| Language | $\kappa_p$ |
|---|---|
| French | 0.0778 |
| Spanish | 0.0775 |
| German | 0.0762 |
| Italian | 0.0738 |
| English | 0.0667 |
| Russian | 0.0529 |

**Table 7.1:** *Estimated roughness constant $\kappa_p$ for various languages (see Fact 7.75).*

A polyalphabetic period $t$ may be determined either by Example 7.76 or the alternative of Example 7.77, based on the same underlying ideas. Once $t$ is determined, the situation is as per after successful completion of the Kasiski method.

**7.77 Example** (*determining period by ciphertext auto-correlation*) Given a sample of polyalphabetic ciphertext, the unknown period $t$ may be determined by examining the number of coincidences when the ciphertext is auto-correlated. More specifically, given a ciphertext sample $c_1c_2\ldots c_L$, starting with $t = 1$, count the total number of occurrences $c_i = c_{i+t}$ for $1 \le i \le L - t$. Repeat for $t = 2, 3, \ldots$ and tabulate the counts (or plot a bar graph). The actual period $t^*$ is revealed as follows: for values $t$ that are a multiple of $t^*$, the counts will be noticeably higher (easily recognized as spikes on the bar graph). In fact, for $L$ appropriately large, one expects approximately $L \cdot \kappa_p$ coincidences in this case, and significantly fewer in other cases. □

In the auto-correlation method of coincidences of Example 7.77, the spikes on the bar graph reveal the period, independent of the source language. Once the period is determined, ciphertext characters from like alphabets can be grouped, and the profile of single-character letter frequencies among these, which differs for each language, may be used to determine the plaintext language.

## 7.4 DES

The Data Encryption Standard (DES) is the most well-known symmetric-key block cipher. Recognized world-wide, it set a precedent in the mid 1970s as the first commercial-grade modern algorithm with openly and fully specified implementation details. It is defined by the American standard FIPS 46–2.

### 7.4.1 Product ciphers and Feistel ciphers

The design of DES is related to two general concepts: product ciphers and Feistel ciphers. Each involves iterating a common sequence or round of operations.

The basic idea of a product cipher (see §1.5.3) is to build a complex encryption function by composing several simple operations which offer complementary, but individually insufficient, protection (note cascade ciphers per Definition 7.29 use independent keys). Basic operations include transpositions, translations (e.g., XOR) and linear transformations, arithmetic operations, modular multiplication, and simple substitutions.

**7.78 Definition** A *product cipher* combines two or more transformations in a manner intending that the resulting cipher is more secure than the individual components.

**7.79 Definition** A *substitution-permutation* (SP) *network* is a product cipher composed of a number of stages each involving substitutions and permutations (Figure 7.7).

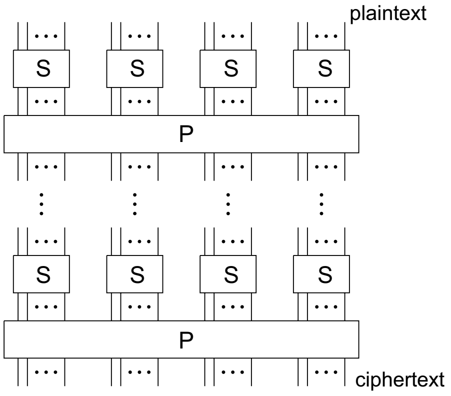


***Figure 7.7:*** *Substitution-permutation (SP) network.*

Many SP networks are iterated ciphers as per Definition 7.80.

**7.80 Definition** An *iterated block cipher* is a block cipher involving the sequential repetition of an internal function called a *round function*. Parameters include the number of rounds $r$, the block bitsize $n$, and the bitsize $k$ of the input key $K$ from which $r$ *subkeys* $K_i$ (round keys) are derived. For invertibility (allowing unique decryption), for each value $K_i$ the round function is a bijection on the round input.

**7.81 Definition** A *Feistel cipher* is an iterated cipher mapping a $2t$-bit plaintext $(L_0, R_0)$, for $t$-bit blocks $L_0$ and $R_0$, to a ciphertext $(R_r, L_r)$, through an $r$-round process where $r \geq 1$. For $1 \leq i \leq r$, round $i$ maps $(L_{i-1}, R_{i-1}) \xrightarrow{K_i} (L_i, R_i)$ as follows: $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, where each *subkey* $K_i$ is derived from the cipher key $K$.

Typically in a Feistel cipher, $r \geq 3$ and often is even. The Feistel structure specifically orders the ciphertext output as $(R_r, L_r)$ rather than $(L_r, R_r)$; the blocks are exchanged from their usual order after the last round. Decryption is thereby achieved using the same $r$-round process but with subkeys used in reverse order, $K_r$ through $K_1$; for example, the last round is undone by simply repeating it (see Note 7.84). The $f$ function of the Feistel cipher may be a product cipher, though $f$ itself need not be invertible to allow inversion of the Feistel cipher.

Figure 7.9(b) illustrates that successive rounds of a Feistel cipher operate on alternating halves of the ciphertext, while the other remains constant. Note the round function of Definition 7.81 may also be re-written to eliminate $L_i$: $R_i = R_{i-2} \oplus f(R_{i-1}, K_i)$. In this case, the final ciphertext output is $(R_r, R_{r-1})$, with input labeled $(R_{-1}, R_0)$.

### 7.4.2 DES algorithm

DES is a Feistel cipher which processes plaintext blocks of $n = 64$ bits, producing 64-bit ciphertext blocks (Figure 7.8). The effective size of the secret key $K$ is $k = 56$ bits; more precisely, the input key $K$ is specified as a 64-bit key, 8 bits of which (bits $8, 16, \ldots, 64$) may be used as parity bits. The $2^{56}$ keys implement (at most) $2^{56}$ of the $2^{64}!$ possible bijections on 64-bit blocks. A widely held belief is that the parity bits were introduced to reduce the effective key size from 64 to 56 bits, to intentionally reduce the cost of exhaustive key search by a factor of 256.

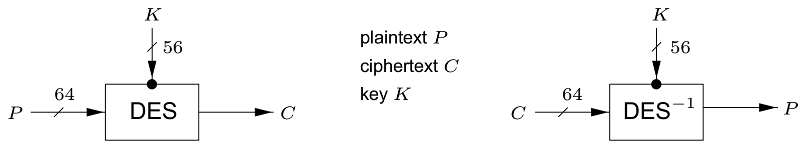


***Figure 7.8:*** *DES input-output.*

Full details of DES are given in Algorithm 7.82 and Figures 7.9 and 7.10. An overview follows. Encryption proceeds in 16 stages or *rounds*. From the input key $K$, sixteen 48-bit subkeys $K_i$ are generated, one for each round. Within each round, 8 fixed, carefully selected 6-to-4 bit substitution mappings (*S-boxes*) $S_i$, collectively denoted $S$, are used. The 64-bit plaintext is divided into 32-bit halves $L_0$ and $R_0$. Each round is functionally equivalent, taking 32-bit inputs $L_{i-1}$ and $R_{i-1}$ from the previous round and producing 32-bit outputs $L_i$ and $R_i$ for $1 \leq i \leq 16$, as follows:

$$L_i = R_{i-1}; \tag{7.4}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \quad \text{where } f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i)) \tag{7.5}$$

Here $E$ is a fixed expansion permutation mapping $R_{i-1}$ from 32 to 48 bits (all bits are used once; some are used twice). $P$ is another fixed permutation on 32 bits. An initial bit permutation (IP) precedes the first round; following the last round, the left and right halves are exchanged and, finally, the resulting string is bit-permuted by the inverse of IP. Decryption involves the same key and algorithm, but with subkeys applied to the internal rounds in the reverse order (Note 7.84).

A simplified view is that the right half of each round (after expanding the 32-bit input to 8 characters of 6 bits each) carries out a key-dependent substitution on each of 8 characters, then uses a fixed bit transposition to redistribute the bits of the resulting characters to produce 32 output bits.

Algorithm 7.83 specifies how to compute the DES round keys $K_i$, each of which contains 48 bits of $K$. These operations make use of tables PC1 and PC2 of Table 7.4, which are called *permuted choice 1* and *permuted choice 2*. To begin, 8 bits ($k_8, k_{16}, \ldots, k_{64}$) of $K$ are discarded (by PC1). The remaining 56 bits are permuted and assigned to two 28-bit variables $C$ and $D$; and then for 16 iterations, both $C$ and $D$ are rotated either 1 or 2 bits, and 48 bits ($K_i$) are selected from the concatenated result.

**7.82 Algorithm** Data Encryption Standard (DES)

INPUT: plaintext $m_1 \dots m_{64}$; 64-bit key $K = k_1 \dots k_{64}$ (includes 8 parity bits).
OUTPUT: 64-bit ciphertext block $C = c_1 \dots c_{64}$. (For decryption, see Note 7.84.)

```
1. (key schedule) Compute sixteen 48-bit round keys $K_i$ from $K$ using Algorithm 7.83.
2. $(L_0, R_0) \leftarrow \mathrm{IP}(m_1 m_2 \dots m_{64})$. (Use IP from Table 7.2 to permute bits; split the
   result into left and right 32-bit halves $L_0 = m_{58} m_{50} \dots m_8$, $R_0 = m_{57} m_{49} \dots m_7$.)
3. (16 rounds) for $i$ from 1 to 16, compute $L_i$ and $R_i$ using Equations (7.4) and (7.5)
   above, computing $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$ as follows:
   (a) Expand $R_{i-1} = r_1 r_2 \dots r_{32}$ from 32 to 48 bits using $E$ per Table 7.3:
       $T \leftarrow E(R_{i-1})$. (Thus $T = r_{32} r_1 r_2 \dots r_{32} r_1$.)
   (b) $T' \leftarrow T \oplus K_i$. Represent $T'$ as eight 6-bit character strings: $(B_1, \dots, B_8) = T'$.
   (c) $T'' \leftarrow (S_1(B_1), S_2(B_2), \dots S_8(B_8))$. (Here $S_i(B_i)$ maps $B_i = b_1 b_2 \dots b_6$
       to the 4-bit entry in row $r$ and column $c$ of $S_i$ in Table 7.8, page 260 where
       $r = 2 \cdot b_1 + b_6$, and $b_2 b_3 b_4 b_5$ is the radix-2 representation of $0 \le c \le 15$. Thus
       $S_1(011011)$ yields $r = 1$, $c = 13$, and output 5, i.e., binary 0101.)
   (d) $T''' \leftarrow P(T'')$. (Use $P$ per Table 7.3 to permute the 32 bits of $T'' = t_1 t_2 \dots t_{32}$,
       yielding $t_{16} t_7 \dots t_{25}$.)
4. $b_1 b_2 \dots b_{64} \leftarrow (R_{16}, L_{16})$. (Exchange final blocks $L_{16}, R_{16}$.)
5. $C \leftarrow \mathrm{IP}^{-1}(b_1 b_2 \dots b_{64})$. (Transpose using $\mathrm{IP}^{-1}$ from Table 7.2; $C = b_{40} b_8 \dots b_{25}$.)
```

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

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

***Table 7.2:*** *DES initial permutation and inverse (IP and IP$^{-1}$).*

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

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

***Table 7.3:*** *DES per-round functions: expansion $E$ and permutation $P$.*

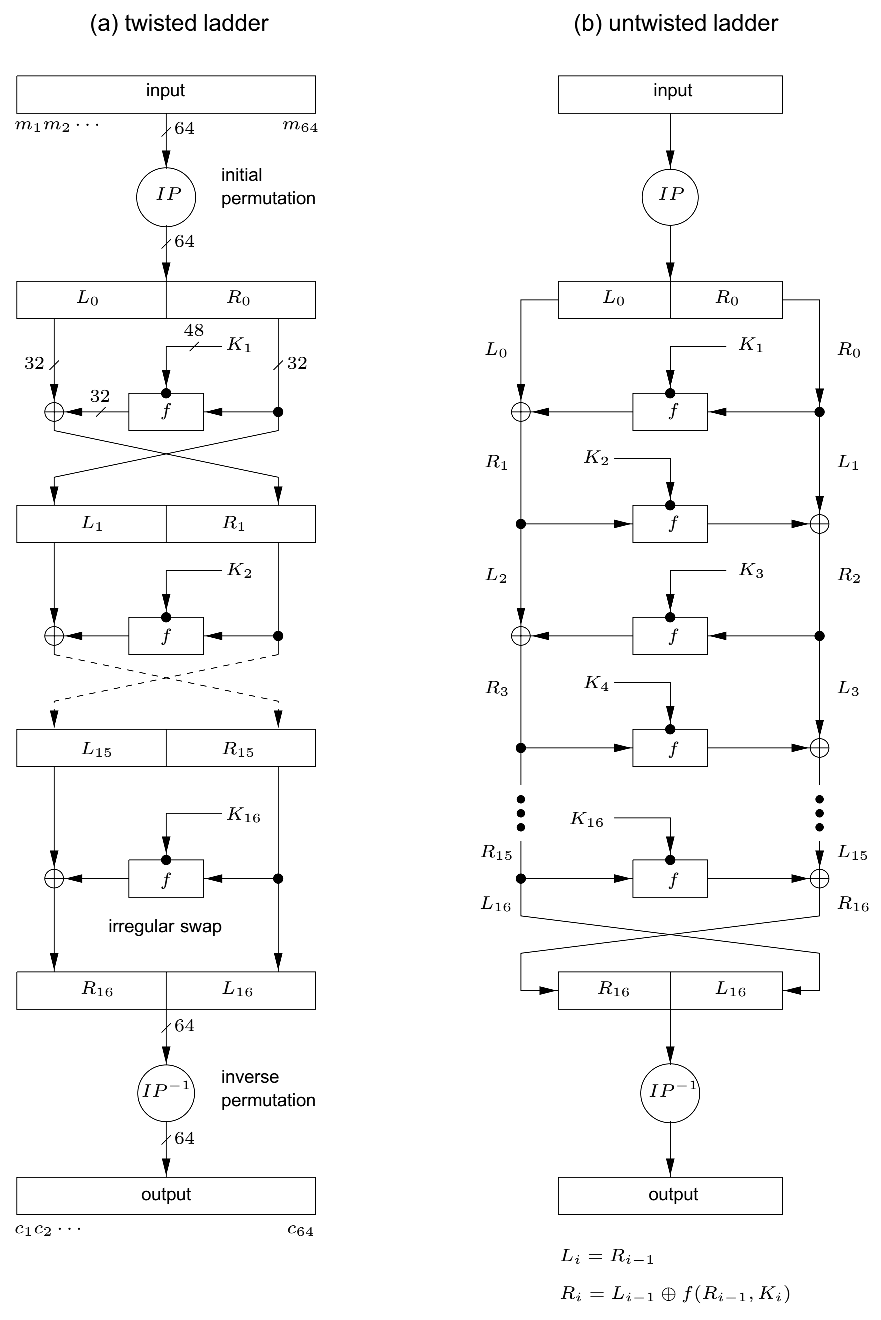
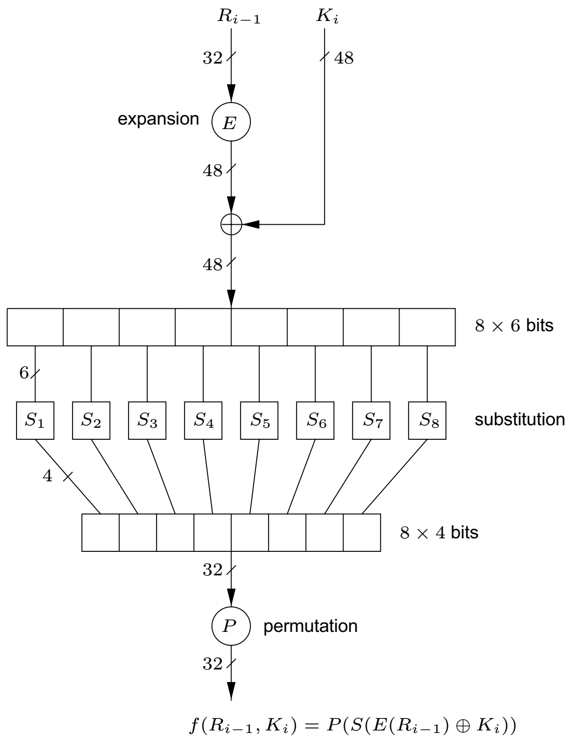


***Figure 7.9:*** *DES computation path.*



$R_{i-1}$
$K_i$
32
48
expansion
$E$
48
48
8 × 6 bits
6
$S_1$ $S_2$ $S_3$ $S_4$ $S_5$ $S_6$ $S_7$ $S_8$
substitution
4
8 × 4 bits
32
$P$
permutation
32
$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$

***Figure 7.10:*** *DES inner function $f$.*

**7.83 Algorithm** DES key schedule

INPUT: 64-bit key $K = k_1 \dots k_{64}$ (including 8 odd-parity bits).
OUTPUT: sixteen 48-bit keys $K_i$, $1 \le i \le 16$.

1. Define $v_i$, $1 \le i \le 16$ as follows: $v_i = 1$ for $i \in \{1, 2, 9, 16\}$; $v_i = 2$ otherwise. (These are left-shift values for 28-bit circular rotations below.)
2. $T \leftarrow$ PC1($K$); represent $T$ as 28-bit halves $(C_0, D_0)$. (Use PC1 in Table 7.4 to select bits from $K$: $C_0 = k_{57}k_{49} \dots k_{36}$, $D_0 = k_{63}k_{55} \dots k_4$.)
3. For $i$ from 1 to 16, compute $K_i$ as follows: $C_i \leftarrow (C_{i-1} \hookleftarrow v_i)$, $D_i \leftarrow (D_{i-1} \hookleftarrow v_i)$, $K_i \leftarrow$ PC2($C_i, D_i$). (Use PC2 in Table 7.4 to select 48 bits from the concatenation $b_1b_2 \dots b_{56}$ of $C_i$ and $D_i$: $K_i = b_{14}b_{17} \dots b_{32}$. '$\hookleftarrow$' denotes left circular shift.)

If decryption is designed as a simple variation of the encryption function, savings result in hardware or software code size. DES achieves this as outlined in Note 7.84.

**7.84 Note** (*DES decryption*) DES decryption consists of the encryption algorithm with the same key but reversed key schedule, using in order $K_{16}, K_{15}, \dots, K_1$ (see Note 7.85). This works as follows (refer to Figure 7.9). The effect of IP$^{-1}$ is cancelled by IP in decryption, leaving $(R_{16}, L_{16})$; consider applying round 1 to this input. The operation on the left half yields, rather than $L_0 \oplus f(R_0, K_1)$, now $R_{16} \oplus f(L_{16}, K_{16})$ which, since $L_{16} = R_{15}$ and $R_{16} = L_{15} \oplus f(R_{15}, K_{16})$, is equal to $L_{15} \oplus f(R_{15}, K_{16}) \oplus f(R_{15}, K_{16}) = L_{15}$. Thus round 1 decryption yields $(R_{15}, L_{15})$, i.e., inverting round 16. Note that the cancellation

| PC1 | | | | | | |
|---|---|---|---|---|---|---|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| above for $C_i$; below for $D_i$ | | | | | | |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

| PC2 | | | | | |
|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 |
| 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 |
| 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 |
| 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |

**Table 7.4:** *DES key schedule bit selections (PC1 and PC2).*

of each round is independent of the definition of $f$ and the specific value of $K_i$; the swapping of halves combined with the XOR process is inverted by the second application. The remaining 15 rounds are likewise cancelled one by one in reverse order of application, due to the reversed key schedule.

**7.85 Note** (*DES decryption key schedule*) Subkeys $K_1, \dots, K_{16}$ may be generated by Algorithm 7.83 and used in reverse order, or generated in reverse order directly as follows. Note that after $K_{16}$ is generated, the original values of the 28-bit registers $C$ and $D$ are restored (each has rotated 28 bits). Consequently, and due to the choice of shift-values, modifying Algorithm 7.83 as follows generates subkeys in order $K_{16}, \dots, K_1$: replace the left-shifts by right-shift rotates; change the shift value $v_1$ to 0.

**7.86 Example** (*DES test vectors*) The plaintext "Now is the time for all ", represented as a string of 8-bit hex characters (7-bit ASCII characters plus leading 0-bit), and encrypted using the DES key specified by the hex string $K$ = `0123456789ABCDEF` results in the following plaintext/ciphertext:

$P$ = `4E6F772069732074 68652074696D6520 666F7220616C6C20`

$C$ = `3FA40E8A984D4815 6A271787AB8883F9 893D51EC4B563B53`. □

## 7.4.3 DES properties and strength

There are many desirable characteristics for block ciphers. These include: each bit of the ciphertext should depend on all bits of the key and all bits of the plaintext; there should be no statistical relationship evident between plaintext and ciphertext; altering any single plaintext or key bit should alter each ciphertext bit with probability $\frac{1}{2}$; and altering a ciphertext bit should result in an unpredictable change to the recovered plaintext block. Empirically, DES satisfies these basic objectives. Some known properties and anomalies of DES are given below.

### (i) Complementation property

**7.87 Fact** Let $E$ denote DES, and $\overline{x}$ the bitwise complement of $x$. Then $y = E_K(x)$ implies $\overline{y} = E_{\overline{K}}(\overline{x})$. That is, bitwise complementing both the key $K$ and the plaintext $x$ results in complemented DES ciphertext.

*Justification*: Compare the first round output (see Figure 7.10) to $(L_0, R_0)$ for the uncomplemented case. The combined effect of the plaintext and key being complemented results

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

**Table 7.8:** *DES S-boxes.*