DES: Data Encryption Standard

The scheme uses a 56 bit key (transmitted as eight septuplets with additional parity bit, hence as 64 bits). The scheme works on blocks of input of length 64b. The operations are:

1. initial shuffling (permutation) of the input block,
2. In 16 rounds, the previous 64 bits are transformed using a 48 bit sub-key of the original 56 bit key.
3. The left half (32 bits) is swapped with the right half (32 bits).
4. The resulting block (64 bits) is subjected to a final shuffling.

The initial and the final permutation are inverses of each other. Thus, if you encode the same data block twice with possibly different keys, the steps (4) of the first encoding and (1) of the second encoding cancel each other out. Since the initial and final permutations are known, they do not increase the security of the scheme.

The round keys $k_i$, $i=1, ..., 12$ are generated from various permutations of the key followed by selecting the 48 bits that make up the key.

For encryption in round $i$, the input block $I_i$ is divided into two 32 bit halves, $L_{i-1}$ and $R_{i-1}$: $I_{i-1} = (L_{i-1}|R_{i-1})$. Then we transform

\[ L_i = R_{i-1} \]
\[ R_i = L_{i-1} \oplus f(K_i, R_{i-1}) \]

to obtain the two halves output $I_i$ of round $i$: $I_i = (L_i|R_i)$. The mangler function $f$ returns a 32 bit result for the 48 bit key and the 32 bit round input $R_{i-1}$.

Decryption works by backtracking. This is easy for all steps but the rounds. If we have $I_i = (L_i|R_i)$, we swap the two halves to obtain $(R_i|L_i)$, which in turn we subject to the round calculation to obtain

\[ (L_i | R_i \oplus f(K_i, L_i)) = (R_{i-1} | L_{i-1} \oplus f(K_i, R_{i-1}) \oplus f(K_j, R_{i-1})) = (R_{i-1}, L_{i-1}) \]

which are of course $I_{i-1}$ with the two halves swapped. Thus, the decryption algorithm uses the same components as the encryption.

The mangler function $f$ takes the 32 bit input $R = (r_0, r_1, r_2, ..., r_{31})$ and generates eight chunks $C_i$. Each chunk $C_i$ consists of 6 adjacent bits of the input, starting at different positions of $R$. The first chunk consists of bits ($r_{31}, r_0, r_1, r_2, r_3, r_4$), the second chunk of bits ($r_5, r_6, r_7, r_8, r_9$), etc. We break the key $K_i$ into eight chunks of six bits as well, but this time in the canonical way since the key consists of 48 bits. The key chunks and the input chunks are then XORed together and the resulting six bits for chunks $i$ are then transformed into four bits using an “$S$ box” $S_i$. The design of the $S$ boxes is somewhat arbitrary – a fact that raises suspicion. The resulting six four-bit chunks are then combined into a 32 bit quantity, that is subjected to a final permutation.

Major criticism of DES include:
• The key is too short. A 56 bit key gives $2^{56}$ or approximately $10^{17}$ possible keys. In 1977 Diffie and Hellman proposed a DES breaking machine that uses exhaustive search for a plaintext ciphertext attack. The machine then cost $20M and could find a DES key in twelve hours. Since then, hardware prices have gone down and performance up. The Electronic Frontier Foundation build a special purpose EFF DES Cracker for $250K that could do the same job in four and a half days. 40% of the costs was the design, so you can build a replacement machine for $150K. However, you could use triple encryption with three different keys and estimates are that this is $2^{56}$ times as difficult to crack.

• The initial and final permutations do not add to the security of DES. They just make it more difficult to build DES in software.

• The design of the S-boxes has no rational. There are people suspecting the benevolence of the U.S. government and believe that they have been chosen to give the NSA a backdoor into monitoring traffic. Since this debate, people use pseudo-random numbers derived from well-known constants such as the square root of two or $\pi$ in their designs.

• There are four “weak” keys. Encrypting with the key yields the same result as decryption. There are twelve semi-weak keys, which come in pairs. Encrypting with one key gives the same result as decrypting with another.