{T}_3) \xrightarrow{P} (\underline{P}_0, \underline{P}_1, \underline{P}_2, \underline{P}_3)$$

$$T_i = (T_{i,0}, T_{i,1}, T_{i,2}, T_{i,3}) \qquad \underline{P}_i = (P_{i,0}, P_{i,1}, P_{i,2}, P_{i,3}), \qquad 0 \le i < 4$$

The input and output vectors to the P-box vectors,

$$\underline{T} = (T_{0,0}, T_{0,1}, T_{0,2}, T_{0,3}, \ldots, T_{3,0}, T_{3,1}, T_{3,2}, T_{3,3})$$

$$\underline{P} = (P_{0,0}, P_{0,1}, P_{0,2}, P_{0,3}, \ldots, P_{3,0}, P_{3,1}, P_{3,2}, P_{3,3})$$

TABLE 9.3 IBM 2984 P-Box Transformation

$P_{0,0} = T_{0,0}\mathtt{i}'_0 + T_{1,0}\mathtt{g}_0$	$P_{0,1} = T_{3,1}\mathtt{c}'_1 + T_{2,1}\mathtt{e}_1$	$P_{0,2} = T_{2,2}\mathtt{e}'_2T_{3,2}\mathtt{c}_2$	$P_{0,3} = T_{1,3}\mathtt{g}'_3 + T_{0,3}\mathtt{i}_3$
$P_{1,0} = T_{1,0}\mathtt{g}'_0 + T_{2,0}\mathtt{e}_0$	$P_{1,1} = T_{0,1}\mathtt{i}'_1 + T_{3,1}\mathtt{c}_1$	$P_{1,2} = T_{3,2}\mathtt{c}'_2 + T_{0,2}\mathtt{i}_2$	$P_{1,3} = T_{2,3}\mathtt{e}'_3 + T_{1,3}\mathtt{g}_3$
$P_{2,0} = T_{2,0}\mathtt{e}'_0 + T_{3,0}\mathtt{c}_0$	$P_{2,1} = T_{1,1}\mathtt{g}'_1 + T_{0,1}\mathtt{i}_1$	$P_{2,2} = T_{0,2}\mathtt{i}'_2 + T_{1,2}\mathtt{g}_2$	$P_{2,3} = T_{3,3}\mathtt{c}'_3 + T_{2,3}\mathtt{e}_3$
$P_{3,0} = T_{3,0}\mathtt{c}'_0 + T_{0,0}\mathtt{i}_0$	$P_{3,1} = T_{2,1}\mathtt{e}'_1 + T_{1,1}\mathtt{g}_1$	$P_{3,2} = T_{1,2}\mathtt{g}'_2 + T_{2,2}\mathtt{e}_2$	$P_{3,3} = T_{0,3}\mathtt{i}'_3 + T_{3,3}\mathtt{c}_3$

are related by

$$P = \begin{pmatrix}
i'_0 & 0 & 0 & 0 & g_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_1 & 0 & 0 & 0 & c'_1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e'_2 & 0 & 0 & 0 & c_2 & 0 \\
0 & 0 & 0 & i_3 & 0 & 0 & 0 & g'_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & g'_0 & 0 & 0 & 0 & e_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & i'_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_1 & 0 & 0 \\
0 & 0 & i_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c'_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & g_3 & 0 & 0 & 0 & e'_3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e'_0 & 0 & 0 & 0 & c_0 & 0 & 0 & 0 \\
0 & i_1 & 0 & 0 & 0 & g'_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & i'_2 & 0 & 0 & 0 & g_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_3 & 0 & 0 & 0 & c'_3 \\
i_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c'_0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & g_1 & 0 & 0 & 0 & e'_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & g'_2 & 0 & 0 & 0 & e_2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & i'_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_3
\end{pmatrix} \underline{T}$$

The IBM 2984 P-box is an invertible key-dependent linear transformation but *not* a permutation. Figure 9.2 is a block diagram of $\mathcal{T}$: $(L, R) \to (L + F(R), R)$.

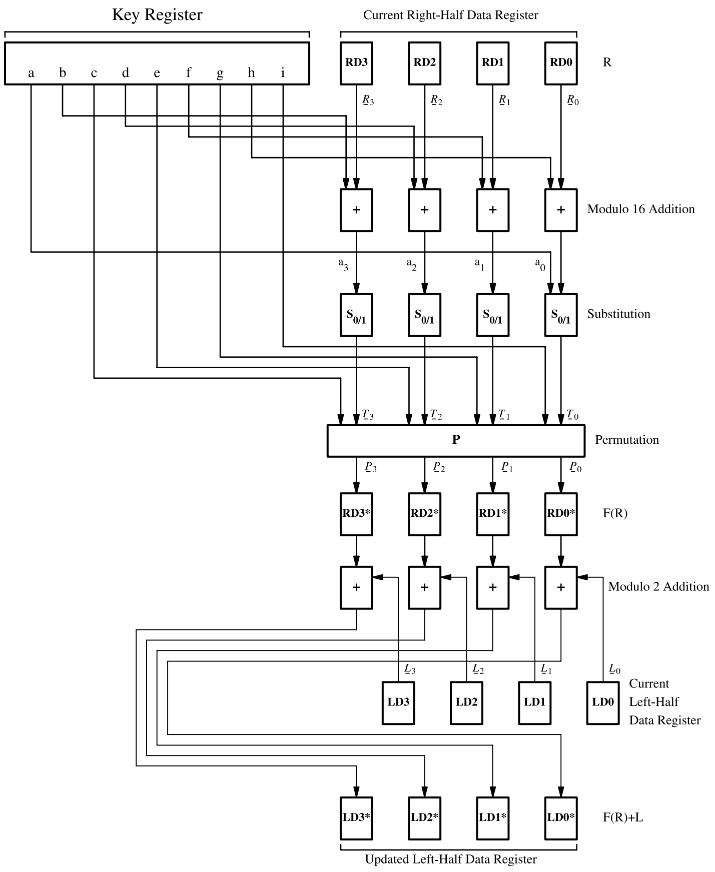


Figure 9.2 IBM 2984 round transformation $\mathcal{T}$.

9.2 DES

DES (Fig. 9.3) is a block cipher where

- plaintext $\underline{x} = (x_0, x_1, \ldots, x_{63}) \in \mathcal{Z}_{64,2}$;
- ciphertext $\underline{y} = (y_0, y_1, \ldots, y_{63}) \in \mathcal{Z}_{64,2}$;
- key $\underline{k} = (k_0, k_1, \ldots, k_{55}) \in \mathcal{Z}_{56,2}$.

$$\mathbf{DES}: \underline{x} \to \underline{y} = \mathrm{DES}_{\underline{k}}\{\underline{x}\}$$

DES is the product (composition) of mappings

$$\mathrm{DES} = \mathrm{IP}^{-1} \times \mathcal{T}_{16} \times \theta \times \mathcal{T}_{15} \times \cdots \times \theta \times \mathcal{T}_2 \times \theta \times \mathcal{T}_1 \times \mathrm{IP}$$
$$\mathcal{T}_i : (\underline{x}_r, \underline{x}_R) \to (\underline{x}_L + F_i(\underline{x}_R), \underline{x}_R)$$

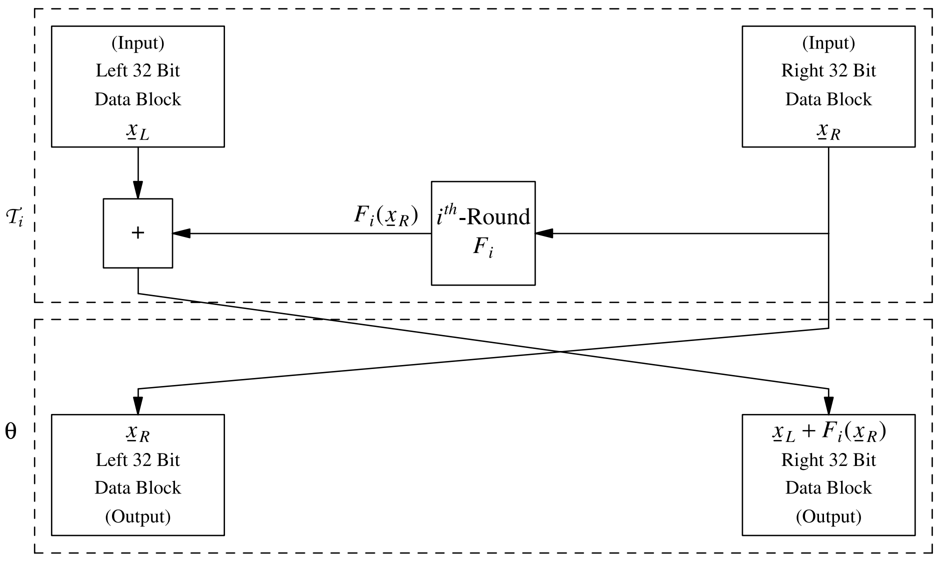
with inverse

$$\mathrm{DES}^{-1} = \mathrm{IP}^{-1} \times \mathcal{T}_1 \times \theta \times \mathcal{T}_2 \times \cdots \times \theta \times \mathcal{T}_{15} \times \theta \times \mathcal{T}_{16} \times \mathrm{IP}.$$

- IP is the *initial permutation* (or wire-crossing, plugboard);
- π_{T_i} F_i are the mappings performed on the left-x_L and right-x_R halves of the input on the ith round;
- θ is the *interchange involution*

$$\theta: (x_0, x_1, \ldots, x_{31}, x_{32}, x_{33}, \ldots, x_{63}) \to (x_{32}, x_{33}, \ldots, x_{63}, x_0, x_1, \ldots, x_{31})$$

The operations involved in the mapping $\mathcal{T}$ are portrayed in Figure 9.4.



(Input) Left 32 Bit Data Block $\underline{x}_L$
(Input) Right 32 Bit Data Block $\underline{x}_R$
$\mathcal{T}_i$
+
$F_i(\underline{x}_R)$
i^{th}-Round F_i
θ
$\underline{x}_R$ Left 32 Bit Data Block (Output)
$\underline{x}_L + F_i(\underline{x}_R)$ Right 32 Bit Data Block (Output)

Figure 9.3 DES.

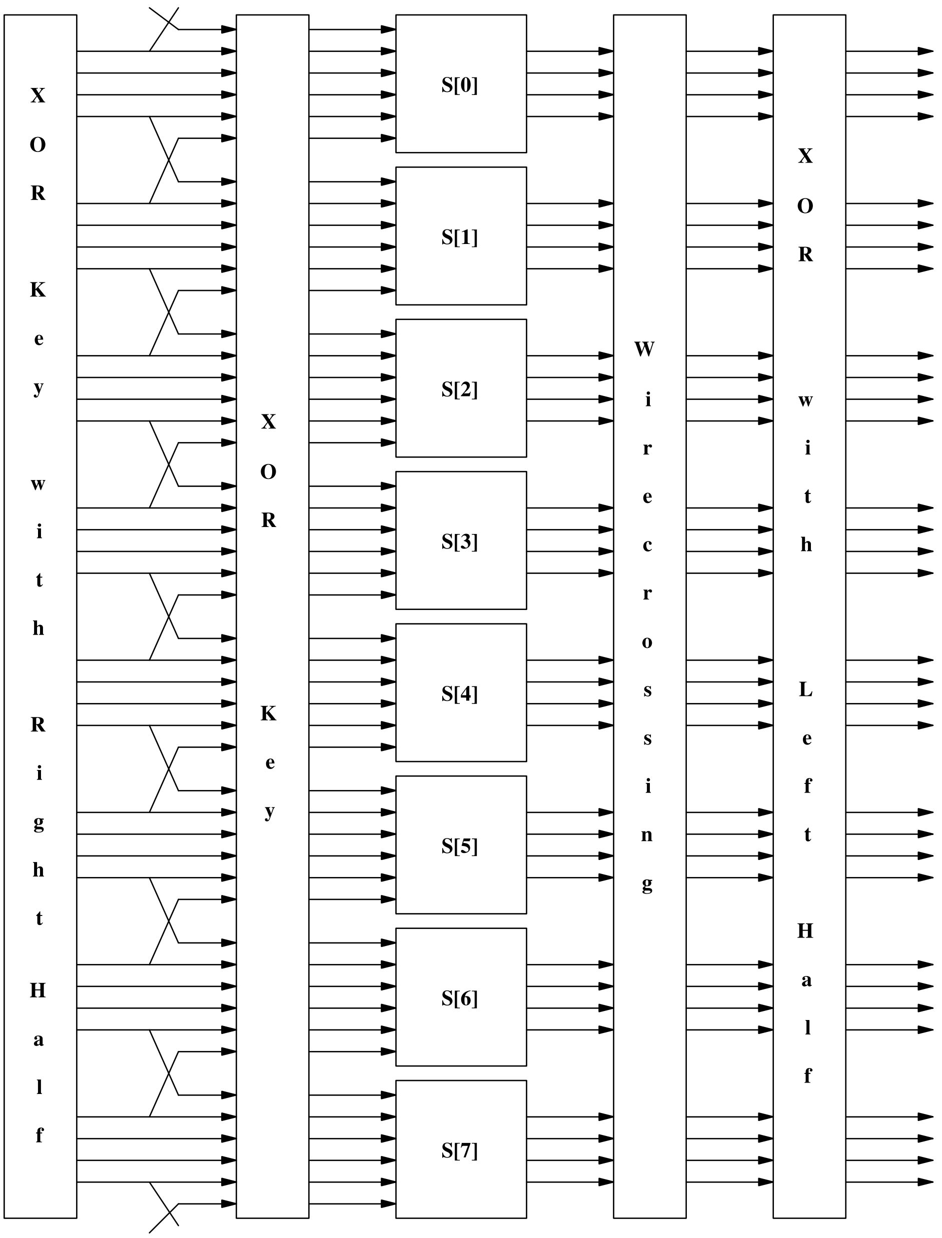


Figure 9.4 The DES transformation $\mathcal{T}$.

9.3 THE DES S-BOXES, P-BOX, AND INITIAL PERMUTATION (IP)

Tables 9.4 to 9.11 specify the seven DES S-boxes, each with a 6-bit input (x_0, x_1, x_2, x_3, x_4, x_5, x_6) and a 4-bit output (y_0, y_1, y_2, y_3); each table contains 4 rows and 15 columns, where

- Bits (x_0, x_6) identify a row in the table, and
- bits (x_1, x_2, x_3, x_4) identify a column in the table.

TABLE 9.4 DES S-Box S[0]

	S[0]															
	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	8
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

S[0] : $(x_0, \underbrace{x_1, x_2, x_3, x_4}_{\text{column}}, x_5) \rightarrow (y_0, y_1, y_2, y_3)$
$(1, 1, 0, 0, 1, 1)$: row 3, column 9, **S**[0]$(1, 1, 0, 0, 1, 1) = 11 = (1, 0, 1, 1)$

TABLE 9.5 DES S-Box S[1]

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

S[1] : $(x_0, x_1, x_2, x_3, x_4, x_5) \rightarrow (y_0, y_1, y_2, y_3)$
$(1, 1, 0, 0, 0, 0)$: row 2, column 8, **S**[1]$(1, 1, 0, 0, 0, 0) = 5 = (0, 1, 0, 1)$

TABLE 9.6 DES S-Box S[2]

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

S[2] : $(x_0, x_1, x_2, x_3, x_4, x_5) \rightarrow (y_0, y_1, y_2, y_3)$
$(0, 0, 1, 1, 1, 1)$: row 1, column 7, **S**[2]$(0, 0, 1, 1, 1, 1) = 10 = (1, 0, 1, 0)$

TABLE 9.7 DES S-Box S[3]

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

S[3] : $(x_0, x_1, x_2, x_3, x_4, x_5) \rightarrow (y_0, y_1, y_2, y_3)$
$(0, 0, 1, 1, 0, 0)$: row 0, column 6, **S**[3]$(0, 0, 1, 1, 0, 0) = 9 = (1, 0, 0, 1)$

TABLE 9.8 DES S-Box S[4]

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

S[4] : $(x_0, x_1, x_2, x_3, x_4, x_5) \rightarrow (y_0, y_1, y_2, y_3)$
(1, 0, 1, 0, 1, 1): row 3, column 5, S[4](1, 0, 1, 0, 1, 1) = 14 = (1, 1, 1, 0)

TABLE 9.9 DES S-Box S[5]

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

S[5] : $(x_0, x_1, x_2, x_3, x_4, x_5) \rightarrow (y_0, y_1, y_2, y_3)$
(1, 0, 1, 0, 0, 0): row 2, column 4, S[5](1, 0, 1, 0, 0, 0) = 2 = (0, 0, 1, 0)

TABLE 9.10 DES S-Box S[6]

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

S[6] : $(x_0, x_1, x_2, x_3, x_4, x_5) \rightarrow (y_0, y_1, y_2, y_3)$
(0, 0, 0, 1, 1, 1): row 1, column 3, S[6](0, 0, 0, 1, 1, 1) = 7 = (0, 1, 1, 1)

TABLE 9.11 DES S-Box S[7]

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

S[7] : $(x_0, x_1, x_2, x_3, x_4, x_5) \rightarrow (y_0, y_1, y_2, y_3)$
(1, 0, 0, 1, 0, 0): row 2, column 2, S[7](1, 0, 0, 1, 0, 0) = 4 = (0, 1, 0, 0)

My description of DES differs slightly from that given in [FIPS, 1988] in two respects:

- I use 0-index origin labeling; for example, a 64-bit plaintext block is $(x_0, x_1, \ldots, x_{63})$ instead of $(x_1, x_2, \ldots, x_{64})$.

TABLE 9.12 DES P-Box

15	6	19	20	28	11	27	16	0	14	22	25	4	17	30	9
1	7	23	13	31	26	2	8	18	12	29	5	21	10	3	24

TABLE 9.13 DES IP

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
56	48	40	32	24	16	8	0
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6

- FIPS 46 speaks of a 64-bit key, although only the first 7 bits in each byte play a role in the encipherment process.

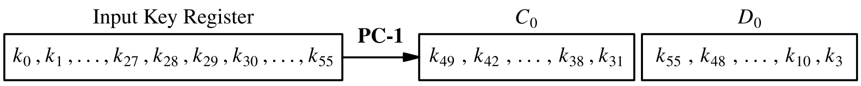
The P-box (Table 9.12) is a permutation of the 32-bit permutation $(x_0, x_1, \ldots, x_{31}) \rightarrow (x_{15}, x_6, \ldots, x_3, x_{24})$ DES plaintext $\underline{x}$ is first permuted by the *initial permutation* **IP** (Table 9.13) before the 16 round operations start:

$$\mathbf{IP} : (x_0, x_1, \ldots, x_{63}) \rightarrow (x_{57}, x_{49}, \ldots, x_{14}, x_6)$$

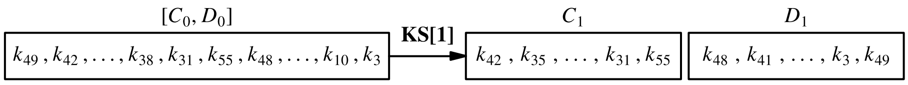
9.4 DES KEY SCHEDULE

Three arrays **PC-1**, **PC-2**, and **KS** specify the 48 key bits that are used on each round (Tables 9.14–9.16). The DES *key schedule* starts with the 56-bit *user key* $\underline{k} = (k_0, k_1, \ldots, k_{55})$ and derives 16 *internal keys* $\underline{k}_i = (k_{i,0}, k_{i,1}, \ldots, k_{i,47})$ with $0 \leq i < 16$, as shown in Figure 9.5. The 48-bit internal key $\underline{k}_i$ used on the ith round is derived as follows:

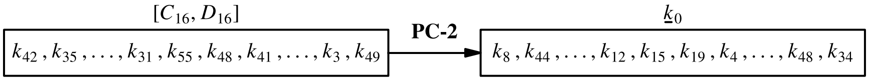
- **KS-1:** The user key $\underline{k}$ is inserted in two 28-bit registers $[C, D]$ according to (**PC-1**).


Input Key Register
C_0
D_0
PC-1
$k_0, k_1, \ldots, k_{27}, k_{28}, k_{29}, k_{30}, \ldots, k_{55}$
$k_{49}, k_{42}, \ldots, k_{38}, k_{31}$
$k_{55}, k_{48}, \ldots, k_{10}, k_3$

- **KS-2:** $[C_0, D_0]$ is the initial state of the registers $[C, D]$.
- **KS-3:** At the start of the ith round, the combined register-pair $[C_{i-1}, D_{i-1}]$ is left-circular shifted by $KS[i]$ positions, producing $[C_i, D_i]$. For example,


$[C_0, D_0]$
C_1
D_1
KS[1]
$k_{49}, k_{42}, \ldots, k_{38}, k_{31}, k_{55}, k_{48}, \ldots, k_{10}, k_3$
$k_{42}, k_{35}, \ldots, k_{31}, k_{55}$
$k_{48}, k_{41}, \ldots, k_3, k_{49}$

- **KS-4:** $\underline{k}_i$ is derived from the 28 bits of the concatenation of $[C_i, D_i]$ according to (**PC-2**).


$[C_{16}, D_{16}]$
$\underline{k}_0$
PC-2
$k_{42}, k_{35}, \ldots, k_{31}, k_{55}, k_{48}, k_{41}, \ldots, k_3, k_{49}$
$k_8, k_{44}, \ldots, k_{12}, k_{15}, k_{19}, k_4, \ldots, k_{48}, k_{34}$

Each bit of the user key is used about 13.7 times in a DES-encipherment. The key schedule is designed to use the key bits of k in as uniform a manner as possible.

TABLE 9.14 PC-1

pc-1						
49	42	35	28	21	14	7
0	50	43	36	29	22	15
8	1	51	44	37	30	23
16	9	2	52	45	38	31
55	48	41	34	27	20	13
6	54	47	40	33	26	19
12	5	53	46	39	32	25
18	11	4	24	17	10	3

TABLE 9.15 PC-2

pc-2					
13	16	10	23	0	4
2	27	14	5	20	9
22	18	11	3	25	7
15	6	26	19	12	1
40	51	30	36	46	54
29	39	50	44	32	47
43	48	38	55	33	52
45	41	49	35	28	31

TABLE 9.16 DES Key Shifts

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
KS[i]	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

It is intended that the key be *randomly* chosen. If $\underline{k}$ has special characteristics, the derived internal keys may fail to sufficiently disguise the plaintext. Of the 2^{56} possible keys, there are a few with this property.

9.4.1 Weak Keys

It was observed quite early in 1973 that certain user keys will produce internal keys with special regularity.

Example 9.1

The contents of the registers C_i and D_i contain a *constant* value so that DES is the 16th power of a transformation. For such a key $\underline{k}$

$$\underline{y} = \text{DES}\{\underline{k}, \underline{x}\} \longleftrightarrow \underline{x} = \text{DES}_{\underline{k}}^{-1}\{\underline{y}\}$$

There are four weak keys corresponding to the register contents $C, D \in \{(0)_{28}, (1)_{28}\}$, where $(0)_{28} \equiv \underbrace{(0, 0, \ldots, 0)}_{28 \text{ bits}}$ and $(1)_{28} \equiv \underbrace{(1, 1, \ldots, 1)}_{28 \text{ bits}}$.

Table 9.17 lists the weak keys written in hexadecimal notation, appending an odd parity check bit on the right. (Note, NIST (formerly NBS) often describes the DES key as a 64-bit key by appending a parity check bit on the right of each 7-bit block. Needless to say, this bit plays *no* role in the encipherment process.)

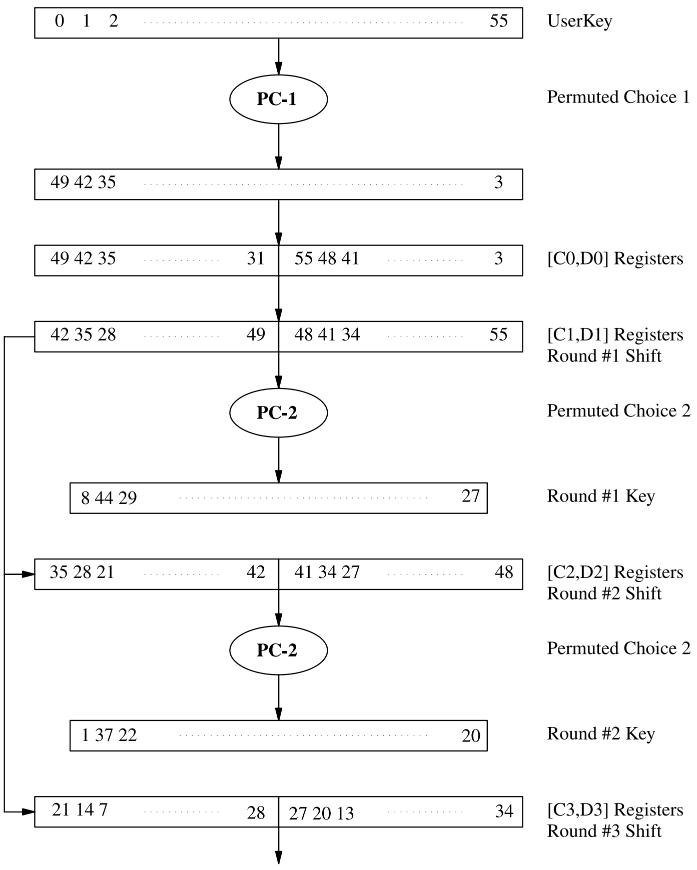


Figure 9.5 DES key schedule.

Example 9.2

A *semi-weak key* results when the contents of the registers C_i and D_i result in *at most* two internal keys. As the vector of key shifts in $KS = (1, 1, (2)_6, 1, (2)_6, 1)$, the only possible register values for C and D are in the set $\{(0,1)_{14}, (1,0)_{14}, (0)_{28}, (1)_{28}\}$.

Table 9.18 lists the six pairs of semi-weak keys in hex (with an odd parity check digit appended on the right).

9.5 SAMPLE DES ENCIPHERMENT

A trace of DES is shown next in Table 9.19 including

- The initialization, including the user key $\underline{k}$, the contents of the registers $[C_0, D_0]$, the plaintext $\underline{x}$, the result of the initial permutation IP$[\underline{x}]$, and the left and right data registers ($L[0]$, $R[0]$);

TABLE 9.17 Weak DES Keys

01	01	01	01	01	01	01	01	$C = (0)_{28}$	$D = (2)_{28}$
1F	1F	1F	1F	1F	1F	1F	1F	$C = (0)_{28}$	$D = (1)_{28}$
E0	E0	E0	E0	E0	E0	E0	E0	$C = (1)_{28}$	$D = (0)_{28}$
FE	FE	FE	FE	FE	FE	FE	FE	$C = (1)_{28}$	$D = (1)_{28}$

TABLE 9.18 Semi-Weak DES Keys

01	FE	01	FE	01	FE	01	FE	$C = (0, 1)_{14}$	$D = (0, 1)_{14}$
FE	01	FE	01	FE	01	FE	01	$C = (1, 0)_{14}$	$D = (1, 0)_{14}$
1F	E0	1F	E0	1F	E0	1F	E0	$C = (0, 1)_{14}$	$D = (1, 0)_{14}$
E0	1F	E0	1F	E0	1F	E0	1F	$C = (1, 0)_{14}$	$D = (0, 1)_{14}$
01	E0	01	E0	01	E0	01	E0	$C = (0, 1)_{14}$	$D = (0)_{28}$
E0	01	E0	01	E0	01	E0	01	$C = (1, 0)_{14}$	$D = (0)_{28}$
1F	FE	1F	FE	1F	FE	1F	FE	$C = (0, 1)_{14}$	$D = (1)_{28}$
FE	1F	FE	1F	FE	1F	FE	1F	$C = (1, 0)_{14}$	$D = (1)_{28}$
01	1F	01	1F	01	1F	01	1F	$C = (0)_{28}$	$D = (0, 1)_{14}$
1F	01	1F	01	1F	01	1F	01	$C = (0)_{28}$	$D = (1, 0)_{14}$
E0	FE	E0	FE	E0	FE	E0	FE	$C = (1)_{28}$	$D = (0, 1)_{14}$
FE	E0	FE	E0	FE	E0	FE	E0	$C = (1)_{28}$	$D = (1, 0)_{14}$

- The transformations on rounds 1, 2, and 16 displaying
  - The entering contents of the left- and right-half-data registers ($L[i-1], R[i-1]$),
  - The entering contents of the registers $[C_{i-1}, D_{i-1}]$,
  - The updated contents of the registers $[C_i, D_i]$,
  - The key KEY[i] used on Round i,
  - The expanded right data block $E[R[i-1]]$,
  - The XOR of KEY[i] and $E[R[i]]$,
  - The output of the S-boxes with input KEY[i] $+ E[R[i]]$,
  - The output of the P-box,
  - The entering left-half data register $L[i-1]$,
  - The XOR of the P-box output and the contents of $L[i-1]$,
  - The concatenation on the right of the P-BOX output $+ L[i-1]$ with $R[i-1]$,
  - The updated ($L[i], R[i]$), and
- The output.

9.6 CHAINING

The DES only specifies the encipherment a block of 64 bits. DES can be extended to encipher plaintext of arbitrary length in two ways.

The *Standard Extension of DES* divides the plaintext $\underline{x} = (x_0, x_1, \ldots, x_{N-1}) \in \mathcal{Z}_{N,2}$ into 8-byte blocks

$$\underline{x}^{(0)} = (x_0, x_1, \ldots, x_{63})$$

$$\underline{x}^{(1)} = x_{64}, x_{65}, \ldots, x_{127}$$

$$\vdots$$

$$\underline{x}^{(n-1)} = (x_{64(n-1)}, x_{64(n-1)+1}, \ldots, x_{64(n-1)})$$

and enciphers each block separately

$$\text{DES} : \underline{x}^{(i)} \to \underline{y}^{(i)} = \text{DES}_{\underline{k}}\{\underline{x}^{(i)}\}.$$

TABLE 9.19 Trace of DES

	Initialization
$\underline{k}$	0001 0011 0011 0100 0101 0011 0111 1001 1001 1011 1011 1100 1101 1111 1111 0001
$[C_0, D_0]$	1111000011001100101010101111 * 0101010101100010011110001111
$\underline{x}$	01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101
$\mathrm{IP}[\underline{x}]$	11111111 11111111 11111111 11111111 00000000 00000000 00000000 00000000
$(L[0], R[0])$	11111111 11111111 11111111 11111111 * 00000000 00000000 00000000 00000000
	Round 1
$(L[0], R[0])$	11111111 11111111 11111111 11111111 * 00000000 00000000 00000000 00000000
$[C_0, D_0]$	1111000011001100101010101111 * 0101010101100010011110001111
$[C_1, D_1]$	1110000110011001010101011111 * 1010101011000100111100011110
KEY[1]	000110 110000 001011 101111 011111 000111 000001 110010
$E[R[0]]$	000000 000000 000000 000000 000000 000000 000000 000000
$E[R[0]]$ + KEY[1]	000110 110000 001011 101111 011111 000111 000001 110010
S-BOX	0001 0101 0100 1000 0110 0010 1101 0110
P-BOX	0000 0010 0011 0111 0100 0000 1111 0011
$L[0]$	1111 1111 1111 1111 1111 1111 1111 1111
P-BOX + $L[0]$	1111 1101 1100 1000 1011 1111 0000 1100
(P-BOX + $L[0]$, $R[0]$)	11111101 11001000 10111111 00001100 * 00000000 00000000 00000000 00000000
$(L[1], R[1])$	00000000 00000000 00000000 00000000 * 11111101 11001000 10111111 00001100
	Round 2
$(L[1], R[1])$	00000000 00000000 00000000 00000000 * 11111101 11001000 10111111 0001100
$[C_1, D_1]$	1110000110011001010101011111 * 1010101011000100111100011110
$[C_2, D_2]$	1100001100110010101010111111 * 0101010110001001111000111101
KEY[2]	011110 011010 111011 011001 110110 101100 100111 100101
$E[R[1]]$	011111 111011 111001 010001 010111 111110 100001 011001
$E[R[1]]$ + KEY[2]	000001 100001 000010 001000 100001 010010 000110 111100
S-BOX	0000 1101 0000 0000 1011 1101 1110 0101
P-BOX	0011 0001 0001 1000 0110 1100 1011 1001
$L[1]$	0000 0000 0000 0000 0000 0000 0000 0000
P-BOX + $L[1]$	0011 0001 0001 1000 0110 1100 1011 1001
(P-BOX + $L[1]$, $R[1]$)	00110001 00011000 01101100 10111001 * 11111101 11001000 10111111 00001100
$(L[2], R[2])$	11111101 11001000 10111111 00001100 * 00110001 00011000 01101100 10111001
	Round 16
$(L[15], R[15])$	00110101 10010111 11000000 00101100 * 11001110 10010001 00110001 01100100
$[C_{15}, D_{15}]$	111100001100110010101010111 * 10101010101100010011110001111
$[C_{16}, D_{16}]$	1111000011001100101010101111 * 0101010101100010011110001111
KEY[16]	110010 110011 110110 001011 000011 100001 011111 100101
$E[R[15]]$	011001 011101 010010 100010 100110 100010 101100 001001
$E[R[15]]$ + KEY[16]	101011 101110 100100 101001 100101 000011 110011 101100
S-BOX	1001 0001 0100 1010 1100 1111 0101 1110
P-BOX	0001 1011 1111 0111 0110 0000 0110 1010
$L[15]$	0011 0101 1001 0111 1100 0000 0010 1100
P-BOX + $L[15]$	0010 1110 0110 0000 1010 0000 0100 0110
(P-BOX + $L[15]$, $R[15]$)	00101110 01100000 10100000 01000110 * 11001110 10010001 00110001 01100100
$(L[16], R[16])$	00101110 01100000 10100000 01000110 * 11001110 10010001 00110001 01100100
	Output
$\underline{y}$	00101000 11000001 11000011 11000000 00101000 01011110 10010011 10100100

There remains the question of how to handle the encipherment of plaintext whose length is not a multiple of $8n$ bytes. More importantly, there are instances in which the encipherment as defined above reveals structure in the plaintext. For example, when we encipher a file containing a picture, the outline of the picture might be detectable in the ciphertext. Also, stereotyped preambles of plaintext messages, like `Dear Mr./Ms.` or `To :` may be visible in the ciphertext. In order to hide the repetitive nature of plaintext and stereotyped preambles, *chaining* was introduced.

The *record chained* encipherment of plaintext $\underline{x} = (\underline{x}^{(0)}, \underline{x}^{(1)}, \ldots, \underline{x}^{(n-1)}, \underline{x}^{(n)})$ of length $8n + k$ bytes with $0 \leq k < 8$ by DES is defined as follows:

1. A nonsecret and randomly chosen 8-byte $\underline{y}^{(-1)}$ *initial chaining value* (ICV) prefixes the ciphertext.
2. The XOR of the ith block of plaintext $\underline{x}^{(i)}$, $0 \leq i < n$, with the $(i-1)$st block of ciphertext $\underline{y}^{(i-1)}$, enciphered by DES, becomes the ith block of ciphertext:

$$\text{Block Chained DES} : \underline{x}^{(i)} \to \underline{y}^{(i)} = \text{DES}_{\underline{k}}\{\underline{x}^{(i)} + \underline{y}^{(i-1)}\}, \qquad 0 \leq i < n.$$

3. If the length $8n + k$ of the plaintext is a multiple of 8 bytes ($k = 0$), the encipherment is complete; otherwise, the *final block* $\underline{x}^{(n)}$ of k-bytes is enciphered by first re-enciphering the $(n-1)$st block of ciphertext

$$\underline{z}^{(n-1)} = \text{DES}_{\underline{k}}\{\underline{y}^{(n-1)}\}$$

and thereafter XORing the leftmost k bytes of the result with $\underline{x}^{(n-1)}$

$$\underline{y}^{(n)} = \underline{x}^{(n)} + \text{Left}_k[\underline{z}^{(n-1)}]$$

where

$$\text{Left}_k[w_0, w_1, \ldots, w_{63}] = (w_0, w_1, \ldots, w_{8k-1})$$

On pages 275–277 of Konheim [1981] it is verified that record chaining is reversible and examples of chaining are given.

9.7 IS DES A RANDOM MAPPING?

In Section 10 of Chapter 8 the mappings of the set $\mathcal{Z}_m = \{0, 1, \ldots, m-1\}$ were described.

Proposition 9.1: If F is a randomly chosen one-to-one mapping of $\mathcal{Z}_m$ and Z is a randomly chosen element of $\mathcal{Z}_m$, then

9.1a The probability that Z belongs to an n-cycle in $\frac{1}{m}$ and

9.1b The average length of the cycle containing Z is $(m+1)/2$.

Proof: First an explanation of what is meant by "random"; there are $m!$ permutations of the elements of $\mathcal{Z}_m$. By the phrase "choose a mapping F randomly", we mean that the permutation F is selected with probability $1/m!$. Similarly, by the phrase "choose $Z \in \mathcal{Z}_m$ randomly", we mean that a particular $Z \in \mathcal{Z}_m$ is selected with probability $1/m$.

Let L denote the length of the cycle of F containing Z. As Z is to be any of m values, there are $(m-1)!$ permutations of the remaining $m-1$ elements so that

- The probability that $F(Z) = Z$ (meaning Z belongs to a 1-cycle) is $\Pr\{L=1\} = (m-1)!/m! = 1/m$.
- The probability that the nth iterate of F satisfies $F^n(Z)\begin{cases} \neq Z, & \text{if } n=1 \\ = Z & \text{if } n=2 \end{cases}$ is

$$\Pr\{L=2\} = \frac{(m-1)(m-2)!}{m!} = \frac{1}{m}.$$

This argument can be extended yielding $\Pr\{L=r\} = \dfrac{1}{m}$.

Next, we consider the analog of **Proposition 9.1** when F is not necessarily a one-to-one mapping of the elements of $\mathcal{Z}_m$.

As observed in Chapter 8, the orbits of mappings that are not one-to-one are composed of cycles with *tails* (Fig. 9.6). As the orbit of z must contain some repetition, we have $F^{(n)}(z) = F^{(j)}(z)$ with $0 \leq j < n$, where n is the first such repetition.

Proposition 9.2: If F is a randomly chosen mapping of $\mathcal{Z}_m$ and Z is a randomly chosen element of $\mathcal{Z}_m$, then

9.2a The probability that the first repeated element in the orbit $Z \rightarrow F(Z) \rightarrow F^{(2)}(Z) \rightarrow \cdots$ occurs at position $L = n$, $F^{(n)}(Z) = F^{(j)}(Z)$ for $0 \leq j < n$ is

$$\Pr\{L=n\} = \frac{n}{m}\prod_{i=0}^{n-1}\left(1-\frac{i}{m}\right)$$

$$\Pr\{L>n\} = \prod_{i=0}^{n-1}\left(1-\frac{i}{m}\right)$$

9.2b The expected value of L is asymptotically $\sqrt{0.5\pi m}$ as $m \rightarrow \infty$.

Proof: If $L = n$ then $Z, F^{(1)}(Z), \ldots, F^{(n-1)}(Z)$ are all distinct elements of $\mathcal{Z}_m$:

- There are $m(m-1)\,(m-2)\cdots(m-(n-1))$ possible choices for these elements;
- As $F^{(n)}(z) \in \{Z, F^{(1)}(Z), \ldots, F^{(n-1)}(Z)\}$, it must be one of n values;
- Each of the values of $F(Z)$ with $Z \notin \{Z, F^{(1)}(Z), \ldots, F^{(n-1)}(Z)\}$ may be chosen in m ways;

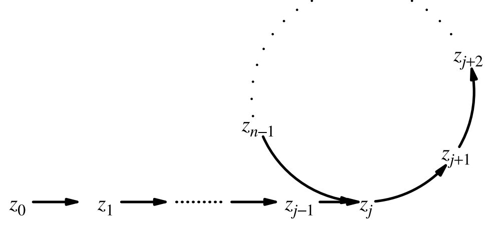


Figure 9.6 An orbit with a tail.

which gives

$$\Pr\{L = n\} = \frac{1}{m^m}[m \times (m-1) \times \cdots \times (m-(n-1)) \times n \times m^{m-n}],$$

proving **Proposition 9.2a**. The expectation of L is

$$E\{L\} = \sum_{n=1}^{m-1} \Pr\{L > n\} = \sum_{n=1}^{m-1} \prod_{n=0}^{m-1} \left(1 - \frac{i}{m}\right).$$

If $\frac{i}{m}$ is small, so that the approximation $1 - \frac{i}{m} \approx e^{-\frac{i}{m}}$.

$$\prod_{i=1}^{n-1} \left(1 - \frac{i}{m}\right) \approx c^{\frac{n^2}{2m}}.$$

This suggests that as $m \to \infty$

$$E\{L\} \approx \sum_{n=1}^{m-1} e^{-\frac{n^2}{2m}} \approx \sqrt{m} \sum_{s \in \{\sqrt{m}, \sqrt{2m} \dots\}} m^{-\frac{1}{2}} e^{\frac{s^2}{2}}.$$

The last summation approximates the Riemann integral $\int_0^\infty e^{-\frac{x^2}{2}} dx$, which gives Proposition **9.2b**. To prove that the approximation of the product by the exponential is valid, the summation for $E\{L\}$ is divided into two parts Σ_1 and Σ_2; the terms with $n \leq B$ are included in Σ_1 and tail terms with $n < B$ in Σ_2.

The approximation is valid for the terms in Σ_1; the second sum Σ_2 converges to 0.

9.8 DES IN THE OUTPUT-FEEDBACK MODE (OFB)

DES may be used to generate a key stream to be XORed to plaintext. DES is the *output feedback mode* (OFB) (Fig. 9.7) and starts with

1. A *non*secret *initial seed* $\underline{z}^{(0)} = (z_0^{(0)}, z_1^{(0)}, \dots, z_{63}^{(0)}) \in \mathcal{Z}_{64,2}$;
2. A key $\underline{k} = (k_0, k_1, \dots, k_{55}) \in \mathcal{Z}_{56,2}$; and
3. A *feedback parameter* m with $1 \leq m \leq 64$.

The key stream $\{\underline{z}^{(i)}: 1 \leq i < \infty\}$ is defined by

$$\underline{z}^{(i)} = \text{Right}_{64-m}(\underline{z}^{(i-1)}), \text{Left}_m \text{DES}_{\underline{k}}\{\underline{z}^{(i-1)}\}$$

where Right_m and Left_m take the *rightmost* and *leftmost* m bits of $\underline{w}$:

$$\text{Right}_m(w_0, w_1, \dots, w_{63}) = (w_{64-m}, w_{65-m}, \dots, w_{63}) \in \mathcal{Z}_{m,2}$$

$$\text{Left}_m(w_0, w_1, \dots, w_{63}) = (w_0, w_1, \dots, w_{m-1}) \in \mathcal{Z}_{m,2}$$

is XORed to plaintext to create the ciphertext.

When $m = 64$, the output-feedback mode mapping depicted in Fig. 9.7 is a one-to-one mapping of $\mathcal{Z}_{64,2}$ onto itself. The average cycle length is 2^{63}.

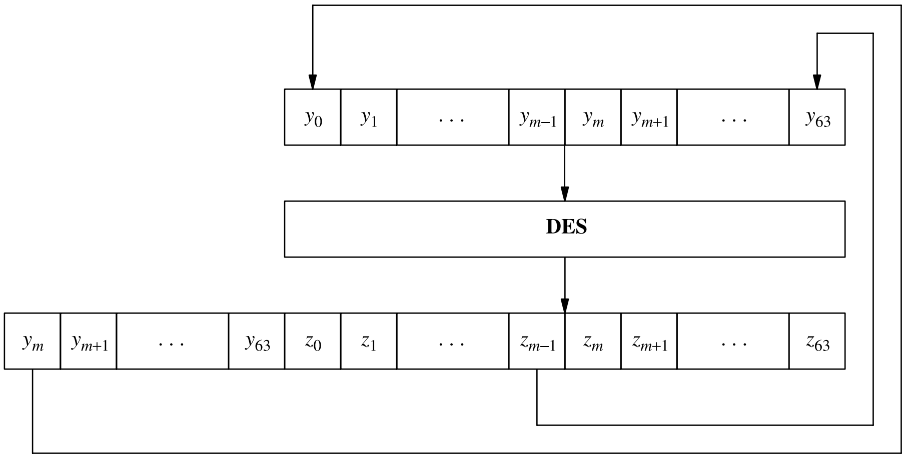


Figure 9.7 Output feedback mode.

When $m < 64$, the OFB mapping is not one to one and its cycle length is $O(2^{32})$, an observation first made by Davies and Parkin [1982]. This means that in a large ciphertext file with $m = 1$, we are likely to see the same key bit used to encipher different bits of the plaintext. And why should any value of $m < 64$ be used?

9.9 CRYPTANALYSIS OF DES

The exportation of American technology is regulated by the Bureau of Export Administration: Office of Strategic Trade and Foreign Policy, an agency within the U.S. Department of Commerce. A list of products covered by 15 CFR chapter VII, subchapter C may be found on the Web page www.bxa.gov; included are commercial encryption devices.

The `LUCIFER` cryptographic facility was incorporated into the IBM Liberty Banking System and a patent application protecting the technology was filed by IBM. United States Patent Office rules require a patent to be first filed in the United States and reviewed by the Patent Office before foreign patent coverage can be sought. In cases where the publication of an application or the granting of a patent would be detrimental to national security, the Commissioner of Patents may issue a *secrecy order* to stop the patent process. This

1. Requires that the invention be kept secret,
2. Withholds the publication of the application or the grant of the patent for such period as the national interest requires,
3. Forbids the dissemination of material related to the patent by all parties, and
4. Restricts filing of foreign patent applications.

The owner of an application that has been placed under a secrecy order has a right to appeal the order to the Secretary of Commerce, 35 U.S.C. 181. If no secrecy order is issued in the six months after the submission of a U.S. patent application, patent applications may be filed outside the United States.

IBM's patent application did not result, in a secrecy order, probably because it dealt with a "banking system" and did not describe the encryption process in the terms NSA is familiar with. IBM filed in France six months later and was granted a patent.

Up to this point, NSA had been the undisputed center of cryptographic competence in the United States. As the cat was out of the bag, so to speak, NSA decided that it would have to "influence" the direction of commercial cryptography rather than forbid it. The IBM Corporation developed a follow-on to LUCIFER in response to encouragement from NSA, and submitted the "improved" algorithm to NBS for certification as a national standard.

During the design of DES, certain desirable properties of S-boxes were formulated. With the exception of three criteria fisted below, they have never been made public at the request of NSA.

C1. No S-box is either a linear or affine[1] function.

C2. S-box constraints

C2a. Changing one bit in the input of an S-box resulting in at least two output bits changing;

C2b. If two inputs to an S-box differ in the middle two bits, their outputs must be different by at least two bits [Coppersmith, 1993];

C2c. If two inputs to an S-box differ in their first two bits and agree on their last two, their two outputs must differ;

C2d. For any nonzero 6-bit difference between S-box inputs, no more than 32 pairs of inputs exhibiting that difference may result in the same output difference.

A *twiddle* of a vector $\underline{x}$ is a vector $\underline{x} + \underline{y}$ that differs from $\underline{x}$ in at least one component. If the length of the XOR is small, say $|\text{ROM}[\underline{x}] + \text{ROM}[\underline{x} + \underline{y}]| < 2$, a twiddle could conceivably propagate in many rounds so that $|\mathcal{T}(\underline{x}) + \mathcal{T}(\underline{x} + \underline{y})|$ might also be small. If many different twiddles are present, they might lead to a determination of several key bits. During design of DES, twiddles were *not* excluded until it was discovered that they could be used as indicated above.

Searching the block cipher parameters for good *differential changes* – the new term for twiddles – and using them for cryptanalysis is the basic idea of Bilham's and Shamir's *differential* cryptanalysis [Bilham and Shamir, 1993], the closely related *linear cryptanalysis* of Matsui [1994], which searches for good linear approximations to the ROMs, and the recent paper by Bilham [1995].

[1]An S-box $F(x)$ is *linear* in $\underline{x} = (x_0, x_1, x_2, x_3, x_4, x_5)$ if

$$F(\underline{x}) = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

F is *affine* if

$$F(\underline{x}) = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} + \begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} & c_{0,4} & c_{0,5} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

C3. The S-boxes were chosen to minimize the differences between the number of 1's and 0's when any single bit is held constant.

This is essentially the criterion discovered by Matsui and the basis for the measure of nonlinearity.

As IBM did not reveal the design principles, there was the suspicion that Big-Blue and NSA had conspired to put a trap into the system. DES has been analyzed over the past 25 years and no *systemic* weakness has been found [Schneier, 1996]. Biham and Shamir [1992, 1993] wrote

The replacement of the order of the eight DES S-boxes (without changing their values) also makes DBS much weaker: DES with 16 rounds of a particular order is breakable in about 2^{38} steps. DES with random S-boxes is shown very easy to break. Even a minimal change of one entry of one of the DES S-boxes can make DES easier to break.

Of course, Bilham and Shamir may be wrong and, in retrospect, the key length of 56 bits seems to be inappropriate.

DES and the controversy stimulated a significant amount of research in the academic community on cryptography. It has produced an extensive literature dealing with the design of S-boxes, in particular with regarding the *nonlinearity* of an S-box [Nyberg, 1992; Seberry and Zheng, 1993; Charnes and Piepzryk, 1993; Detombe and Tavares, 1993; Seberry et al., 1994, 1995; O'Connor, 1995a, b].

9.10 DIFFERENTIAL CRYPTANALYSIS

Suppose two plaintexts are enciphered by S-box S[0] with the same key.

$$\underline{y}_1 = S[0](\underline{x}_1 + \underline{k}), \qquad \underline{y}_2 = S[0](\underline{x}_2 + \underline{k}).$$

We conclude that

$$\underline{y}_1 + \underline{y}_2 = S[0](\underline{x}_1 + \underline{k}) + S[0](\underline{x}_2 + \underline{k})$$

and write this last relationship as

$$S[0] : \Delta\underline{x} \to \Delta\underline{y}$$

where

$$\begin{aligned} &\text{(Input XOR)} & \Delta x &\equiv \underline{x}_1 + \underline{x}_2 = (\underline{x}_1 + \underline{k}) + (\underline{x}_2 + \underline{k}) \\ &\text{(Output XOR)} & \Delta\underline{y} &\equiv \underline{y}_1 + \underline{y}_2 \end{aligned}$$

How much of the 6-bit key is revealed by corresponding pairs of plain- and cipher-text $(\underline{x}_i, \underline{y}_i)$ $(i = 1, 2)$ enciphered by S-box S[0] with the same *un*known key? That is, how many solutions are there to

$$\underline{y}_1 = S[0](\underline{x}_1 + \underline{k}), \qquad \underline{y}_2 = S[0](\underline{x}_2 + \underline{k})$$

given

$$\Delta\underline{x} \equiv \underline{x}_1 + \underline{x}_2, \qquad \Delta\underline{y} \equiv \underline{y}_1 + \underline{y}_2.$$

Define

$$\mathcal{D}(\Delta\underline{x}, \Delta\underline{y}) \equiv \{(\underline{z}_1, \underline{z}_2) : \Delta\underline{x} = \underline{z}_1 + \underline{z}_2, \Delta\underline{y} = S[0](\underline{z}_1) + S[0](\underline{z}_2)\}.$$

A pair $(\underline{z}_1, \underline{z}_2)$ in $\mathcal{D}(\Delta\underline{x}, \Delta\underline{y})$ determines a possible unknown key by setting

$$\underline{k} = \underline{x}_1 + \underline{z}_1$$

If the size of $\mathcal{D}(\Delta\underline{x}, \Delta\underline{y})$ is much smaller than $64 = 2^6$, then the *differentials* $(\Delta\underline{x}, \Delta\underline{y})$ reveal a great deal of the key. Of course, the encipherment process described above uses only one S-box and it will be necessary to extend this to the full DES encipherment process. This is the principle of *differential cryptanalysis*, a *known corresponding plain/ciphertext* attack whose objective is to attempt to identify the *un*known key from corresponding plain/ciphertext differentials $(\Delta\underline{x}, \Delta\underline{y})$ enciphered with the same key.

Tables 9.20 and 9.21 list the size of the set $|\mathcal{D}(\Delta\underline{x}, \Delta\underline{y})|$ for all pairs of Input/Output XOR $(\Delta\underline{x}, \Delta\underline{y})$. A row is labeled by the 6-bit Input XOR $\Delta(\underline{x})$ (as two hex digits), a column

TABLE 9.20 $|\mathcal{D}(\Delta\underline{x}, \Delta\underline{y})|$ **for S-Box S[0]**

Input XOR	Output XOR of S[0]															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
02	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
03	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
04	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
05	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
06	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
07	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
08	0	0	0	12	0	8	8	4	0	6	2	8	8	2	2	4
09	10	2	4	0	2	4	6	0	2	2	8	0	10	0	2	12
0A	0	8	6	2	2	8	6	0	6	4	6	0	4	0	2	10
0B	2	4	0	10	2	2	4	0	2	6	2	6	6	4	2	12
0C	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
0D	6	6	4	8	4	8	2	6	0	6	4	6	0	2	0	2
0E	0	4	8	8	6	6	4	0	6	6	4	0	0	4	0	8
0F	2	0	2	4	4	6	4	2	4	8	2	2	2	6	8	8
10	0	0	0	0	0	0	2	14	0	6	6	12	4	6	8	6
11	6	8	2	4	6	4	8	6	4	0	6	6	0	4	0	0
12	0	8	4	2	6	6	4	6	6	4	2	6	6	0	4	0
13	2	4	4	6	2	0	4	6	2	0	6	8	4	6	4	0
14	0	8	8	0	10	0	4	2	8	2	2	4	4	8	4	0
15	0	4	6	4	2	2	4	10	6	2	0	10	0	4	6	4
16	0	8	10	8	0	2	2	6	10	2	0	2	0	6	2	6
17	4	4	6	0	10	6	0	2	4	4	4	6	6	6	2	0
18	0	6	6	0	8	4	2	2	2	4	6	8	6	6	2	2
19	2	6	2	4	0	8	4	6	10	4	0	4	2	8	4	0
1A	0	6	4	0	4	6	6	6	6	2	2	0	4	4	6	8
1B	4	4	2	4	10	6	6	4	6	2	2	4	2	2	4	2
1C	0	10	10	6	6	0	0	12	6	4	0	0	2	4	4	0
1D	4	2	4	0	8	0	0	2	10	0	2	6	6	6	14	0
1E	0	2	6	0	14	2	0	0	6	4	10	8	2	2	6	2
1F	2	4	10	6	2	2	2	8	6	8	0	0	0	4	6	4

by the Output XOR $\Delta(\underline{y})$ (as two hex digits) for the S-box S[0]; for example, the entry in row `1A` = (01 1010) and column `C` = (1100) is 4.

Note that

- The sum of the entries in a row is 64, the average is 4;
- The distribution of values in a row is *not* uniform; and
- If $\Delta\underline{x} \neq \underline{0}$; then if $(\underline{z}_1, \underline{z}_2)$ is in $\mathcal{D}(\Delta\underline{x}, \Delta\underline{y})$, then so is the pair $(\underline{z}_2, \underline{z}_1)$.

Example 9.3

Table 9.22 is derived from the row `34` data in Table 9.21 and the S-box description of S[0] in Table 9.4; it lists the input pairs $(\underline{z}_1, \underline{z}_2)$ (as two hex digits) that satisfy

$$\Delta\underline{x} = \underline{z}_1 + \underline{z}_2 = (1, 1, 0, 1, 0, 0) = 34 \qquad \text{(Input XOR)}$$

TABLE 9.21 $|\mathcal{D}(\Delta\underline{x}, \Delta\underline{y})|$ **for S-Box S[0]**

Input XOR	Output XOR of S[0]															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20	0	0	0	10	0	12	8	2	0	6	4	4	4	2	0	12
21	0	4	2	4	4	8	10	0	4	4	10	0	4	0	2	8
22	10	4	6	2	2	8	2	2	2	2	6	0	4	0	4	10
23	0	4	4	8	0	2	6	0	6	6	2	10	2	4	0	10
24	10	0	0	2	2	2	2	0	14	14	2	0	2	6	2	4
25	6	4	4	12	4	4	4	10	2	2	2	0	4	2	2	2
26	0	0	4	10	10	10	2	4	0	4	6	4	4	4	2	0
27	0	4	2	0	2	4	2	0	4	8	0	4	8	8	4	4
28	12	2	2	8	2	6	12	0	0	2	6	0	4	0	6	2
29	4	2	2	10	0	2	4	0	0	14	10	2	4	6	0	4
2A	4	2	4	6	0	2	8	2	2	14	2	6	2	6	2	2
2B	12	2	2	2	4	6	6	2	0	2	6	2	6	0	8	4
2C	4	2	2	4	0	2	10	4	2	2	4	8	8	4	2	6
2D	6	2	6	2	8	4	4	4	2	4	6	0	8	2	0	6
2E	6	6	2	2	0	2	4	6	4	0	6	2	12	2	6	4
2F	2	2	2	2	2	6	8	8	2	4	4	6	8	2	4	2
30	0	4	6	0	12	6	2	2	8	2	4	4	6	2	2	4
31	4	8	2	10	2	2	2	2	6	0	0	2	2	4	10	8
32	4	2	6	4	4	2	2	4	6	6	4	8	2	2	8	0
33	4	4	6	2	10	8	4	2	4	0	2	2	4	6	2	4
34	0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6
35	2	2	4	0	8	0	0	0	14	4	6	8	0	2	14	0
36	2	6	2	2	8	0	2	2	4	2	6	8	6	4	10	0
37	2	2	12	4	2	4	4	10	4	4	2	6	0	2	2	4
38	0	6	2	2	2	0	2	2	4	6	4	4	4	6	10	10
39	6	2	2	4	12	6	4	8	4	0	2	4	2	4	4	0
3A	6	4	6	4	6	8	0	6	2	2	6	2	2	6	4	0
3B	2	6	4	0	0	2	4	6	4	6	8	6	4	4	6	2
3C	0	10	4	0	12	0	4	2	6	0	4	12	4	4	2	0
3D	0	8	6	2	2	6	0	8	4	4	0	4	0	12	4	4
3E	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
3F	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

for all S-box S[0] Output XORs $\Delta\underline{y} \neq (0, 0, 0, 0)$ (as two hex digits) and is constructed as follows:

- For each pair of input values to S-box S[0], $(z_0, z_1, z_2, z_3, z_4, z_5)$ and $(z'_0, z'_1, z'_2, z'_3, z'_4, z'_5)$ for which $\underline{z}_1 + \underline{z}_2 = (1, 1, 0, 1, 0, 0) = (z_0 + z'_0, z_1 + z'_1, z_2 + z'_2, z_3 + z'_3, z_4 + z'_4, z_5 + z'_5)$, compute
- $\underline{y}_1 = S[0](z_0, z_1, z_2, z_3, z_4, z_5)$ and $\underline{y}_2 = S[0]\ (z'_0, z'_1, z'_2, z'_3, z'_4, z'_5)$.

There is an entry (in hex)

$$\begin{pmatrix} z_0, z_1, z_2, z_3, z_4, z_5 \\ z'_0, z'_1, z'_2, z'_3, z'_4, z'_5 \end{pmatrix}$$

in Table 9.13C provided $\Delta\underline{y} = \underline{y}_1 + \underline{y}_2$. For example,

- Table 9.21 shows that there are 8 pairs $(\underline{z}_1 + \underline{z}_2)$ if $\Delta\underline{y} = (0, 0, 0, 1) = \mathtt{1}$;
- The entry $\begin{pmatrix} \mathtt{03} \\ \mathtt{37} \end{pmatrix}$ in Table 9.22 corresponds to

$$\underline{z}_1 = (0, 0, 0, 0, 1, 1),\ \underline{z}_2 = (1, 1, 0, 1, 1, 1) \text{ with sum } (1, 1, 0, 1, 0, 0);$$

- Table 9.4 shows the (row 1, column 1) S-box S[0] entry for $\underline{z}_1 = (0, 0, 0, 0, 1, 1)$ is $15 = (1, 1, 1, 1)$;
- Table 9.4 shows the (row 3, column 11) S-box S[0] entry for $\underline{z}_2 = (1, 1, 0, 1, 1, 1)$ is $14 = (1, 1, 1, 0)$;
- The sum of these two S-box S[0]-entries is $1 = (1, 1, 1, 1) + (1, 1, 1, 0)$.

leading to the entry $\begin{pmatrix} \mathtt{03} \\ \mathtt{37} \end{pmatrix}$ in the row corresponding to $\Delta\underline{y} = 1$.

TABLE 9.22 Input Pairs $(\underline{z}_1, \underline{z}_2)$ Satisfying $\underline{z}_1 + \underline{z}_2 = (1, 1, 0, 1, 0, 0)$

$\Delta\underline{y}$	$\underline{z}_1 + \underline{z}_2 = (1, 1, 0, 1, 0, 0)$															
1	03	0F	1E	1F	2A	2B	37	3B								
	37	3B	2A	2B	1E	1F	03	0F								
2	04	05	0E	11	12	14	1A	1B	20	25	26	2E	2F	30	31	3A
	30	31	3A	25	26	20	2E	2F	14	11	12	1A	1B	04	05	0E
3	01	02	15	21	35	36										
	35	36	25	15	11	10										
4	13	27														
	27	13														
7	00	08	0D	17	18	1D	23	29	2C	34	39	3C				
	34	3C	39	23	2C	29	17	1D	18	00	0D	08				
8	09	0C	19	2D	38	3D										
	3D	38	2D	19	0C	09										
D	06	10	16	1C	22	24	28	32								
	32	24	22	28	16	12	1C	06								
F	07	0A	0B	33	3E	3F										
	33	3E	3F	07	0A	0B										

Example 9.3 (continued)
Suppose we have

- $\underline{x}_1 = (1, 1, 1, 1, 1, 1)$, $\underline{x}_2 = (0, 0, 1, 0, 1, 0)$, $\Delta\underline{x} = (1, 1, 0, 1, 0, 0)$ and
- $\underline{y}_1 = S[0](\underline{x}_1 + \underline{k}) = (0, 1, 1, 0)$, $\underline{y}_2 = S[0](\underline{x}_2 + \underline{k}) = (0, 0, 1, 0)$.

There is essentially one entry in Table 9.22 and it shows that

- $\underline{z}_1 = (0, 1, 0, 0, 1, 1) = \texttt{13}$, $\underline{z}_2 = (1, 0, 0, 1, 1, 1) = \texttt{27}$ satisfies $\underline{z}_1 + \underline{z}_2 = (1, 1, 0, 1, 0, 0) = \texttt{34}$ and
- There are *only* two possible keys $\underline{k} = (1, 0, 1, 1, 0, 0) = \texttt{2C}$ and $\underline{k} = (0, 1, 1, 0, 0, 0) = \texttt{18}$.

The inference of the key in *Example 9.3* can be generalized to a one-round characteristic of DES as shown in Figure 9.8 where

$$\mathbf{L}(\underline{y}_i) = \mathbf{R}(\underline{x}_i) \qquad \mathbf{R}(\underline{y}_i) = \mathbf{L}(\underline{x}_i) + \mathbf{F}[\mathbf{R}(\underline{x}_i)], \quad i = 1, 2$$

$$\Delta\underline{x} = \underline{x}_1 + \underline{x}_2 \qquad \Delta\underline{y} = \underline{y}_1 + \underline{y}_2$$

$$\mathbf{L}(\Delta\underline{x}) = \mathbf{L}(\underline{x}_1) + \mathbf{L}(\underline{x}_2) \qquad \mathbf{R}(\Delta\underline{x}) = \mathbf{R}(\underline{x}_1) + \mathbf{R}(\underline{x}_2)$$

$$\mathbf{L}(\Delta\underline{y}) = \mathbf{L}(\underline{y}_1) + \mathbf{L}(\underline{y}_2) \qquad \mathbf{R}(\Delta\underline{y}) = \mathbf{R}(\underline{y}_1) + \mathbf{R}(\underline{y}_2).$$

The inputs to the one-round DES characteristic are

- The XOR $\Delta\underline{x}$ of plaintext $\underline{x}_1$ and $\underline{x}_2$ and
- The XOR $\Delta\underline{y}$ of the ciphertext $\underline{y}_1 = \text{DES}_{\underline{k}}\{\underline{x}_1\}$ and $\underline{y}_2 = \text{DES}_{\underline{k}}\{\underline{x}_2\}$.

The probability of the one-round DES characteristic is the conditional probability

$$\Pr\{\underline{x}_1, \underline{x}_2, \underline{y}_1, \underline{y}_2 / \Delta\underline{x}, \Delta\underline{y}\}$$

computed assuming a uniform distribution on plaintext and key.
Note that the difference $\mathbf{R}(\Delta\underline{y})$ depends on

- The key,
- The plaintext $(\mathbf{R}(\underline{x}_1), \mathbf{R}(\underline{x}_2))$ and
- The plaintext difference $\mathbf{L}(\Delta\underline{x})$.

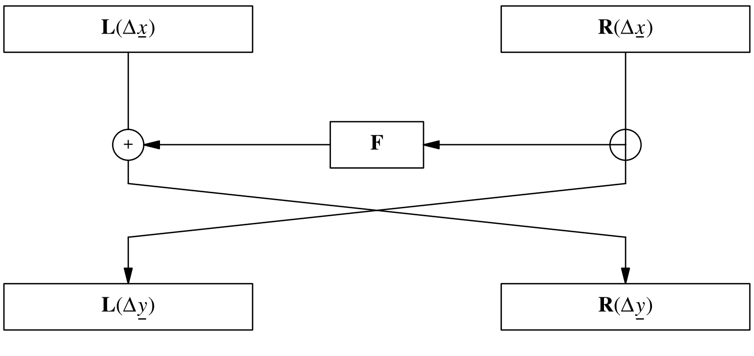


Figure 9.8 One-round DES generic characteristic.

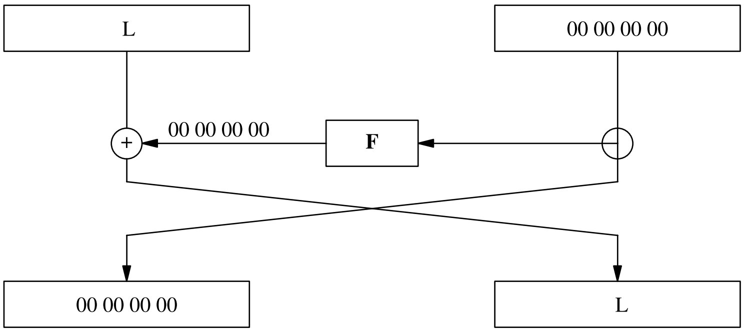


Figure 9.9 A one-round DES characteristic of probability 1.

Differential cryptanalysis infers the key by computing the probability of a specified pair of XORs ($\Delta \underline{x}$, $\Delta \underline{y}$) assuming the undetermined variables are chosen independently with the distribution.

Example 9.4

A one-round DES characteristic of probability 1 is shown in Figure 9.9. Table 9.20 shows that if the input XOR is c, then the output XOR is E for 14 of the 64 possible keys.

Example 9.5

A one-round DES characteristic of probability 14/64 is shown in Figure 9.10. By combining the one-round characteristics in *Examples 9.4* and *9.5*, we obtain *Example 9.6*.

Example 9.6

A two-round DES characteristic of probability 14/64 is shown in Figure 9.11.

Example 9.7

A three-round DES characteristic of probability $(14/64)^2$ is shown in Figure 9.12. This is as far as we will go in the exposition. The complete details are to be found in Bilham and Shamir [1993]. Differential cryptanalysis would offer a significant improvement over exhaustive key search for DES if there were fewer than 16 rounds. With 16 rounds, a time complexity of 2^{37} uses 2^{36} plain/ciphertext pairs pruned from larger pool of 2^{47} pairs. Nevertheless, differential cryptanalysis is the first and only attack on DES with complexity less than 2^{55}.

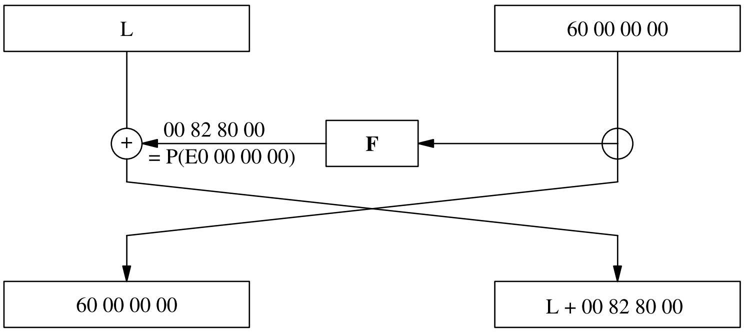


Figure 9.10 A one-round DES characteristic of probability 14/64.

Differential cryptanalysis has been successfully applied against other cryptosystems [Bilham and Shamir, 1991, 1992].

9.11 THE EFS DES-CRACKER

IBM's submitted DES in response to the National Bureau of Standards request in the *Federal Register* of August 27, 1974, for a national data encryption standard. After the publication in DES in March 1975, two workshops on DES were organized, the second to review the cryptanalysis effort on DES. There were three contentious areas:

1. Did DES contain any hidden *trap doors* whose knowledge might permit the decipherment of DES ciphertext without the key?
2. What design principles were used in DES?
3. Why was the key length chosen to be 56 bits?

Very few answers were forthcoming. IBM does business throughout the world and feels itself required to abide by the wishes of the U.S. Government. In any event $2^{56} = 72{,}057{,}594{,}037{,}927{,}936$ seemed like to large a number of key trial and the cost of building a machine required to perform key trial seemed to make the possibility remote.

A practical architecture for a *DES-cracker* with custom chips was proposed in 1993 by Michael Wiener of Bell Northern Research [Wiener, 1993]. The *Electronic Frontier Foundation* (EFF) founded in 1990 is a nonprofit public-interest group of "passionate people lawyers, technologists, volunteers, and visionaries working to protect your digital rights." The EFF seeks to educate individuals, organizations, companies, and governments about the issues that arise when computer and communications technologies change. The EFF sponsored the design and assembling of a DES-cracker [EFF, 1998].

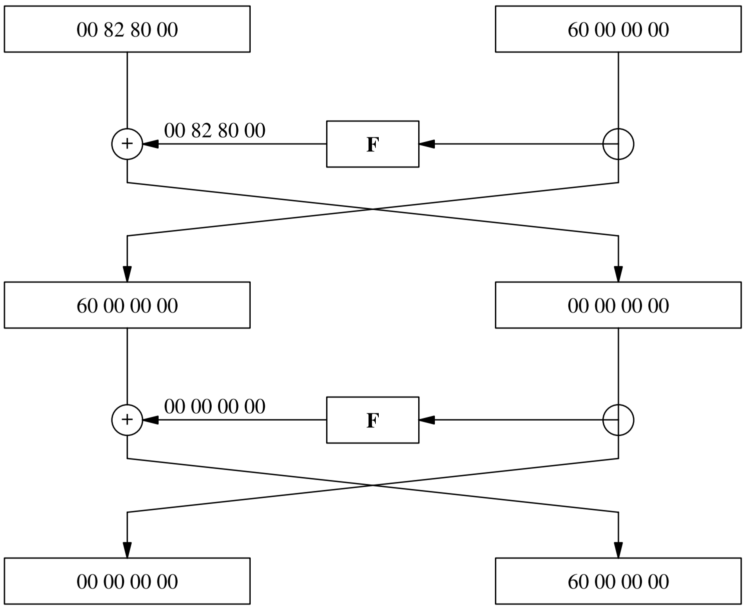


Figure 9.11 A two-round DES characteristic of probability 14/64.

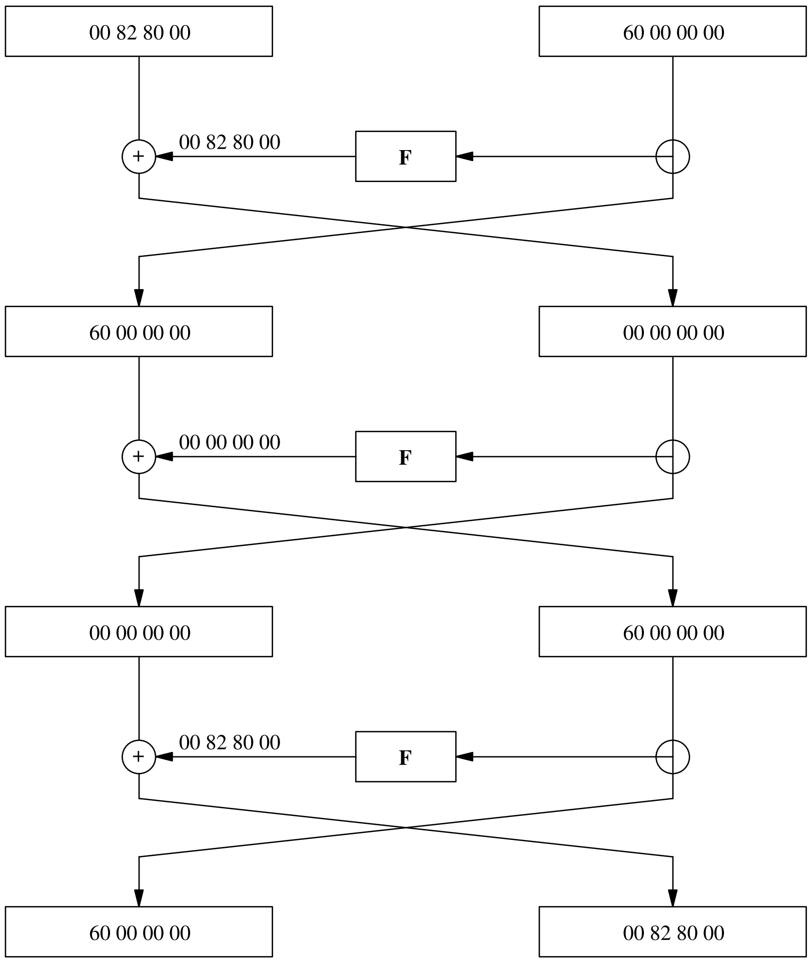


Figure 9.12 A three-round DES characteristic of probability $(14/64)^2$.

9.11.1 The Architecture

The basic component of the DES-cracker is the *search unit*, which has hardware including two 64-bit ciphertext registers and a 56-bit key register. The DES-cracker contains

1. 24 search units contained within a *custom chip*;
2. 64 customer chips mounted on a *board*;
3. 64 boards in each *chassis*; and
4. two chassis.

9.11.2 Key Search Algorithm

The ciphertext registers contain two 64-bit ciphertext blocks

$$\underline{y}_1 = \text{DES}_{\underline{k}^{(*)}}\{\underline{x}_1\} \qquad \underline{y}_2 = \text{DES}_{\underline{k}^{(*)}}\{\underline{x}_2\},$$

whose plaintexts $\underline{x}_1$, $\underline{x}_2$ and key $\underline{k}^{(*)}$ are unknown.

The DES-cracker tests a key $\underline{k}$ by deciphering $\underline{y}_1$ and $\underline{y}_2$ with it and deciding if the resulting plaintext is determining if each of the 8 bytes is contained in the table INTER of *interesting* (bytes). If it is only known that the plaintext consists only of alphanumeric text, the INTER consists of the EBCIDIC (8-bit ASCII) coding of the following 69 characters:

1. The alphabetic characters `a b ⋯ z A B ⋯ Z`
2. The digits `0 1 ⋯ 9`
3. Blank space and nine punctuation symbols `. , ? ; : ( ) ] [.`

9.11.3 Testing a Key

A key $\underline{k}$ is tested in two steps as follows.

S1. The first 64-bit ciphertext block $\underline{y}_1$ is deciphered $\mathrm{DES}_{\underline{k}}^{-1}\{\underline{y}_1\}$ and checked to see if all of its 8 bytes are interesting:

(a) If the plaintext block is *not* interesting, a new key is tested;

(b) If the plaintext block is interesting, then S2.

S2. The second 64-bit ciphertext block $\underline{y}_2$ is deciphered $\mathrm{DES}_{\underline{k}}^{-1}\{\underline{y}_2\}$ and checked to see if all of its 8 bytes are interesting:

(a) If the plaintext block is *not* interesting, a new key is tested.

The probability that a random 8-bit (0,1)-block of ciphertext will yield an interesting byte upon decipherment is

$$\frac{69}{256} \approx \frac{1}{4}.$$

The probability that a random 64-bit ciphertexts will yield 8 interesting bytes upon decipherment is approximately

$$\frac{1}{4^8} = \frac{1}{2^{16}}.$$

The probability that two random 64-bit ciphertexts will yield 16 interesting bytes upon decipherment is approximately

$$\frac{1}{4^{16}} = \frac{1}{2^{32}}.$$

If we assume that only one of the 2^{56} keys will give the true plaintext; the number of keys that will pass both steps S1 and S2 is about 2^{40}. These keys will require further testing using additional ciphertext.

A search unit performs one decipherment in 16 clock cycles. Since the clock runs at 40 MHz (40 million cycles/second) a search unit can test 2.5 million keys/second. A board therefore tests 4.8 billion keys/second and the DES-cracker tests 92,160,000,000 keys/second. On the average only half of the $2^{56} = 72{,}057{,}594{,}037{,}927{,}936$ keys need to be tested before a match is discovered.

The cost to build of the DES-cracker was \$220,000. Its proud parents announced on July 17, 1998, that it had found a DES key in 3 days.

9.12 WHAT NOW?

If the key length were 2^{64} it would have taken the DES-cracker 768 days; if the key length were 2×56, the DES-cracker would have to work a *very* long time to find the key. This points out the power of exponentiation and the advantage enjoyed by the designer of a cryptosystem over the cryptanalyst. Adding one bit to the key doubles the time for exhaustive search. If the designers of DES had been careless and there was some intrinsic weakness, or a trap-door, such a statement would not necessarily be true.

Walter Tuchman of IBM's Kingston Facility was a designer and implementor of DES. He also proposed *triple DES* [FIPS PUB 46-3, 1999] defined by[2]

$$\text{DES3}: \underline{x} \to \underline{y} = \text{DES}_{\underline{k}_1}\{\text{DES}_{\underline{k}_2}^{-1}\{\text{DES}_{\underline{k}_3}\{\underline{x}\}\}\}.$$

If $\underline{k}_1 = \underline{k}_2$, DES3 reduces to ordinary DES.

The U.S. Munitions List is part of the secondary regulations (the International Traffic in Arms Regulations or ITAR) that defines which defence articles and services are subject to licensing. Cryptographic products are included in the products (Category XIII – Auxiliary Military Equipment) regulated by ITAR.

Current export rules do not permit the export of DES3 to certain countries. An article in the *Wall Street Journal* (September 17, 1998) entitled "Encryption Export Rules Relaxed" claims that the current 56-bit limitation will be relaxed, asserting

U.S. vendors also won more freedom to export network-encryption products used primarily by Internet-service provides and communication carriers.

In "Draft Encryption Export Regulations" (dated November 23, 1999) changes in the rules were proposed. Included are:

1. Encryption commodities, software and technology for U.S. subsidiaries. You may export and re-export any encryption item of any key length under ECCNs 5A002, 5D002, and 5E002 to foreign subsidiaries of U.S. firms (as defined in part 772).[3] This includes source code and technology for internal company proprietary use, including the development of new products. U.S. firms may also transfer encryption technology (5E002) to a foreign national in the United States (except foreign nationals from Cuba, Iran, Iraq, Libya, North Korea, Sudan, and Syria) for internal company proprietary use, including the development of new products. All items developed with U.S. encryption commodities, software, and technology are subject to the EAR.
2. Encryption commodities and software. You may export and re-export any encryption commodities and software including components of any key length under ECCNs 5A002 and 5D002 to individuals, commercial firms, and other nongovernment endusers.

Export controls were transferred from the Department of Commerce to the State Department and a new policy was announced on December 9, 2004. It provides for a review for cryptographic products with key length larger than 64 bits. Details can be found at www.bis.doc.gov/encryption/default.htm.

[2]FIPS PUB 46-3, October 25, 1999, specifies what I refer to as DES3. It is also described in ANSI X9.52-1998, "Triple Data Encryption Algorithm Modes of Operation".

[3]ECCN is the the Export Control Classification Number.

9.13 THE FUTURE ADVANCED DATA ENCRYPTION STANDARD

DES was first approved as FIPS Standard 46-1 in 1977. It has been (reluctantly) reaffirmed as a standard several times, most recently in 1993, and then *only* until December 1998. At that time, the affirmation included the statement

At the next review (1998), the algorithm specified in this standard will be over twenty years old. NIST will consider alternatives which offer a higher level of security. One of these alternatives, may be proposed as a replacement standard at the 1998 review.

The National Institute of Standards (NIST) solicited proposals in the *Federal Register* (January 1, 1997) for an *Advanced Encryption Standard* (AES). The rules included

R1. AES shall be publicly defined.

R2. AES shall be a symmetric block cipher.

R3. AES shall be designed so that its key length may be increased as needed.

R4. AES shall be implementable in both hardware and software.

R5. AES shall either be
  (a) freely available, or
  (b) available under terms consistent with the ANSI Patent Policy.

R6. Algorithms which meet the above requirements will be judged based on the following factors:
  (a) security (resistance to cryptanalysis),
  (b) computational efficiency,
  (c) memory requirements,
  (d) hardware and software suitability,
  (e) simplicity,
  (f) flexibility, and
  (g) licensing requirements.

A subsequent announcement in the *Federal Register* (September 12, 1997) specified the (key, block) sizes to be supported by the AES; (128, 128) (192, 128) (256, 128). The statistical tests to be applied to evaluate the strength of the AES standard are described in Chapter 5 and specified in [FIPS, 1994, FIPS 140-1]. The selection process has involved two rounds; 15 submissions were made in Round 1. Of these, five survived in Round 2.

9.14 AND THE WINNER IS!

Rijndael was announced as the winning algorithm in October 2000 [Daemen and Rijmen, 1999] and is specified in [FIPS, 2001, FIPS-197]. Susan Landau [2004] wrote

Daemen and Rijmen sought simiplicity – simplicity of specification and simplicity of analysis. Not every cryptographer sees simplicity as an important goal – two AES finalists, MARS and Twofish, have far more complex designs. Some observers felt that this complexity was part of the reason the two algorithms were not chosen as the Advanced Encryption Standard, as their round functions were simply too difficult to analyze fully.

Too difficult to analyze! Indeed!

Rijndael is a block cipher supporting a variety of plaintext block sizes and cipher key lengths. The cipher key k is an array of dimension $4 \times Nk$ (a total of Nk 4-byte *words*)

$$\underline{k} = \begin{pmatrix} k_{0,0} & k_{0,1} & \cdots & k_{0,Nk-1} \\ k_{1,0} & k_{1,1} & \cdots & k_{1,Nk-1} \\ k_{2,0} & k_{2,1} & \cdots & k_{2,Nk-1} \\ k_{3,0} & k_{3,1} & \cdots & k_{3,Nk-1} \end{pmatrix}.$$

Each $\{k_{i,j}\}$ is regarded as both

- An 8-bit byte, that is, an element in the set $\mathcal{Z}_{2,8}$, and
- An integer in the set $\mathcal{Z}_{256}$.

Rijndael supports the Nk values of 4, 6, and 8 words (128, 192, and 256 bits).

The cipher key $\underline{k}$ is read into and from the array by columns from left to right

$$\underline{k} = (k_{0,1}, k_{1,0}, k_{2,0}, k_{3,0}, \ldots, k_{0,Nk-1}, k_{1,Nk-1}, k_{2,Nk-1}, k_{3,Nk-1}).$$

Plaintext $\underline{x}$ is an array of dimension $4 \times Nb$ (a total of Nb words)

$$\underline{x} = \begin{pmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,Nb-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,Nb-1} \\ x_{2,0} & x_{2,1} & \cdots & x_{2,Nb-1} \\ x_{3,0} & x_{3,1} & \cdots & x_{3,Nb-1} \end{pmatrix}.$$

Each $\{x_{i,j}\}$ is regarded as both

- An 8-bit byte, that is, an element in the set $\mathcal{Z}_{2,8}$, and
- An integer in the set $\mathcal{Z}_{256}$.

Rijndael supports the Nb values of 4, 6, and 8 words (128, 192, and 256 bits).

The plaintext x is read into and from the array by columns from left to right

$$\underline{x} = (x_{0,1}, x_{1,0}, x_{2,0}, x_{3,0}, \ldots, x_{0,Nb-1}, x_{1,Nb-1}, x_{2,Nb-1}, x_{3,Nb-1})$$

A Rijndael *state* $\omega = (\omega_{i,j,})$ is an array

$$\underline{\omega} = \begin{pmatrix} \omega_{0,0} & \omega_{0,1} & \cdots & \omega_{0,Nb-1} \\ \omega_{1,0} & \omega_{1,1} & \cdots & \omega_{1,Nb-1} \\ \omega_{2,0} & \omega_{2,1} & \cdots & \omega_{2,Nb-1} \\ \omega_{3,0} & \omega_{3,1} & \cdots & \omega_{3,Nb-1} \end{pmatrix}$$

of dimension $4 \times Nb$ whose entries are integers in $\mathcal{Z}_{256}$.

Like DES, the Rijndael encipherment process is the composition of transformations on the state, also referred to as *rounds* by Rijndael:

$$RIJ(\underline{x}) = \underline{y} = (T_{Nr} * T_{Nr-1} * \ldots * T_1 * T_0)(\underline{x}).$$

where the * (asterisk) denotes composition of mappings.

The number of rounds Nr depends on the values of Nb and Nk as shown in Table 9.23. The domain and range of a round T_i is a state $\underline{\omega}$ with data type `array [0..3,0..Nb]` of $\mathcal{Z}_{256}$.

The initial round T_0 is an exclusive XOR of $4Nb$ bytes of *round key* (R-key) to the state (plaintext). As in DES, subsequent rounds modify the state ω as a result of several transformations, referred to by Rijndael as *layers*:

L1. Linear Mixing Layer – the transformations `ShiftRow` and `MixColumn`;

L2. Nonlinear Layer – the transformation `ByteSub`;

L3. Key Addition Layer – the transformation `AddRoundKey`.

In order to simplify the decipherment process, DES employed a *Fiestel structure*, each round only modifying part of the data.

As π_{T_i} and θ are involutions in the Feistel structure, the inverse of the transformation $\theta * \pi_{T_i}$ on 64-bit blocks is $\pi_{T_i} * \theta$. The Feistel structure was introduced to simplify computation of the inverse transformation.

Rijndael does not follow this paradigm; each round modifies all of the bits in the data. The inverse to Rijndael is the composition

$$RIJ^{-1}(\underline{y}) = \underline{x} = (T_0^{-1} * T_1^{-1} * \cdots * T_{Nr-1}^{1} * T_{Nr}^{-1})(\underline{y})$$

of the necessarily invertible round transformations $\{T_i\}$.

9.15 THE RIJNDAEL OPERATIONS

Rijndael uses a second interpretation for the components in a byte $\underline{x} = (x_0, x_1, \ldots, x_6, x_7) \in \mathcal{Z}_{2,8}$, namely, as the coefficients of a polynomial of degree 7

$$x(\zeta) \equiv x_0\zeta^7 + x_1\zeta^6 + \cdots + x_6\zeta + x_0 \leftrightarrow \underline{x} = (x_0, x_1, \ldots, x_6, x_7).$$

The addition of bytes $\underline{x} + \underline{y}$ is according to the usual rules for the addition of polynomials, Rijndael refers to addition as EXOR rather than XOR.

Associating a byte with a polynomial provides a way to define the multiplication; if

$$x(\zeta) \equiv x_0\zeta^7 + x_1\zeta^6 + \cdots + x_6\zeta + x_7 \leftrightarrow \underline{x} = (x_0, x_1, \ldots, x_6, x_7)$$
$$y(\zeta) \equiv y_0\zeta^7 + y_1\zeta^6 + \cdots + y_6\zeta + y_7 \leftrightarrow \underline{y} = (y_0, y_1, \ldots, y_6, y_7)$$

then

$$z(\zeta) \equiv z_0\zeta^7 + z_1\zeta^6 + \cdots + z_6\zeta + z_7 \leftrightarrow \underline{z} = (z_0, z_1, \ldots, z_6, z_7) \equiv x \cdot y,$$

TABLE 9.23 Number of Rijndael Rounds *Nr*

	$Nb = 4$	$Nb = 6$	$Nb = 8$
$Nk = 4$	10	12	14
$Nk = 6$	12	12	14
$Nk = 8$	14	14	14

where

$$z(\zeta) = x(\zeta)\, y(\zeta)(\text{modulo } m(\zeta))$$

and

$$m(\zeta) = 1 + \zeta + \zeta^3 + \zeta^4 + \zeta^8$$

where $m(\zeta)$ is a primitive (see Table 8.3) but *not* irreducible polynomial.

For fixed $x = (x_1, x_1, \ldots, x_6, x_7) \in \mathcal{Z}_{2,8}$, the transformation

$$z(\zeta) = x(\zeta)\, y(\zeta)(\text{modulo } m(\zeta)) \tag{9.1}$$

with

$$\underline{y} = (y_0, y_1, \ldots, y_6, y_7) \neq (0)_8$$

is a transformation on $\mathcal{Z}_{256} - \{0\}$.

Proposition 9.3: The transformation in Equation (9.1) is invertible; given $\underline{w} = (w_0, w_1, \ldots, w_6, w_7) \neq (0)_8$, there exists a unique $\underline{y} = (y_0, y_1, \ldots, y_6, y_7) \neq (0)_8$ such that

$$w(\zeta) = x(\zeta)\, y(\zeta)\ (\text{modulo } m(\zeta)).$$

Proof: If $y_1(\zeta)$ and $y_2(\zeta)$ satisfy

$$x(\zeta) = y_1(\zeta)\ (\text{modulo } m(\zeta)) = x(\zeta)\ y_2(\zeta)\ (\text{modulo } m(\zeta))$$

then

$$0 = x(\zeta) = (y_1(\zeta) + y_2(\zeta))\ (\text{modulo } m(\zeta))$$

which contradicts the irreducibility of $m(\zeta)$ unless $y_1(\zeta) = y_2(\zeta)$.

It follows that $y(\zeta) \rightarrow x(\zeta)\ y(\zeta)$ (modulo $m(\zeta)$) is a 1-to-1 mapping on $\mathcal{Z}_{256} - \{0\}$ for each fixed x.

Proposition 9.3 implies that for each $x \neq (0)_8$, there must be a unique byte x^{-1} such that

$$x \cdot x^{-1} = (1, (0)_7)$$

or equivalently, for each polynomial $x(\zeta) \neq 0$, there exists a polynomial $x^{-1}(\zeta)$ such that

$$x(\zeta)x^{-1}(\zeta) = 1\ (\text{modulo } m(\zeta)).$$

The computation of the (multiplicative) inverse of $\underline{x}$ uses the extended Euclidean algorithm, which we will now describe.

Using the notation in Chapter 8,

- The polynomial $r(\zeta)$ in $\mathcal{P}[z]$ is a *divisor* of polynomials $p(\zeta)$ and $p(\zeta)$ *in* $\mathcal{P}[z]$ if $r(\zeta)$ is a factor of both polynomials;
- $r(\zeta)$ is the *greatest common divisor* of $p(\zeta)$ and $q(\zeta)$ if it is a divisor and has the maximum degree of all common divisors.

$\gcd\{p(\zeta), q(\zeta)\}$ denotes the greatest common divisor of $p(\zeta)$ and $q(\zeta)$.

Proposition 9.4 (Extended Euclidean Algorithm for Polynomials with Coefficients in $\mathcal{Z}_2$):

9.4a If $p(\zeta)$ and $q(\zeta)$ are polynomials in $\mathcal{P}[z]$, the sequence of remainders $\{r_j(\zeta) : j \geq 2\}$

$$\begin{aligned}
&r_0(\zeta) = p(\zeta) \\
&r_1(\zeta) = q(\zeta) \\
&r_0(\zeta) = c_1(\zeta)r_1(\zeta) + r_2(\zeta); && 0 \leq \deg(r_2) < \deg(r_1) \\
&r_1(\zeta) = c_2(\zeta)r_2(\zeta) + r_3(\zeta); && 0 \leq \deg(r_3) < \deg(r_2) \\
&\vdots \qquad\qquad \vdots && \vdots \\
&r_{s-2}(\zeta) = c_{s-1}(\zeta)r_{s-1}(\zeta) + r_s(\zeta); && 0 \leq \deg(r_s) < \deg(r_{s-1}) \\
&r_{s-1}(\zeta) = c_s(\zeta)r_s(\zeta) + r_{s+1}(\zeta); && 0 \leq \deg(r_{s+1}) < \deg(r_s)
\end{aligned}$$

is ultimately identically 0.

9.4b If s is the first index for which $r_{s+1}(\zeta) = 0$, then $r_s(\zeta) = \gcd\{p(\zeta), q(\zeta)\}$.

9.4c If $\deg(p) > \deg(q)$, the time to compute $\gcd\{p(\zeta), q(\zeta)\}$ is $O\ ((\log_2 \deg(p))^3)$.

Example 9.8

$$\begin{aligned}
&p(\zeta) = 1 + \zeta^4 + \zeta^5 + \zeta^6 + \zeta^8 + \zeta^9 + \zeta^{10} \\
&q(\zeta) = 1 + \zeta^2 + \zeta^3 + \zeta^5 + \zeta^6 + \zeta^9 \\
&r_0(\zeta) = p(\zeta) \\
&r_1(\zeta) = q(\zeta) \\
&r_0(\zeta) = (1 + z)r_1(\zeta) + r_2(\zeta) && r_2(\zeta) = z + z^2 + z^6 + z^7 + z^8 \\
&r_1(\zeta) = (z + 1)r_2(\zeta) + r_3(\zeta) && r_3(\zeta) = 1 + z + z^2 + z^5 \\
&r_2(\zeta) = (z^3 + z^2 + z + 1)r_3(\zeta) + r_4(\zeta) && r_4(\zeta) = 1 + z + z^3 \\
&r_3(\zeta) = (z^2 + 1)r_4(\zeta) + r_5(\zeta) && r_5(\zeta) = 0 \\
&\gcd\{p(\zeta), q(\zeta)\} = 1 + z + z^3
\end{aligned}$$

The Operations `ByteSub` *and* `InvByteSub` are defined first for bytes $\underline{x}$ as follows:

$$\mathrm{BS}_1(\underline{x}) = \underline{z} \equiv \begin{cases} 0, & \text{if } \underline{x} = (0)_8 \\ \underline{x}^{-1}, & \text{if } \underline{x} \neq (0)_8 \end{cases}$$

$$BS_2(\underline{z}) = A\underline{z} + B$$

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad B = (1, 1, 0, 0, 1, 1, 0)$$

$$BS(\underline{x}) = BS_2(BS_1(\underline{x}))$$

Remarks:

1. $\mathrm{BS}_1^{-1} = \mathrm{BS}_1$.
2. A simple computation shows that the transpose A^t of A is equal to A^{-1} so that $\mathrm{BS}_2^{-1} = \mathrm{BS}_2$.

The operation `ByteSub` is defined for a state

$$\underline{\omega} = \begin{pmatrix} \omega_{0,0} & \omega_{0,1} & \cdots & \omega_{0,Nb-1} \\ \omega_{1,0} & \omega_{1,1} & \cdots & \omega_{1,Nb-1} \\ \omega_{2,0} & \omega_{2,1} & \cdots & \omega_{2,Nb-1} \\ \omega_{3,0} & \omega_{3,1} & \cdots & \omega_{3,Nb-1} \end{pmatrix}$$

by

$$BS(\underline{\omega}) = \begin{pmatrix} \mathrm{BS}(\omega_{0,0}) & \mathrm{BS}(\omega_{0,1}) & \cdots & \mathrm{BS}(\omega_{0,Nb-1}) \\ \mathrm{BS}(\omega_{1,0}) & \mathrm{BS}(\omega_{1,1}) & \cdots & \mathrm{BS}(\omega_{1,Nb-1}) \\ \mathrm{BS}(\omega_{2,0}) & \mathrm{BS}(\omega_{2,1}) & \cdots & \mathrm{BS}(\omega_{2,Nb-1}) \\ \mathrm{BS}(\omega_{3,0}) & \mathrm{BS}(\omega_{3,1}) & \cdots & \mathrm{BS}(\omega_{3,Nb-1}) \end{pmatrix}.$$

`ByteSub` plays the role of the S-box in DES and is the *only* nonlinear element in Rijndael.

The Operations `ShiftRow` *and* `InvShiftRow` are cyclic left and right shifts of the rows of a state $\underline{\omega}$. SR cyclically left-shifts row $\underline{i}$ of $\underline{\omega}$ by C_i bytes as listed in Table 9.24. For example, when $Nb = 4$

$$\mathrm{SR}: \underline{\omega} = \begin{pmatrix} \omega_{0,0} & \omega_{0,1} & \omega_{0,2} & \omega_{0,3} \\ \omega_{1,0} & \omega_{1,1} & \omega_{1,2} & \omega_{1,3} \\ \omega_{2,0} & \omega_{2,1} & \omega_{2,2} & \omega_{2,3} \\ \omega_{3,0} & \omega_{3,1} & \omega_{3,2} & \omega_{3,3} \end{pmatrix} \to \begin{pmatrix} \omega_{0,0} & \omega_{0,1} & \omega_{0,2} & \omega_{0,3} \\ \omega_{1,1} & \omega_{1,2} & \omega_{1,3} & \omega_{1,0} \\ \omega_{2,2} & \omega_{2,3} & \omega_{2,0} & \omega_{2,1} \\ \omega_{3,3} & \omega_{3,0} & \omega_{3,1} & \omega_{3,2} \end{pmatrix}.$$

The inverse `InvShiftRow` is a cyclic right shift of the row i of a state $\underline{\omega}$ by C_i bytes.

TABLE 9.24 Rijndael Row Shift Parameters

Nb	C_0	C_1	C_2	C_3
4	0	1	2	3
6	0	1	2	3
8	0	1	3	4

The Operations `MixColumn` *and* `InvMixColumn` are defined in terms of multiplication of polynomials whose coefficients are bytes. We write $\underline{x} = \langle\texttt{ab}\rangle$ to show that the byte $\underline{x}$ is composed of the two hexadecimal digits `a` and `b`. Table 9.25 shows a coding between $\underline{x}$ and $\langle\texttt{ab}\rangle$. To compute the product

$$c(\zeta) = a(\zeta) \times b(\zeta) \text{ (modulo } (1 + \zeta^4)) \tag{9.2}$$

$$c(\zeta) = c_3\zeta^3 + c_2\zeta^2 + c_1\zeta + c_0$$

with

$$a(\zeta) = a_3\zeta^3 + a_2\zeta^2 + a_1\zeta + a_0$$
$$b(\zeta) = b_3\zeta^3 + b_2\zeta^2 + b_1\zeta + b_0$$

the sum of the products of the coefficient of z^i in $a(\zeta)$ and the coefficient of z^j in $b(\zeta)$ with $i + j = k$ (modulo 4) for fixed k with $k = 0, 1, 2, 3$ is computed. This may be written as

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{9.3}$$

$$\begin{aligned} c_0 &= (a_0 \cdot b_0) + (a_3 \cdot b_1) + (a_2 \cdot b_2) + (a_1 \cdot b_3) \\ c_1 &= (a_1 \cdot b_0) + (a_0 \cdot b_1) + (a_3 \cdot b_2) + (a_2 \cdot b_2) \\ c_2 &= (a_2 \cdot b_0) + (a_1 \cdot b_1) + (a_0 \cdot b_2) + (a_3 \cdot b_3) \\ c_3 &= (a_3 \cdot b_0) + (a_2 \cdot b_1) + (a_1 \cdot b_2) + (a_0 \cdot b_3) \end{aligned} \tag{9.4}$$

Example 9.9

We compute $c(\zeta) = a(\zeta) \times b(\zeta)$ (modulo $(1 + \zeta^4)$) with

$$a(\zeta) = \langle\texttt{02}\rangle + \langle\texttt{01}\rangle\zeta + \langle\texttt{01}\rangle\zeta^2 + \langle\texttt{03}\rangle\zeta^3$$
$$b(\zeta) = \langle\texttt{0E}\rangle + \langle\texttt{09}\rangle\zeta + \langle\texttt{0D}\rangle\zeta^2 + \langle\texttt{0B}\rangle\zeta^3$$

TABLE 9.25 Byte-to-Hex Coding Table

Bits	Hex	Bits	Hex	Bits	Hex	Bits	Hex
0000	0	0100	4	1000	8	1100	C
0001	1	0101	5	1001	9	1101	D
0010	2	0110	6	1010	A	1110	E
0011	3	0111	7	1011	B	1111	F

by adding the products of the coefficient of z^i in a (ζ) and the coefficient of z^j in $b(\zeta)$ with $i + j = k$ (modulo 4) for fixed k with $k = 0, 1, 2, 3$.

1. The coefficient of ζ^0 in $c(\zeta)$ is the sum of the products of
   (a) the coefficient of ζ^i in $a(\zeta)$ and
   (b) the coefficient of ζ^j in $b(\zeta)$ with $i + j = 0$ (modulo 4); that is,

$$\begin{aligned}
\langle\texttt{02}\rangle \cdot \langle\texttt{0E}\rangle &\leftrightarrow \zeta(\zeta + \zeta^2 + \zeta^3) = \zeta^2 + \zeta^3 + \zeta^4 \\
\langle\texttt{01}\rangle \cdot \langle\texttt{0B}\rangle &\leftrightarrow 1(1 + \zeta + \zeta^3) = 1 + \zeta + \zeta^3 \\
\langle\texttt{01}\rangle \cdot \langle\texttt{D, 0}\rangle &\leftrightarrow 1(1 + \zeta^2 + \zeta^3) = 1 + \zeta^2 + \zeta^3 \\
\langle\texttt{03}\rangle \cdot \langle\texttt{09}\rangle &\leftrightarrow (1 + \zeta)(1 + \zeta^3) = 1 + \zeta + \zeta^3 + \zeta^4
\end{aligned}$$

   with value 1.

2. The coefficient of ζ^1 in $c(\zeta)$ is the sum of the products of
   (a) the coefficient of ζ^i in $a(\zeta)$ and
   (b) the coefficient of ζ^j in $b(\zeta)$ with $i + j = 1$ (modulo 4); that is,

$$\begin{aligned}
\langle\texttt{02}\rangle \cdot \langle\texttt{09}\rangle &\leftrightarrow \zeta(1 + \zeta^3) = \zeta + \zeta^4 \\
\langle\texttt{01}\rangle \cdot \langle\texttt{0E}\rangle &\leftrightarrow 1(\zeta + \zeta^2 + \zeta^3) = \zeta + \zeta^2 + \zeta^3 \\
\langle\texttt{01}\rangle \cdot \langle\texttt{0B}\rangle &\leftrightarrow 1(1 + \zeta + \zeta^3) = 1 + \zeta + \zeta^3 \\
\langle\texttt{03}\rangle \cdot \langle\texttt{0D}\rangle &\leftrightarrow (1 + \zeta)(1 + \zeta^2 + \zeta^3) = 1 + \zeta + \zeta^2 + \zeta^4
\end{aligned}$$

   with value 0.

3. The coefficient of ζ^2 in $c(\zeta)$ is the sum of the products of
   (a) the coefficient of ζ^i in $a(\zeta)$ and
   (b) the coefficient of ζ^j in $b(\zeta)$ with $i + j = 2$ (modulo 4); that is,

$$\begin{aligned}
\langle\texttt{02}\rangle \cdot \langle\texttt{0D}\rangle &\leftrightarrow \zeta(1 + \zeta^2 + \zeta^3) = \zeta + \zeta^3 + \zeta^4 \\
\langle\texttt{01}\rangle \cdot \langle\texttt{09}\rangle &\leftrightarrow 1(1 + \zeta^3) = 1 + \zeta^3 \\
\langle\texttt{01}\rangle \cdot \langle\texttt{0E}\rangle &\leftrightarrow 1(\zeta + \zeta^2 + \zeta^3) = \zeta + \zeta^2 + \zeta^3 \\
\langle\texttt{03}\rangle \cdot \langle\texttt{0B}\rangle &\leftrightarrow (1 + \zeta)(1 + \zeta + \zeta^3) = 1 + \zeta^2 + \zeta^3 + \zeta^4
\end{aligned}$$

   with value 0.

4. The coefficient of ζ^3 in $c(\zeta)$ is the sum of the products of
   (a) the coefficient of ζ^i in $a(\zeta)$ and
   (b) the coefficient of ζ^j in $b(\zeta)$ with $i + j = 3$ (modulo 4); that is,

$$\langle \texttt{02}\rangle \cdot \langle \texttt{0B}\rangle \leftrightarrow \zeta(1+\zeta+\zeta^3) = \zeta+\zeta^2+\zeta^4$$
$$\langle \texttt{01}\rangle \cdot \langle \texttt{0D}\rangle \leftrightarrow 1(1+\zeta^2+\zeta^3) = 1+\zeta^2+\zeta^3$$
$$\langle \texttt{01}\rangle \cdot \langle \texttt{09}\rangle \leftrightarrow 1(1+\zeta^3) = 1+\zeta^3$$
$$\langle \texttt{03}\rangle \cdot \langle \texttt{0E}\rangle \leftrightarrow (1+\zeta)(\zeta+\zeta^2+\zeta^3) = \zeta+\zeta^4$$

with value 0.

Example 9.9 shows that

$$c(\zeta) = a(\zeta) \times b(\zeta) \text{ (modulo}(1+\zeta^4)) = 1$$

when

$$a(\zeta)=\langle \texttt{02}\rangle + \langle \texttt{01}\rangle\zeta + \langle \texttt{01}\rangle\,\zeta^2 + \langle \texttt{03}\rangle\zeta^3 \quad b(\zeta) = \langle \texttt{0E}\rangle + \langle \texttt{09}\rangle\zeta + \langle \texttt{0D}\rangle\,\zeta^2 + \langle \texttt{0B}\rangle\zeta^3.$$

This computation proves Proposition 9.5.

Proposition 9.5: If $a(\zeta) = \langle \texttt{02}\rangle + \langle \texttt{01}\rangle\zeta + \langle \texttt{01}\rangle\zeta^2 + \langle \texttt{03}\rangle\zeta^3$, then the transformation

$$T_a : b(\zeta) \to a(\zeta)\, b(\zeta) \text{ (modulo}(1+\zeta^4))$$

is invertible with inverse

$$T_a^{-1} : b(\zeta) \to a^{-1}(\zeta)\, b(\zeta) \text{ (modulo}(1+\zeta^4))$$

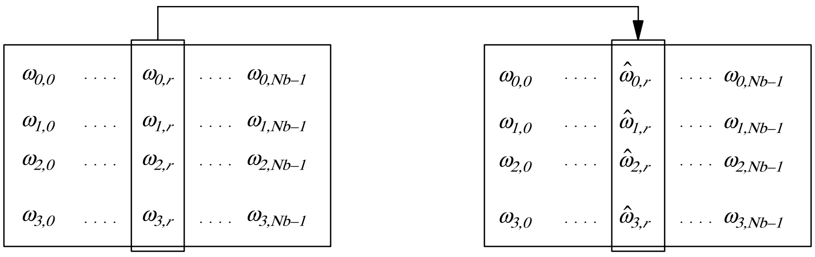
with

$$a^{-1}(\zeta) = \langle \texttt{0E}\rangle + \langle \texttt{09}\rangle\zeta + \langle \texttt{0D}\rangle\zeta^2 + \langle \texttt{0B}\rangle\zeta^3.$$

A column in the state

$$\underline{\omega} = \begin{pmatrix} \omega_{0,0} & \omega_{0,1} & \cdots & \omega_{0,Nb-1} \\ \omega_{1,0} & \omega_{1,1} & \cdots & \omega_{1,Nb-1} \\ \omega_{2,0} & \omega_{2,1} & \cdots & \omega_{2,Nb-1} \\ \omega_{3,0} & \omega_{3,1} & \cdots & \omega_{3,Nb-1} \end{pmatrix}$$

is identified with a polynomial of degree (at most) three, whose coefficients are bytes. The linear transformation `MixColumn` (MC) consists of the application of MC to each of the *Nb* columns of a state $\underline{\omega}$ (Fig. 9.13):



$\omega_{0,0}$ .... $\omega_{0,r}$ .... $\omega_{0,Nb-1}$
$\omega_{1,0}$ .... $\omega_{1,r}$ .... $\omega_{1,Nb-1}$
$\omega_{2,0}$ .... $\omega_{2,r}$ .... $\omega_{2,Nb-1}$
$\omega_{3,0}$ .... $\omega_{3,r}$ .... $\omega_{3,Nb-1}$
$\omega_{0,0}$ .... $\hat{\omega}_{0,r}$ .... $\omega_{0,Nb-1}$
$\omega_{1,0}$ .... $\hat{\omega}_{1,r}$ .... $\omega_{1,Nb-1}$
$\omega_{2,0}$ .... $\hat{\omega}_{2,r}$ .... $\omega_{2,Nb-1}$
$\omega_{3,0}$ .... $\hat{\omega}_{3,r}$ .... $\omega_{3,Nb-1}$

Figure 9.13 `MixColumn` applied to the *r*th column of the state.

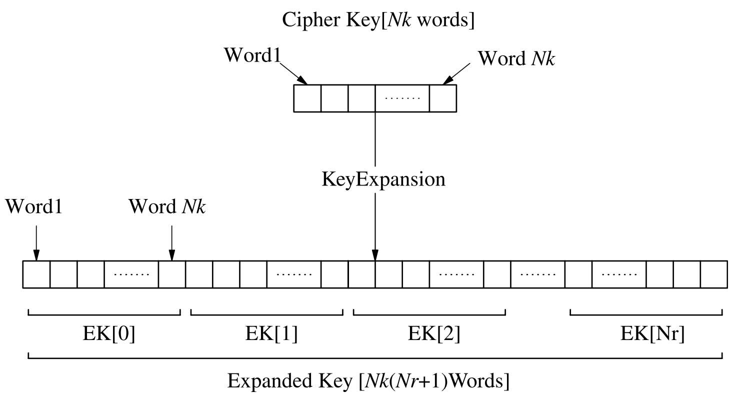


Figure 9.14 Rijndael key expansion.

$$(\hat{\omega}_{0,\,r}, \hat{\omega}_{1,\,r}, \hat{\omega}_{2,\,r}, \hat{\omega}_{3,\,r}) = \mathrm{MC}(\omega_{0,\,r}, \omega_{1,\,r}, \omega_{2,\,r}, \omega_{3,\,r})$$

$$\begin{pmatrix} \hat{\omega}_{0,r} \\ \hat{\omega}_{1,r} \\ \hat{\omega}_{2,r} \\ \hat{\omega}_{3,r} \end{pmatrix} = \begin{pmatrix} \langle 02\rangle & \langle 03\rangle & \langle 01\rangle & \langle 01\rangle \\ \langle 01\rangle & \langle 02\rangle & \langle 03\rangle & \langle 01\rangle \\ \langle 01\rangle & \langle 01\rangle & \langle 02\rangle & \langle 03\rangle \\ \langle 03\rangle & \langle 01\rangle & \langle 01\rangle & \langle 02\rangle \end{pmatrix} \begin{pmatrix} \omega_{0,r} \\ \omega_{1,r} \\ \omega_{2,r} \\ \omega_{3,r} \end{pmatrix}$$

The Operation `AddRoundKey` is the exclusive-OR of *Nb* words of R-key to a state $\underline{\omega}$. The *Nb* words of the R-key used in each round are derived from expanding the *Nk* words of cipher key into $Nb(Nr + 1)$ words of R-key (Fig. 9.14):

$$\underline{\mathrm{EK}} = (\mathrm{EK}[0], \mathrm{EK}[1], \mathrm{EK}[2], \ldots, \mathrm{EK}[Nr]).$$

The algorithm for key expansion is different for $Nk \leq 6$ and $Nk > 6$.

Key Expansion Algorithm ($Nk \leq 6$)

```
1. for $i := 0$ to $Nk - 1$

   $\mathrm{EK}[i] = (k_{0,i}, k_{1,i}, k_{2,i}, k_{3,i})$ $k_{j,i}$ is a word;
2. for $i := Nk$ to $NkNr - 1$

temp = $\mathrm{EK}[i - 1]$
   if $0 \neq (i \bmod Nk)$, then $\mathrm{EK}[i] = \mathrm{temp} + \mathrm{EK}[i - Nk]$;
   if $0 = (i \bmod Nk)$, then temp = $\mathrm{BS}(\mathrm{RB}(\mathrm{temp})) + \mathrm{R\_Con}(\lfloor i/Nk)$
```

where

- The transformation `RotByte` (RB) is the left-cyclic shift fay one byte of a word $(\omega_0, \omega_1, \omega_2, \omega_3)$

$$\mathrm{RB} : (\omega_0, \omega_1, \omega_2, \omega_3) \to (\omega_1, \omega_2, \omega_3, \omega_0).$$

- `ByteSub` (BS) is applied to each of bytes of $\mathrm{RB}(\omega_0, \omega_1, \omega_2, \omega_3)$

$$\mathrm{BS(RB)} : (\omega_0, \omega_1, \omega_2, \omega_3) \to (\mathrm{BS}(\omega_1), \mathrm{BS}(\omega_2), \mathrm{BS}(\omega_3), \mathrm{BS}(\omega_0)).$$

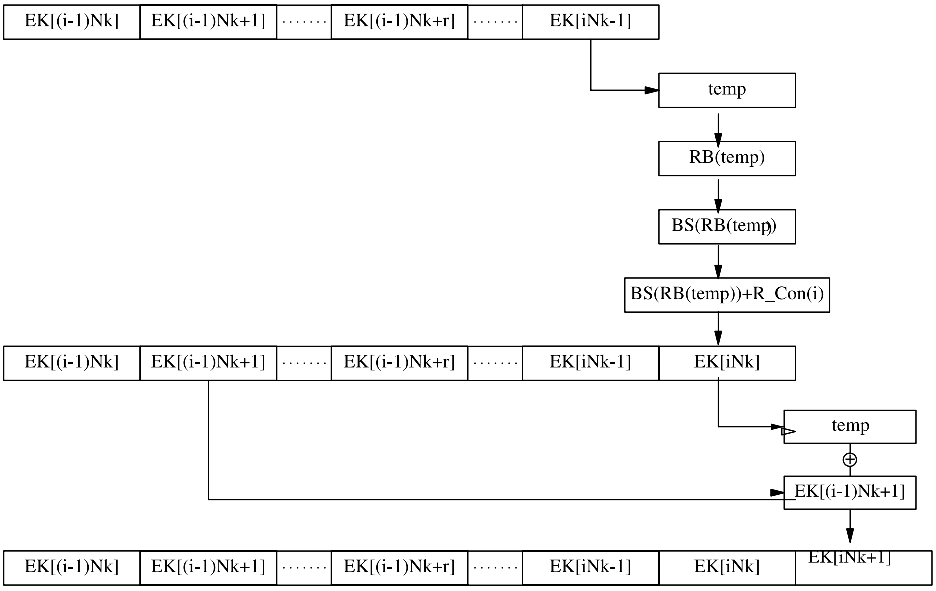


Figure 9.15 Two intermediate steps in Rijndael key expansion.

- The *round constants* {R_Con(j)} of type `array [0..3]` of $\mathcal{Z}_{256}$ are defined by

$$\begin{aligned} R\_\text{Con}(,j) &= (RC[j], \langle \texttt{00}\rangle, \langle \texttt{00}\rangle, \langle \texttt{00}\rangle) \\ RC[1] &= \langle \texttt{01}\rangle \\ RC[2] &= x = \langle \texttt{02}\rangle \\ RC[i] &= x \cdot RC[i-1] \end{aligned}$$

Key Expansion Algorithm ($Nk > 6$)

```
1. for $i := 0$ to $Nk - 1$

   EK[$i$] = $(k_{0,i}, k_{1,i}, k_{2,i}, k_{3,i})$ $k_{j,i}$ is a word;
2. for $i := Nk$ to $NkNr - 1$

   temp = EK[$i - 1$]
       if $0 \neq (i \bmod Nk)$, then EK[$i$] = temp + EK[$i - Nk$];
       if $4 = (i \bmod Nk)$, then temp = BS(temp);
```

Initial Round

`AddRoundKey`

Rounds 1-Nr

`ByteSub —→ ShiftRow —→ MixColumn —→ AddRoundKey`

Final Round

`ByteSub—→ ShiftRow —→ AddRoundKey`

Figure 9.16 The order of operations in the Rijndael Cipher.

Two intermediate steps in the Rijndael expansion for $Nk \leq 6$ are shown in Figure 9.15. Any Nk consecutive word of R-key determine the complete R-key.

9.16 THE RIJNDAEL CIPHER

The order in which the transformations `ByteSub`, `ShiftRow`, `MixColumn`, and *`AddRoundKey`* are to be applied is as shown in Figure 9.16.

9.17 RIJNDAEL'S STRENGTH: PROPAGATION OF PATTERNS

Although there is no proof that Rijndael can resist all cryptographic attacks

- The authors have tested whether several existing cryptanalytic techniques when applied to Rijndael can recover die key with a work factor less than exhaustive key trial, and
- Rijndael has been exposed to a careful scrutiny by outside cryptanalysts.

We summarize some of the unsuccessful attacks on Rijndael.

9.17.1 Differential Cryptanalysis

Define the *byte weight* of two states $\underline{\omega}_1$ and $\underline{\omega}_2$ as the number of nonzero bytes in $\underline{\omega}_1 \oplus \underline{\omega}_2$. Differential cryptanalysis has two phases:

1. A search for pairs of states $(\underline{\omega}_1, \underline{\omega}_2)$ whose byte weight does not change significantly over several rounds when the states $\underline{\omega}_i$ are enciphered with the same key, and
2. An attempt to use such pairs to infer key bits.

The Rijndael round transformation on a state

$$T : \underline{\omega} \rightarrow \texttt{AddRoundKey(MixColumn(ShiftRow(ByteSub}(\omega))))$$

is a permutation on the states in $\mathcal{Z}_{4Nb,8}$.

Nyberg [1993] and Beth and Ding [1993] introduced a measure of nonlinearity for permutations F on $\mathcal{Z}_{n,2}$ defining

$$N_F = \max_{b \in \mathcal{Z}_{n,2}} N_F(a)$$

$$N_F(a) = |\{z \in \mathcal{Z}_{n,2} : F(z+a) - F(z) = b\}|, \quad a \neq 0,$$

where $|\cdots|$ is the size of the set $\cdots$. Note that if F is a linear transformation, then $F(z+a) - F(z) = b$ has either 0 or 2 solution.

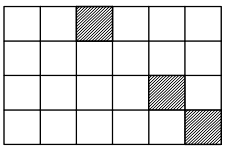


Figure 9.17 An $Nk = 6$ Rijndael Activity Pattern.

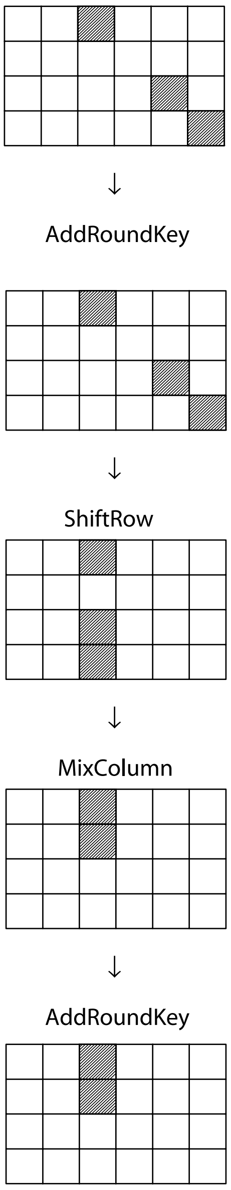


Figure 9.18 Effect of the round transformation the $Nk = 6$ Rijndael activity pattern.

Nyberg calls F *differentially δ-uniform* if $N_F \leq \delta$ and proves Proposition 9.6.

Proposition 9.6:

9.6a $N_F \leq 2$.

9.6b If F is differentially δ-uniform and A and B are linear transformations, then $A * F * B$ is differentially δ-uniform.

9.6c The permutation `ByteSub` is differentially 4-uniform.

An *active byte* in a state is a nonzero byte. An *activity pattern* is a description of the active bytes in a pair of states $(\underline{\omega}_1, \underline{\omega}_2)$.

Example 9.10

An activity pattern for $Nk = 6$ is illustrated in Figure 9.17; bytes (0,2), (2,4), and (3,5) are active. The effect of a Rijndael round transformation on an activity pattern uses the following observations:

- An activity pattern remains *un*changed under `AddRoundKey`, `ByteSub`, and `ShiftRow`;
- `MixColumn` only alters the columns containing an active byte.

A possible effect of the Rijndael round transformation on the activity pattern in *Example 9.10* is shown in Figure 9.18.

Example 9.10 shows that the number of active bytes depends on the number of *active columns*; that is, columns with an active byte.

Daemen and Rijmen define an m-round differential trail as a sequence of state-pairs

$$\begin{matrix}\underline{\omega}_1 \\ \underline{\omega}_1 + \underline{a}_1\end{matrix} \xrightarrow{T_1} \begin{matrix}\underline{\omega}_2 \\ \underline{\omega}_2 + \underline{a}_2\end{matrix} \xrightarrow{T_2} \cdots \xrightarrow{T_{m-1}} \begin{matrix}\underline{\omega}_m \\ \underline{\omega}_m + \underline{a}_m\end{matrix}$$

related by chaining

$$\begin{matrix}\underline{\omega}_i \\ \underline{\omega}_i + \underline{a}_i\end{matrix} \xrightarrow{T_1} \begin{matrix}\underline{\omega}_{i+1} \\ \underline{\omega}_{i+1} + \underline{a}_{i+1}\end{matrix}, \quad i = 1, 2, \ldots m-1.$$

The fraction of key values that are consistent for the ith segment is denoted by

$$R\left(\begin{matrix}\underline{\omega}_i \\ r\underline{\omega}_i + {}_i\end{matrix} \xrightarrow{T_1} \begin{matrix}\underline{\omega}_{i+1} \\ \underline{\omega}_{i+1} + \underline{a}_{i+1}\end{matrix}\right)$$

Daemen and Rijmen argue in Daemen [1995] and in the supplementary annex [Daemen and Rijmen, 1999a] that when the fractions of consistent keys

$$R\left(\begin{matrix}\underline{\omega}_i \\ \underline{\omega}_i + \underline{a}_i\end{matrix} \xrightarrow{T_i} \begin{matrix}\underline{\omega}_{i+1} \\ \underline{\omega}_{i+1} + \underline{a}_{i+1}\end{matrix}\right), \quad i = 1, 2, \ldots, m-1$$

are small, the keys act *independently* and the fractions may be multiplied to give

$$R\left(\begin{matrix}\underline{\omega}_1 \\ \underline{\omega}_1 + \underline{a}_1\end{matrix} \xrightarrow{T_1} \begin{matrix}\underline{\omega}_2 \\ \underline{\omega}_2 + \underline{a}_2\end{matrix} \xrightarrow{T_2} \cdots \xrightarrow{T_{m-1}} \begin{matrix}\underline{\omega}_m \\ \underline{\omega}_m + \underline{a}_m\end{matrix}\right)$$
$$\approx \prod_{i=1}^{m-1} R\left(\begin{matrix}\underline{\omega}_i \\ \underline{\omega}_i + \underline{a}_i\end{matrix} \xrightarrow{T_i} \begin{matrix}\underline{\omega}_{i+1} \\ \underline{\omega}_{i+1} + \underline{a}_{i+1}\end{matrix}\right).$$

In Daemen and Rijmen [1999b], the authors state Proposition 9.7.

Proposition 9.7

9.7a The number of active bytes after two rounds is at least 5.

9.7b The number of active bytes after four rounds is at least 25.

Combining **Proposition 9.7b** with Nyberg's result shows 2^{-150} to be the probability that a four-round differential attack will be successful.

9.18 WHEN IS A PRODUCT BLOCK-CIPHER SECURE?

In LUCIFER, DES, and Rijndael, the substitution (S-box) provides the only nonlinear element in the encipherment transformation. In the 16 years various authors have studied the general design principles of strong product block-ciphers, which have been investigated since the beginning of the 1980s. Susan Landau's paper [Landau, 2004] is a very fine summary of the concepts.

$\mathcal{Z}_{2,\,n}$ will continue to denote the set of all binary n-vectors. The *Hamming distance* $d(\underline{x}, \underline{y})$ between two n-vectors $\underline{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\underline{y} = (y_0, y_1, \ldots, y_{n-1})$ is the number of coordinates in which they differ.

If

$$\underline{0} = \underbrace{0, 0, \ldots, 0}_{n \text{ copies}} \quad \underline{1} = \underbrace{1, 1, \ldots, 1}_{n \text{ copies}}$$

$$\underline{u}_i = \begin{cases} (1, (0)_{n-1}), & \text{if } i = 0 \\ (\underbrace{0, 0, \ldots, 0}_{(i-1) \text{ terms}}, 1, \underbrace{0, 0, \ldots, 0}_{(n-i) \text{ terms}}), & \text{if } 0 < i < n-1 \\ ((0)_{n-1}, 1), & \text{if } i = n-1 \end{cases}$$

$$\underline{x} = (x'_0, x'_1, \ \ldots \ , x'_{n-1})$$

where, indicates complementation, then

$$\begin{array}{llll} n & = d(\underline{0}, \underline{1}) & 2 = d(\underline{u}_i, \underline{u}_j), & 0 \le i < j < n \\ n & = d(\underline{x}, \underline{x}') & 1 = d(\underline{0}, \underline{u}_i), & 0 \le i < n \end{array}$$

An S-box is *Boolean function*; that is, a mapping

$$f : \mathcal{Z}_{2,\,n} \to \mathcal{Z}_{2,\,m}.$$

We use the notations

- $\mathcal{B}_n$ for the set of all Boolean functions on $\mathcal{Z}_{2,n}$ with values in $\mathcal{Z}_2$,
- $\mathcal{L}_n$ for the set of all *linear Boolean functions* $f(\underline{x}) = a_0x_0 + a_1x_1 + \cdots + a_{n-1}x_{n-1}$ where the coefficient vector $\underline{a} = (a_0, a_1, \ldots, a_{n-1})$ is in $\mathcal{Z}_{2,\,n}$, and
- $\mathcal{A}_n$ for the set of all affine Boolean functions $f(\underline{x}) = b + a_0x_0 + a_1x_1 + \cdots + a_{n-1}\,x_{n-1}$ where the coefficient vector $\underline{a} = (a_0, a_1, \ldots, a_{n-1})$ is in $\mathcal{Z}_{2,\,n}$, and $b \in \mathcal{Z}_2$.

Although Feistel's paradigm

$$\mathcal{T} : (L, R) \to (F(R) + L, R)$$

does *not* require F to be invertible, some form of nonlinearity must be part of the design. Pierpryzk's paper [1990] proposed measuring the *nonlinearity* of $f \in \mathcal{B}$ by

$$\mathcal{N}(f) = d(f, \mathcal{B}_n) \equiv \min_{g \in \mathcal{L}_n} d(f, g)$$

where the Hamming distance between two functions $f(\underline{x})$ and $g(\underline{x})$ is

$$d(f, g) = \#\{\underline{x} : f(\underline{x}) \neq g(\underline{x})\}$$

and $\#\{\cdots\}$ is the cardinality of $\{\cdots\}$.

The nonlinearity $\mathcal{N}(f)$ of a permutation $\underline{f} = (f_0, f_0, \ldots, f_{n-1})$ of $\mathcal{Z}_{2,\,n}$ is

$$\mathcal{N}(f) \equiv \lim_{0 \leq i < n} \mathcal{N}(f_i).$$

Another interpretation is possible where an element $\underline{x} = (x_0, x_1, \ldots, x_{n-2}, x_{n-1})$ of $\mathcal{Z}_{2,n}$ may be interpreted as the coefficient of the polynomial of degree at most $n-1$

$$p_{\underline{x}}(z) \equiv x_{n-1} + x_{n-2}z + \cdots + x_1 z^{n-2} + x_0 z^{n-1} \Leftrightarrow \underline{x} = (x_0, x_1, \ldots, x_{n-2}, x_{n-1}).$$

The vector space $\mathcal{Z}_{2,n}$ is then identified with the space of polynomials $\mathcal{P}_{n-1}[z]$ of degree at most $n-1$.

The addition and multiplication of integers in $\mathcal{Z}_2$ is trivial; similarly, the addition and multiplication of n-vectors in $\mathcal{Z}_{2,n}$ may be defined. The idea is central to understanding Rijndael.

This identification of vectors with polynomials is fruitful; Pierpryzk proved that $p(z) = z^{2^k} + 1$ for $k > 2$ has maximum nonlinearity.

Nyberg [1993] argues that a better definition of $\mathcal{N}_f$ is to find the *best* affine approximation

$$\mathcal{N}_f = d(f, \mathcal{B}_n) \equiv \min_{g \in \mathcal{L}_n} d(f, g)$$

as he proves that, with his definition, the nonlinearity of an *invertible* f is the same as f^{-1}. That is, the measure of the nonlinearity of f is the closest distance to it by a line or affine approximation.

The Boolean functions with Nyberg's maximum nonlinearity have been studied previously in cryptography by Rothhaus [1976]. He called a Boolean function f on $\mathcal{Z}_{2,\mathrm{n}}$ *bent* if its distance to the space of affine functions is a maximum. Various equivalent definitions have been found. First, the *discrete Fourier* (*Hadamard*) or *Walsh transform* of a Boolean function $f(\underline{x})$ is defined by

$$\widehat{F}(\underline{x}) = \sum_{\underline{x} \in \mathcal{Z}_{2,n}} (-1)^{f(\underline{x}) + (\underline{x} \cdot \underline{y})}.$$

The transform operator $f \to \widehat{F}$ satisfies the *Parseval's formula*

$$\sum_{\underline{y} \in \mathcal{Z}_{2,n}} (\widehat{F}(\underline{y}))^2 = 2^{2n}.$$

Rothaus proved that f is bent for $n = 2m$ provided

$$\widehat{F}(y) = \pm 2^m.$$

second, if f is bent and

$$h(\underline{x}) = (\underline{a} \cdot \underline{x}) + b$$

if affine, then

$$(-1)^b \widehat{F}(\underline{y}) = 2^n - 2d(f, h).$$

9.19 GENERATING THE SYMMETRIC GROUP

Product block ciphers acting on plaintext on $\mathcal{Z}_{2,n}$ are often constructed from certain primitives; for example, XOR, addition-with-carry, and circular-shift. DES, `LUCIFER`, and IDEA (defined in Chapter 17) are examples. The *symmetric group* of $\mathcal{Z}_{2,n}$ is the group containing the $2^n!$ permutations of the elements of $\mathcal{Z}_{2,n}$. It is the richest possible cryptographic family; to specify an element of this symmetric group requires $\log_2 2^n! \approx n2^n$ bits

In the design of a product block cipher it seems reasonable to ask if the components of the cipher generate the symmetric group or as large as possible group.

Proposition 9.8: The group generated by the following two operators acting on the n-vectors in $\mathcal{Z}_{2,n}$

9.8a α: addition (with carry) on elements of $\mathcal{Z}_{2,n}$ and

9.8a ρ $[\rho^{-1}]$: shift-left [-right] circular

is the symmetric group of permutations of $\mathcal{Z}_{2,n}$.

Proof: This result does *not* state that a particular group of operators generated by α and ρ is the symmetric group. It does imply, however, that when sufficiently long "strings" of these operations are allowed, then the group "approximates" the symmetric group.

To prove Proposition **9.8** we show that every two-element transposition

$$(i, j), \quad i \equiv \underline{x} \quad j \equiv \underline{y},$$

can be constructed by a suitable composition of $\{\alpha, \rho\}$. The notation $i \equiv \underline{x}$ above means that the integer i is the decimal value of the n-vector in $\underline{x} \in \mathcal{Z}_{2,n}$.

1. The operator $\beta \equiv \rho^{-1}\alpha^2\rho\,\alpha^{-1}$ interchanges the n-vectors $\underbrace{(1, 0, 0, \cdots, 0)}_{(n-1)\text{ copies}}$ and $\underline{0} = \underbrace{(0, 0, \ldots, 0)}_{n\text{ copies}}$.

$$\underbrace{(1, 0, 0, \ldots, 0)}_{(n-1)\text{ copies}} \xrightarrow{\alpha^{-1}} (0, \underbrace{1, 1, \ldots, 1}_{(n-1)\text{ copies}}) \xrightarrow{\rho} (\underbrace{1, 1, \ldots, 1, 0}_{(n-1)\text{ copies}}) \xrightarrow{\alpha^2} (\underbrace{0, 0, \ldots, 0}_{n\text{ copies}}) \xrightarrow{\rho^{-1}} \underbrace{(0, 0, \ldots, 0)}_{\text{copies}}$$

$$(\underbrace{0, 0, \ldots, 0}_{n\text{ copies}}) \xrightarrow{\alpha^{-1}} (\underbrace{1, 1, \ldots, 1}_{n\text{ copies}}) \xrightarrow{\rho} (\underbrace{1, 1, \ldots, 1}_{n\text{ copies}}) \xrightarrow{\alpha^2} (\underbrace{0, 0, \ldots, 0,}_{(n-1)\text{ copies}} 1) \xrightarrow{\rho^{-1}} (\underbrace{1, 0, 0, \ldots, 0}_{(n-1)\text{ copies}}).$$

Furthermore, as we show next, *all* other n-vectors in $\mathcal{Z}_2$ are fixed points under β:

$$(0, \underline{u}, 1, \underbrace{0, 0, \ldots, 0}_{k\text{ copies}}) \xrightarrow{\alpha^{-1}} (0, \underline{u}, 0, \underbrace{1, 1, \ldots, 1}_{k\text{ copies}}) \xrightarrow{\rho} (\underline{u}, 0, \underbrace{1, 1, \ldots, 1, 0}_{k\text{ copies}})$$

$$\xrightarrow{\alpha^2} (\underline{u}, 1, \underbrace{1, 1, \ldots, 1}_{k+1\text{ copies}}) \xrightarrow{\rho^{-1}} (0, \underline{u}, \underbrace{1, 0, 0, \ldots, 0}_{k\text{ copies}})$$

$$(1, \underline{u}, \underbrace{1, 0, 0, \ldots, 0}_{k\text{ copies}}) \xrightarrow{\alpha^{-1}} (1, \underline{u}, 0, \underbrace{1, 1, \ldots, 1}_{k\text{ copies}}) \xrightarrow{\rho} (\underline{u}, 0, \underbrace{1, 1, \ldots, 1}_{(k+1)\text{ copies}})$$

$$\xrightarrow{\alpha^2} (\underline{u}, 1, \underbrace{0, 0, \ldots, 1}_{k\text{ copies}}) \xrightarrow{\rho^{-1}} (1, \underline{u}, 1, \underbrace{0, 0, \ldots, 0}_{k\text{ copies}}).$$

2. Next, we observe that $\gamma \equiv \rho\beta\rho^{-1}$ interchanges the n-vectors $(0, \underbrace{0, 0, \ldots, 0}_{(n-1)\text{ copies}})$ and $\underbrace{(0, 0, \ldots, 0, 1)}_{(n-1)\text{ copies}}$:

$$(\underbrace{0, 0, \ldots, 0, 1}_{(n-1)\text{ copies}}) \xrightarrow{\rho^{-1}} (1, \underbrace{0, 0, \ldots, 0}_{(n-1)\text{ copies}}) \xrightarrow{\beta} (0, \underbrace{0, 0, \ldots, 0}_{(n-1)\text{ copies}}) \xrightarrow{\rho} (0, \underbrace{0, 0, \ldots, 0}_{(n-1)\text{ copies}})$$

$$(0, \underbrace{0, 0, \ldots, 0}_{(n-1)\text{ copies}}) \xrightarrow{\rho^{-1}} (0, \underbrace{0, 0, \ldots, 0}_{(n-1)\text{ copies}}) \xrightarrow{\beta} (1, \underbrace{0, 0, \ldots, 0}_{(n-1)\text{ copies}}) \xrightarrow{\rho} (\underbrace{0, 0, \ldots, 0, 1}_{(n-1)\text{ copies}})$$

and an identical argument as in **1**. above shows that *all* other n-vectors in $\mathcal{Z}_{2,n}$ are fixed points of $\rho^{-1}\beta\rho$.

3. Finally, the operation $\alpha^{r}\gamma\alpha^{-x}$ produces the two-element transposition $(r, r+1)$.

It follows that all two-element transpositions (i, j) may be produced by a *word* involving α, its inverse α^{-1}, together with ρ and its inverse ρ^{-1}. This proves that the group generated is the symmetric group of (invertible) transformations on $\mathcal{Z}_n$.

Every permutation on a finite set S can be written as a product of 2-element transpositions. While this representation is not unique, the parity of a representation is always either even, meaning an even number of 2-element transpositions, or odd. The *alternating group* is composed of those permutations whose transpositions have even parity. Coppersmith and Grossman [1975] show that the *round* transformations of DES and LUCIFER can potentially generate the *alternating group* composed of the elements of the symmetric group of (invertible) transformations with even parity.

9.20 A CLASS OF BLOCK CIPHERS

A "Cryptographic Device" designed by my former colleague Dr. Roy L. Adler is described in IBM [1974] and in U.S. Patent #4.255,811 "Key Control Block Cipher System", issued to Adler on March 10, 1981. This algorithm provides the cryptographic feature in a key-card entry system to be described in Chapter 18.

128-bit plaintext blocks are enciphered to 128-bit ciphertext blocks under ,the control of a 128-bit key:

$$\underline{x} = (x_0, x_1, \ldots, x_{127}) \rightarrow y = (y_0, y_1, \ldots, y_{127}).$$

Like LUCIFER and DES, encipherment is the result of r rounds; the $(3 \times 128) + 7$ bits of key used in a round are derived from the user-supplied 128-bit key in a manner to be described shortly.

First, a 128-bit key $\underline{a}_0 = (a_{0,0}, a_{0,1}, \ldots, a_{0,127})$ derived by the *key processor* from the user-supplied key is added modulo 2^{128} to the 128-bit plaintext block $\underline{x} = (x_0, x_1, \ldots, x_{127})$:

$$\underline{x} \rightarrow \underline{y} \equiv \underline{x} + \underline{a}_0$$

Using the key supplied by the processing device, the steps in the ith round are:

R*i*-1 Modulo 2^{128}-addition of 128-bit key $\underline{b}_i = (b_{i,0}, b_{i,1}, \ldots, b_{i,127})$

$$\underline{y}_0 \rightarrow \underline{y} + \underline{b}_i.$$

R*i*-2 128-to-128 wire-crossing $\underline{\theta} = (\theta_0, \theta_1, \ldots, \theta_{127})$

$$\underline{y}_0 \rightarrow \underline{\theta}(\underline{y} + \underline{b}_i).$$

R*i*-3 7- or 8-bit shift-left-circular ρ_i determined by key $\underline{\beta}_i = (\beta_{i,0}, \beta_{i,1}, \ldots, \beta_{i,7})$

$$\underline{y}_0 \rightarrow \rho_{\underline{\beta}}(\underline{\theta}(\underline{y} + \underline{b}_i)).$$

R*i*-4 128-to-128 inverse wire-crossing $\underline{\theta}_i^{-1} = (\theta_{i,0}^{-1}, \theta_{i,1}^{-1}, \ldots, \theta_{i,7}^{-1})$

$$\underline{y}_0 \rightarrow \underline{\theta}^{-1}(\rho_{\underline{\beta}}(\underline{\theta}(\underline{y} + \underline{b}_i))).$$

R*i*-5 Exclusive-OR of 128-bit key $\underline{c}_i = (c_{i,0}, c_{i,1}, \ldots, c_{i,127})$

$$\underline{y}_0 \rightarrow (\underline{\theta}^{-1}(\rho_{\underline{\beta}}(\underline{\theta}(\underline{y} + \underline{b}_i)))) + \underline{c}_i.$$

The derivation of the internal key by the key processor is depicted in Figure 9.19. The steps in the generation, of the internal key are:

KP-0 The user-supplied key resides in a 128-bit key register **K**;

KP-1 The content of **K** is loaded into registers **Rl**, **R2**, **R3**, and **R4** of sizes 35, 33, 31, and 29 bits;

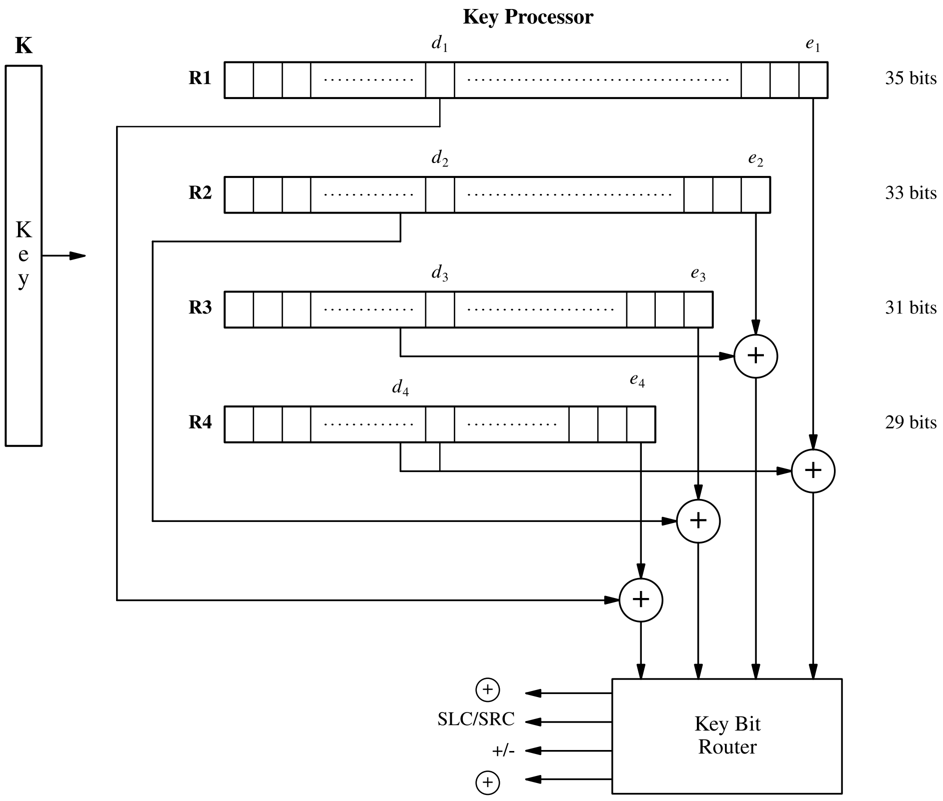


Figure 9.19 Key control block Cipher system.

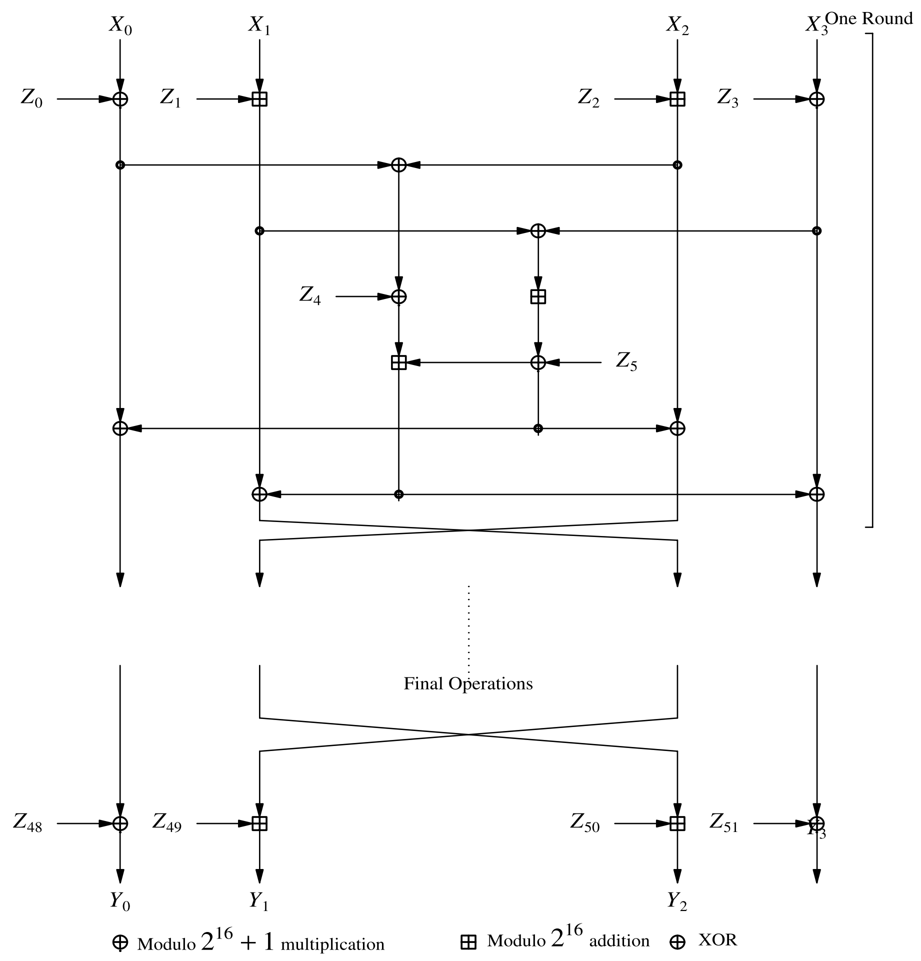


Figure 9.20 The IDEA algorithm.

KP-2 Bits are tapped from register **Ri** at positions d_i and e_i for $i = 1, 2, 3, 4$ in each cycle – the choice tap positions is dependent on the number of rounds r;

KP-3 The XORs

$d_1 \oplus e_4$	$d_2 \oplus e_3$	$d_3 \oplus e_2$	$d_4 \oplus e_1$

are computed at four modulo 2 adders to generate the 4-bit input to tbe *key bit router*;

KP-4 The registers are left-shifted one position after the read operation;

KP-5a Each round takes 32 cycles to generate the required 128 bits for the vector a_0;

KP-5b Each round takes 98 cycles to generate the required $392 = (3 \times 128) + 8$ bits.

As since the lengths of the shift registers. **Rl**, **R2**, **R3**, and **R4** are relatively prime, the key generation process is periodic with period $P = 1{,}038{,}345 = 35 \times 33 \times 31 \times 29$

9.21 THE IDEA BLOCK CIPHER

IDEA (Fig. 9.20) is a block cipher design by Xuejia Lai and James Massey. Its design was influenced by DES; it uses eight rounds to mix the key and plaintext. In each round the basic operations applied to 16-bit variables X_1, X_1, X_2, X_3 are XOR, modulo $2^{16}+1$ multiplication, and modulo 2^{16} addition. Additionally, at the end of each round there is an interchange of the processed blocks X_1 and X_2. At the end of the eighth round there is an additional combination of the key and processed plaintext.

9.21.1 The IDEA Key Schedule

The IDEA key of length 128-bits is divided into 8 blocks of 16 bits $\underline{K} = (K_0, K_i, \ldots, K_7)$. IDEA uses six blocks of 16 bits in each of the eight rounds and four blocks for the final operation. The blocks used in each round are derived as follows:

1. $K_0, K_1, \ldots, K_5$ are used in round 1; K_6, K_7 are the first two blocks in round 2.
2. The 128-bit block $\underline{K}$ is left-shifted 25 places, and the first four 16-bit blocks K_8, K_9, K_{10}, K_{12} are used in round 2.

The process is repeated.

REFERENCES

T. J. Beth and C. Ding, "Almost Perfect Nonlinear Permutations", *Advances in Cryptology – EUROCRYPT '93*, Springer-Verlag (New York, NY), pp. 65–76.

E. Bilham, "On Matsui's Linear Cryptanalysis", in *Advances in Cryptology, EUROCRYPT '94*, 1995, Springer-Verlag (New York, NY), pp. 341–355.

E. Bilham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", *Journal of Cryptology*, **4**, 3–72 (1991).

E. Bilham and A. Shamir, "Differential cryptanalysis of Snefru, Khafre, REDOC II, LOKI and LUCIFER", in *Advances in Cryptology – CRYPTO '91*, Springer-Verlag (New York, NY), 1992. pp. 156–171.

E. Bilham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag (New York, NY), 1993.

C. Charnes and J. Piepzryk, "Linear Nonequivalence Versus Nonlinearity", in *Advances in Cryptology–AUSCRYPT '92*, Springer-Verlag (New York, NY), 1993, pp. 156–164.

D. Coppersmith, "The Data Encryption Standard (DES) and its Strength Against Attacks", *IBM Journal of Research and Development*, **30**, 243–250 (1993).

D. Coppersmith and E. Grossman, "Generators for Certain Alternating Groups with Applications to Cryptography", *SIAM Journal of Applied Mathematics*, **29**, 624–627 (1975).

J. Daemen, "Cipher and Hash Function Design Strategies Based on Linear and Differential Cryptanalysis." Doctoral Dissertation, K. U. Leuven. March 1995.

D. W. Davies and G. I. P. Parkin, "The Average Cycle Size of the Key Stream in Output Feedback Encipherment", *Cryptography, Proceedings of the Workshop on Cryptography*, Berg Feuerstein, Germany, March 1992, Springer-Verlag (New York, NY), pp. 263–279.

J. Detombe and S. Tavares, "Constructing Large Cryptographically Strong S-Boxes", in *Advances in Cryptology – AUSCRYP '92*, Springer-Verlag (New York, NY), 1993, pp. 165–181.

Electronic Frontier Foundation, *Secrets of Encryption Research: Wiretap Politics & Chip Design*, Electronic Frontier Foundation, O'Reilly & Associates (Cambridge, Massachusetts), 1998.

H. Feistel "Cryptography and Computer Privacy", *Scientific American*, **228**, 15–23 (1993).

National Bureau of Standards, Federal Information Processing Standards Publication 46-1, "Data Encryption Standard (DES)", National Bureau of Standards, January 22, 1988; superseded by Federal Information Processing Standards Publication 46-2, December 30, 1993, and reaffirmed as FIPS PUB 46-2, October 25, 1999.

National Bureau of Standards, Federal Information Processing Standards Publication 140-1, "Security Requirements for Cryptographic Modules", January 11, 1994.

National Bureau of Standards, Federal information Processing Standards Publication 197, "Advanced Encryption Standard (AES)", November 26, 2001.

"IBM, 2984-1 Data Protection Programming and Procedure Guide". IBM Corporation, November 22, 1971.

IBM, *IBM Technical Disclosure Bulletin*, **16**(10), 3406–3409 (1974).

A. G. Konheim, *Cryptography: A Prime*, Wiley, 1981.

S. Landau, "Polynomials in the Nation's Service: Using Algebra to Design the Advanced Encryption Standard", *American Mathematical Monthly*, February, 89–117 (2004).

M. Matsui, "Linear Cryptanalysis Method for DES Cipher", in *Advances in Cryptology – EUROCRYPT '93*, Springer-Verlag (New York, NY), 1994, pp. 386–397.

K. Nyberg, "On the Construction of Highly Nonlinear Permutations", in *Advances in Cryptology – EUROCRYPT '92*, Springer-Verlag (New York, NY), pp. 92–98, 1993.

K. Nyberg, "Differentially Uniform Mappings for Cryptology", *Advances in Cryptology – EUROCRYPT '93*, Springer-Verlag (New York, NY), pp. 55–64.

L. O'Connor, "Convergence in Differential Distributions", in *Advances in Cryptology – EUROCRYPT '94*, Springer-Verlag (New York, NY), 1995a, pp. 13–23.

L. O'Connor, "S-boxes and Round Functions with Controllable Linearity and Differentials Uniformity", in *Fast Software Encryption*, Springer-Verlag (New York, NY), 1995b, pp. 111–130.

R. Outerbridge, "Some Design Criteria for Feistel-Cipher Key Schedules, *Cryptologia*, **10**, 142–156 (1986).

J. P. Pierpryzk, "Nonlinearity of Exponent Polynomials", in *Advances in Cryptology: EUROCRYPT '89*, J.-J. Quisquater and J. Vandewalle (eds), Springer-Verlag (New York, NY), 1990, pp. 89–92.

J. Daemen and V. Rijmen, "AES Proposal: Rijndael" (1999b). Available at http://crsc.nist.gov.encryption/aes/rijndael/.

J. Daemen and V. Rijmen, "Annex to AES Proposal: Chapter 5, Propagation and Correlation" (1999a). Available at http://crsc.nist.gov.encryption/aes/rijndael/.

O. S. Rothhaus. "On Bent Functions", *Journal of Combinatorial Theory*, **20A**, pp. 300–305 (1976).

J. Seberry and Xian-Mo Zheng, "Highly Nonlinear 0-1 Balanced Boolean Functions Satisfying Strict Avalanche Criterion", in *Advances in Cryptology – AUSCRYPT '92*, Springer-Verlag (New York, NY), 1993, pp. 145–155.

J. Seberry, Xian-Mo Zhang, and Y. Zheng, "Improving the Strict Avalanche Characteristics of Cryptographic Functions", *Information Processing Letters*, **50**, 1994, pp. 37–41.

J. Seberry, Xian-Mo Zhang, and Y. Zheng, Relationships Among Nonlinearity Criteria", in *Advances in Cryptology – EUROCRYPT '94*, Springer-Verlag (New York, NY), 1995, pp. 376–386.

J. L. Smith, "The Design of Lucifer, A Cryptographic Device for Data Communications", IBM Research Report RC3326, Yorktown Heights, New York, 1971.

A. Sorkin, "Lucifer, A Cryptographic Algorithm", *Cryptologia*, **8**, pp. 22–41 (1984); with addendum *Cryptologia*, **84**, 260–261 (1984).

M. Wiener, "Efficient DES Key Search", reprinted in *Practical Cryptography for Data Internetworks*, William Stallings (ed.), IEEE Computer Society Press (New York, NY), 1996, pp. 31–79.

Bruce Schneier Applied Cryptography (2nd edn), John Wiley, 1993, p. 296.