THE DATA ENCRYPTION STANDARD (DES)

1. Description of DES

DES (FIPS 46-3) is a block cipher that operates on 64-bit blocks. Keys are 64 bits long, but 8 of these are parity bits, leaving 56 actual key bits. The basic operation of DES is as follows:

(1) The 64 plaintext bits are permuted in a fixed order (transposition cipher).
(2) The block is divided into two 32-bit words L_0 and R_0.
(3) The block undergoes 16 substitution "rounds." In each round, one word is transformed using XOR and a substitution function, after which the two words are swapped.
(4) In the last round, the two words are not swapped.
(5) The original permutation is reversed.

Thus $DES_{key}(M) = IP^{-1}(S_{16}(S_{15}(\ldots(S_2(S_1(IP(M))))\ldots)))$. The overall structure of DES is illustrated pictorially in Figure 1.

We now describe the individual components of DES.

1.1. **Initial Permutation.**

Simply a fixed re-ordering of the input bits, described in the FIPS document. Notation: the first bit of the output is the 58th bit of the input.

1.2. **Substitution Rounds.**

In round i, the right word R_{i-1} is combined with the ith subkey K_i via the function f. The output of f is XORed with L_{i-1} to form R_i. The next left word L_i is the previous right word ($L_i = R_{i-1}$).

1.3. **The Function f.**

f accepts as input R_i (32 bits) and the ith round subkey K_i (48 bits). The subkey generation is described below. The function f works as follows:

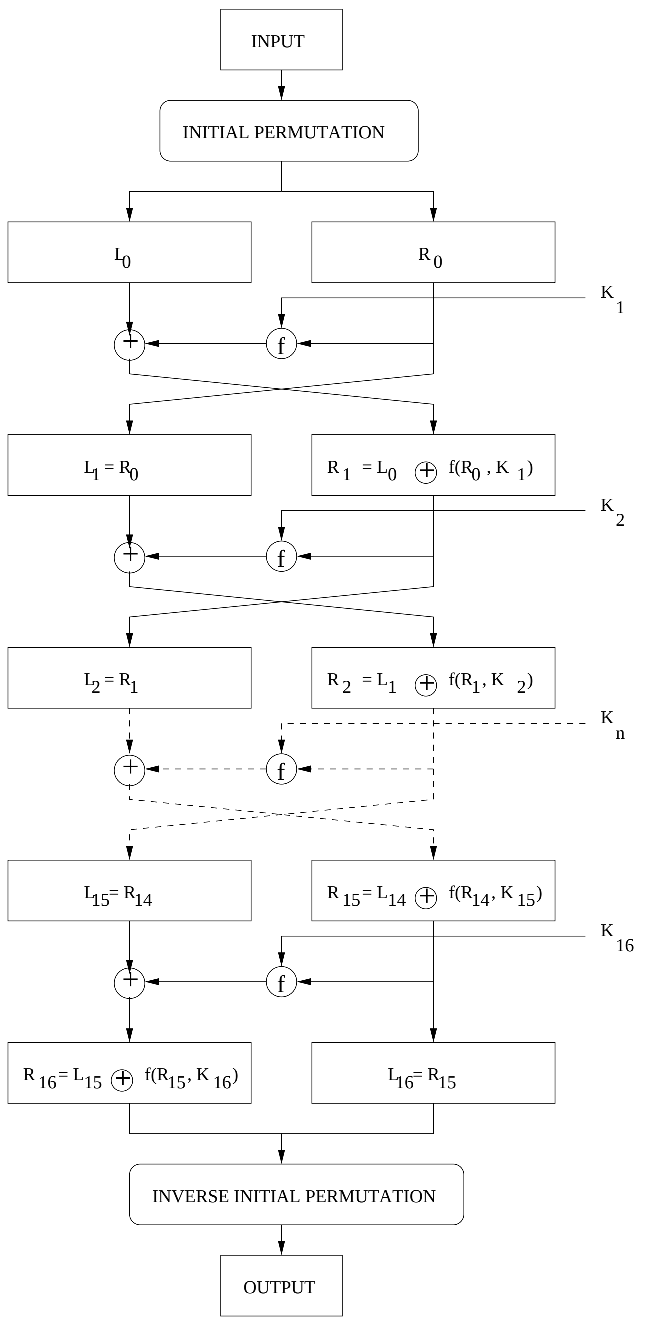
(1) R_i is expanded to the 48 bit $R_i^{'}$ via the expansion function E, which simply repeats some of the bits of R_i in generating $R_i^{'}$ (see the FIPS publication for the specification of E).
(2) $R_i^{'}$ is XORed with K_i (both are 48 bits long).
(3) $R_i^{'} \oplus K_i$ is broken into 8 6-bit words. Each of these words is replaced by a 4-bit word according to the 8 (different) S-boxes $S_1, S_2, \ldots, S_8$. The result of applying the S-boxes is a 32-bit string.
(4) The 32-bit string is permuted according to the fixed permutation P (see the FIPS publication).

Figure 2 contains a pictorial description of f.

1.4. **Generation of the Subkeys K_i.**

(1) The key K (56 bits) is permuted according to the fixed permutation "PERMUTED CHOICE 1" (see FIPS publication) and separated into two 28-bit words C_0 and D_0.
(2) Each word is rotated either one or two places to the left according to the fixed schedule below, yielding C_1 and D_1.

FIGURE 1. Diagram of DES



Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of left shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

(3) K_1 is obtained from C_1 and D_1 via "PERMUTED CHOICE 2," which selects 48 bits from C_1 and D_1 according to a fixed ordering.
(4) Steps 2 and 3 are repeated with C_i and D_i to obtain the remaining 15 subkeys.

This process is illustrated in Figure 3

Figure 2. DES function f

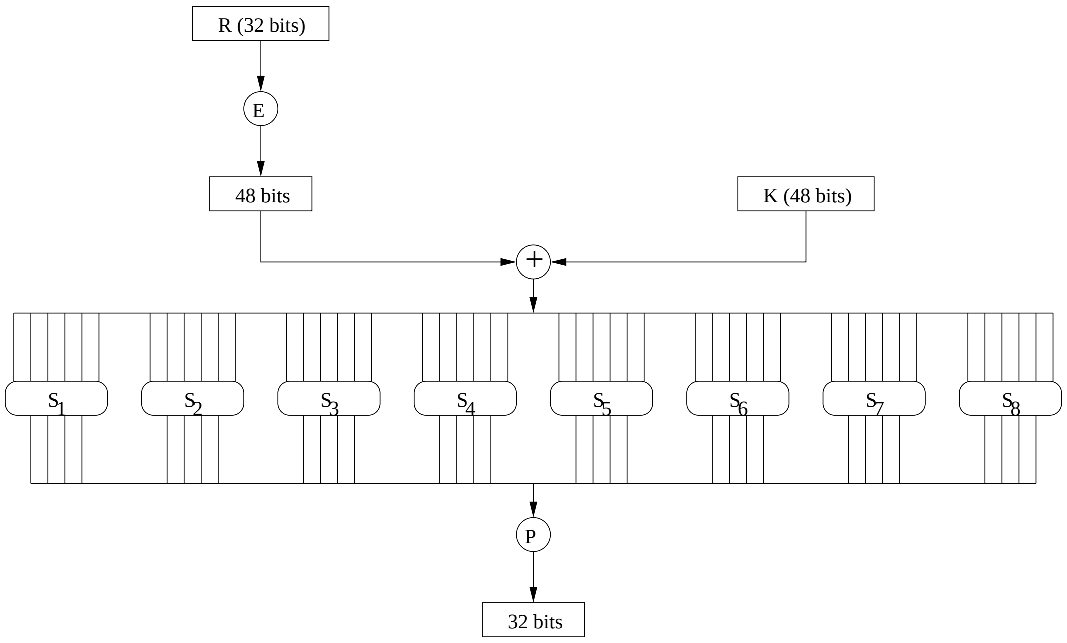
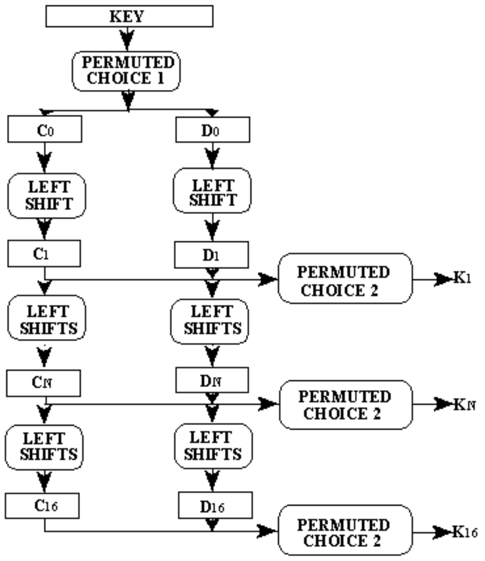


Figure 3. DES Key Schedule Algorithm



1.5. **DES Decryption.** The structure of DES allows for easy decryption as long as the key is known. Before describing the DES decryption algorithm, we state a few properties of the components of the algorithm. We denote by $K_n = KS(n, key)$ the nth subkey, where KS is the key schedule.

It is easy to see that the following properties all hold:

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus f(R_i, K_{i+1}) \quad i = 0, 1, \ldots, 15 \\ L_i &= R_{i+1} \oplus f(R_i, K_{i+1}) \\ R_{i+1} \oplus L_i &= f(R_i, K_{i+1}) \end{aligned}$$

In addition, we have

$$\begin{aligned} C &= IP^{-1}(R_{16}, L_{16}) \\ IP(C) &= (R_{16}, L_{16}) \end{aligned}$$

where IP^{-1} is the inverse of the initial permutation function.

Suppose $DES_{key}(M) = C$. Denote by

$$K_i^{'} = KS(17 - i, key), \quad i = 1, 2, \ldots, 16$$

(the key schedule in reverse order). Now run the DES device on C, using K_i' instead of K_i. We have $IP(C) = (R_{16}, L_{16})$, and thus

$$\begin{aligned} L_0^{'} &= R_{16} & R_0^{'} &= L_{16} \\ L_1^{'} &= R_0^{'} = L_{16} & R_1^{'} &= L_0^{'} \oplus f(R_0^{'}, K_1^{'}) \\ &= R_{15} & &= R_{16} \oplus f(L_{16}, K_{16}) \\ & & &= R_{16} \oplus f(R_{15}, K_{16}) \\ & & &= R_{16} \oplus R_{16} \oplus L_{15} \\ & & &= L_{15} \end{aligned}$$

In fact, by continuing this argument we get

$$L_i^{'} = R_{16-i} \quad R_i^{'} = L_{16-i}$$

and hence

$$IP^{-1}(R_{16}^{'}, L_{16}^{'}) = IP^{-1}(L_0, R_0) = M$$

Thus, decryption of DES is simply running the DES algorithm on C with the reverse key schedule.

Note. The invertibility of DES is *independent* of the function f. Regardless of what function is used for f, decryption of DES works exactly as described above. This works largely because the individual parts of DES are *involutions* — functions that are their own inverses ($g(g(x)) = x$)

2. Strengths and Weaknesses of DES

Under DES, encryption of 0 (plaintext consisting of 64 zeros) under the key 0 produces:

```
Plaintext   0000  0000  0000  0000 (in hex)
L1 R1       0000  0000  D8D8  DBBC
L2 R2       D8D8  DBBC  E73A  ED4F
                     ⋮
L16 R16     BBEA  0DC2  1C20  87FC
Ciphertext  8CA6  4DE9  C1B1  23A7
```

2.1. **Error Propagation.** Small modifications in the plaintext should cause a lot of modification in the corresponding ciphertext. In particular, we want that changing one bit of plaintext causes one half of the ciphertext bits to change on average (strict avalanche condition).

In the following example, the keys are the same and the plaintexts differ only in the last bit:

$$K = 0^{56} \qquad P = (10)^{32} \qquad C = \text{3AE7 1695 4DC0 4E25}$$
$$K = 0^{56} \qquad P = (10)^{31}(11) \qquad C = \text{17D8 E9C3 74D1 4494}$$

Note that the ciphertexts differ in 34 bits. Statistically, they are as far apart from each other as random sequences.

Similarly, we want that changing one key bit causes one half of the ciphertext bits to change on average:

$$K = 0^{56} \qquad P = (01)^{32} \qquad C = \text{B109 FD80 3EB2 D05E}$$
$$K = 0^{55}1 \qquad P = (01)^{32} \qquad C = \text{451F 0C33 F24F B8DC}$$

Again, 34 bits differ between the two ciphertexts.

2.2. **Cryptanalysis of DES.**

(1) Exhaustive search: There are $2^{56} \approx 10^{17}$ possible keys. If one key is tested every μsec, the complete search would take 2.2 centuries in the worst case (1.1 centuries on average).
(2) Parallelism: A search machine consisting of 10^6 chips each testing one key per μsec (total of 10^{12} keys per second) would find a key in one day. Electronic Frontier Foundation has built a DES cracker for $250000 which finds a single DES key in 56 hours (tests 8800 keys per μsec). A combination of the DES cracker and 100000 PC's on the internet has found a DES key in 22.25 hours (tests 245000 keys per μsec).
(3) Time-memory tradeoff (Hellman): Shorten time by using a lot of memory — applies to any brute-force attack on a cryptosystem.

The time-memory tradeoff idea can be applied as follows. Fix the following notation:

$$\begin{aligned} P &= \text{64-bit plaintext} \\ C &= \text{64-bit ciphertext} \\ K &= \text{56-bit key} \\ DES : C &= S_K(P) \end{aligned}$$

Let P_0 be a fixed plaintext block, for example, 8 ASCII blanks (this is a chosen-text attack).

Define $f(K) = R(S_K(P_0))$, where R is a reducing function which throws away 8 bits.

Start with m points $SP_1, SP_2, \ldots, SP_m$ selected at random from the key space $\{0, 1, \ldots, 2^{56} - 1\}$. For $1 \le i \le m$ let $X_{i,0} = SP_i$. Compute $X_{i,j} = f(X_{i,j-1})$ for $1 \le j \le t$:

$$\begin{aligned} SP_1 = X_{1,0} &\to X_{1,2} \to \cdots \to X_{1,t} = EP_1 \\ SP_2 = X_{2,0} &\to X_{2,2} \to \cdots \to X_{2,t} = EP_2 \\ &\vdots \\ SP_m = X_{m,0} &\to X_{m,2} \to \cdots \to X_{m,t} = EP_m \end{aligned}$$

The endpoints $EP_i = f^{(t)}(SP_i)$. Discard all the intermediate points and sort

$$S = \{(SP_i, EP_i) \mid i = 1, 2, \ldots, m\}$$

on the end points in ascending order. The steps up to this point amount to a precomputation *before* starting the real attack.

Now, suppose someone selects a key K, and the cryptanalyst intercepts $C_0 = S_K(P_0)$ (i.e., he knows which ciphertext block C_0 corresponds to P_0). Let $Y_1 = R(C_0) = f(K)$. He then checks whether Y_1 is one of the endpoints in S (very fast since S is sorted). If $Y_i = EP_i$, then either $K = X_{i,t-1}$ or EP_i has more than one inverse under f. The cryptanalyst can then check for a false solution by checking that $C_0 = S_K(P_0)$ for $K = X_{i,t-1}$ (he needs P_0 at this point). If no endpoint is Y_i or we have a false solution, try $Y_2 = f(Y_1)$ and compare Y_2 to the endpoints. If $Y_2 = EP_i$ then $K = X_{i,t-2}$ or we have a false solution. Continue the process on Y_3, etc. until we find K.

Hellman proved that if $f(i)$ is a random function and $t^2 m = N = 2^{56}$, the expected probability of success by this method is

$$\frac{mt}{N} = \frac{1}{t} \ .$$

Here $N = 2^{56}$, so if we set $t = \sqrt[3]{N}$ we get $1/t \approx 10^{-6}$ (small probability of success). Thus, in order to get an expected probability of success closer to 1, we compute $t \approx 10^6$ tables like S, where for each of these tables we use a different reducing function R. This yields:

Expected time:	$tt = t^2$
Expected memory:	$mt = t^2$ (since $m = t = \sqrt[3]{N}$)

2.3. **Analytic Attacks on DES.** If a cryptosystem is *linear* then

$$C = AP + BK$$

where C is the ciphertext, P is the plaintext, K is the key, and A and B are matrices determined by the system (A and B are public). Note that B may or may not be square. To decrypt:

$$P = A^{-1}(C - BK) \ .$$

A cryptanalyst can easily mount a known text attack (P and C known) as follows:

$$\begin{aligned} BK &= C - AP \\ B^T BK &= B^T(C - AP) \\ K &= (B^T B)^{-1} B^T (C - AP) \end{aligned}$$

Thus, linear cryptosystems are not secure (Hill cipher is a simple example). What about DES?

If DES were linear, we would have the following matrix sizes:

$$A : 64 \times 64, \quad B : 64 \times 56,$$
$$K : 56 \times 1, \quad P : 64 \times 1, \quad C : 64 \times 1$$

In DES, the expansion operation E, the permutations, and XOR are all linear. All that is left is the S-boxes — if they are also linear, then DES is linear (and hence insecure).

If the S-boxes are *affine*, i.e., $y = Gx + h$, where

G is a 4×6 matrix
h is a 4×1 matrix
x is the 6×1 input
y is the 4×1 output,

then we can find a matrix H such that for DES

$$C = AP + BK + H$$
$$K = (B^T B)^{-1} B^T (C - AP - H) \ .$$

Thus, an affine cipher is as easy to break as a linear cipher using KTA (note A, B, and H are public).

Suppose the DES S-boxes were affine. If we modify only 4 entries in each of them, the resulting S boxes need no longer be affine, but affine cryptanalysis would nevertheless yield a solution one time in 3870. To see this, note that there are 60/64 unmodified entries in each S-box, 8 S-boxes in total, and 16 rounds, so the probability of running DES without accessing any of the modified S-box entries is

$$\left(\frac{60}{64}\right)^{8\times 16} \approx \frac{1}{3870} \ .$$

Thus, linearity or affineness can be exploited if the cryptosystem is *close* to being affine or linear. This technique is called *linear cryptanalysis* (M. Matusi, EUROCRYPT 1993), and will break DES in time 2^{43} (better than exhaustive search), and Matusi actually used this method to become the first person to recover a DES key in 50 days using twelve workstations. However, this attack is a known text attack and requires 2^{43} matched pairs of plaintext and ciphertext (not very practical in general).

Differential cryptanalysis (Biham and Shamir, Journal of Cryptology, 1991): chosen text attack requiring 2^{47} chosen plaintext/ciphertext pairs.

Large-scale, parallel, brute-force attack is the most practical attack.

2.4. **Strengths of DES.**

(1) No S-box is affine
(2) No portion of an S-box is affine.
(3) The overall algorithm is not affine.
(4) No S-box is translation invariant ($S(x) \neq S(x + E)$)
(5) Changing a single input bit to any S-box causes at least two output bits to change.
(6) P and E are coupled to guarantee that the four outputs of each S-box are taken to six different S-boxes at the next round.
(7) Rapid error-propagation hinders a key clustering attack (testing a smaller set of similar keys)