3 DATA ENCRYPTION STANDARD DES

The DES algorithm is widely used in banking, government and embedded applications. For example, it is the standard in automatic teller machine networks. It is a Feistel cipher, with a 64-bit block and 56-bit key. Its round function operates on 32-bit half blocks and consists of three operations:

- first, the block is expanded from 32 bits to 48;
- next, 48 bits of round key are mixed in using exclusive-or;
- the result is passed through a row of eight S-boxes, each of which takes a six-bit input and provides a four-bit output;
- finally, the bits of the output are permuted according to a fixed pattern.

The effect of the expansion, key mixing and S-boxes is shown in Figure

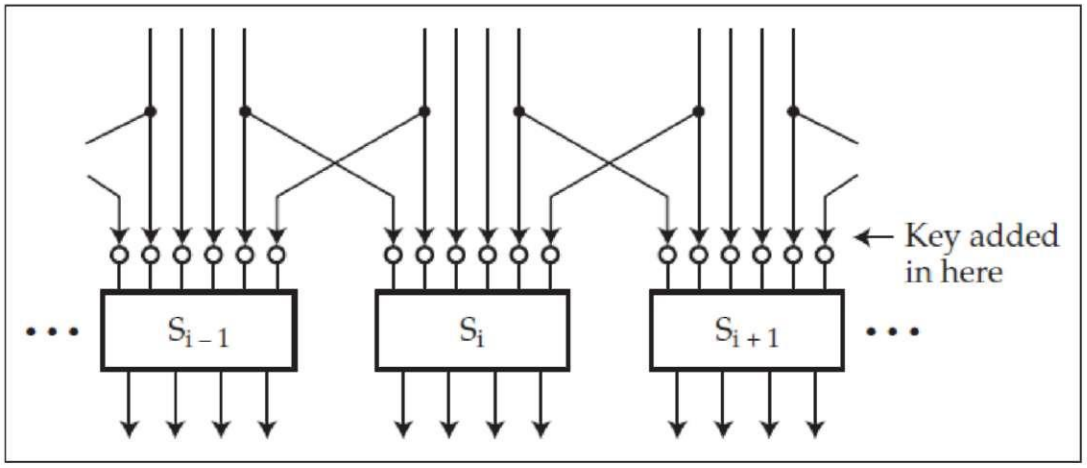

Figure The DES round function

The round keys are derived from the user-supplied key by using each user key bit in twelve different rounds according to a slightly irregular pattern. A full specification of DES is given in [936]; code can be found in [1125] or downloaded from many places on the web.

DES was introduced in 1974 and caused some controversy. The most telling criticism was that the key is too short. Someone who wants to find a 56 bit key using brute force, that is by trying all possible keys, will have a *total exhaust time* of 2^{56} encryptions and an *average solution time* of half that, namely 2^{55} encryptions. Whit Diffie and Martin Hellman argued in 1977 that a DES keysearch machine could be built with a million chips, each testing a million keys a second; as a million is about 2^{20}, this would take on average 2^{15} seconds, or a bit over 9 hours, to find the key. They argued that such a machine could be built for $20 million dollars in 1977 [386]. IBM, whose scientists invented DES, retorted that they would charge the US government $200 million to build such a machine. (Perhaps both were right.)

During the 1980's, there were persistent rumors of DES keysearch machines being built by various intelligence agencies, but the first successful public keysearch attack took place in 1997. In a distributed effort organised over the net, 14,000 PCs took more than four months to find the key to a challenge. In 1998, the Electronic Frontier Foundation (EFF) built a DES keysearch machine called Deep Crack for under $250,000 which broke a DES challenge in 3 days. It contained 1,536 chips run at 40MHz, each chip containing 24 search units which each took 16 cycles to do a test decrypt. The search rate was thus 2.5 million test decryptions per second per search unit, or 60 million keys per second per chip. The design of the cracker is public and can be found at [423]. By 2006, Sandeep Kumar and colleagues at the universities of Bochum and Kiel built a machine using 120 FPGAs and costing $10,000, which could break DES in 7 days on average [755].

Another criticism of DES was that, since IBM kept its design principles secret at the request of the US government, perhaps there was a 'trapdoor' which would give them easy access. However, the design principles were published in 1992 after differential cryptanalysis was invented and published [326]. Their story was that IBM had discovered these techniques in 1972, and the US National Security Agency (NSA) even earlier. IBM kept the design details secret at the NSA's request.

We now have a fairly thorough analysis of DES. The best known *shortcut attack*, that is, a cryptanalytic attack involving less computation than keysearch, is a linear attack using 2^{42} known texts. DES would be secure with more than 20 rounds, but for practical purposes its security is limited by its keylength. I don't know of any real applications where an attacker might get hold of even 2^{40} known texts. So the known shortcut attacks are not an issue. However, its growing vulnerability to keysearch makes DES unusable in its original form. If Moore's law continues, than by 2020 it might be possible to find a DES key on a single PC in a few months, so even low-grade systems such as taxi meters will be vulnerable to brute force-cryptanalysis. As with AES, there are also attacks based on timing analysis and power analysis, but because of DES's structure, the latter are more serious.

The usual way of dealing with the DES keysearch problem is to use the algorithm multiple times with different keys. Banking networks have largely moved to *triple-DES*, a standard since 1999 [936]. Triple-DES does an encryption, then a decryption, and then a further encryption, all done with independent keys. Formally:

$$3DES(k_0, k_1, k_2; M) = DES(k_2; DES^{-1}(k_1; DES(k_0; M)))$$

The reason for this design is that by setting the three keys equal, one gets the same result as a single DES encryption, thus giving a backwards compatibility mode with legacy equipment. (Some banking systems use *two-key triple-DES* which sets $k_2 = k_0$; this gives an intermediate step between single and triple DES). New systems now use AES as of choice, but banking systems are deeply committed to using block ciphers with an eight-byte block size, because of the message formats used in the many protocols by which ATMs, point-of-sale terminals and bank networks talk to each other, and because of the use of block ciphers to generate and protect customer PINs (which I discuss in Chapter 10). Triple DES is a perfectly serviceable block cipher for such purposes for the foreseeable future.

Another way of preventing keysearch (and making power analysis harder) is *whitening*. In addition to the 56-bit key, say k_0, we choose two 64-bit whitening keys k_1 and k_2, xor'ing the first with the plaintext before encryption and the second with the output of the encryption to get the ciphertext afterwards. This composite cipher is known as DESX, and is used in the Win2K encrypting file system. Formally,

$$DESX(k_0, k_1, k_2; M) = DES(k_0; M \oplus k_1) \oplus k_2$$

It can be shown that, on reasonable assumptions, DESX has the properties you'd expect; it inherits the differential strength of DES but its resistance to keysearch is increased by the amount of the whitening [717]. Whitened block ciphers are used in some applications.

The structure of the *f*-function DES is shown in Figure.

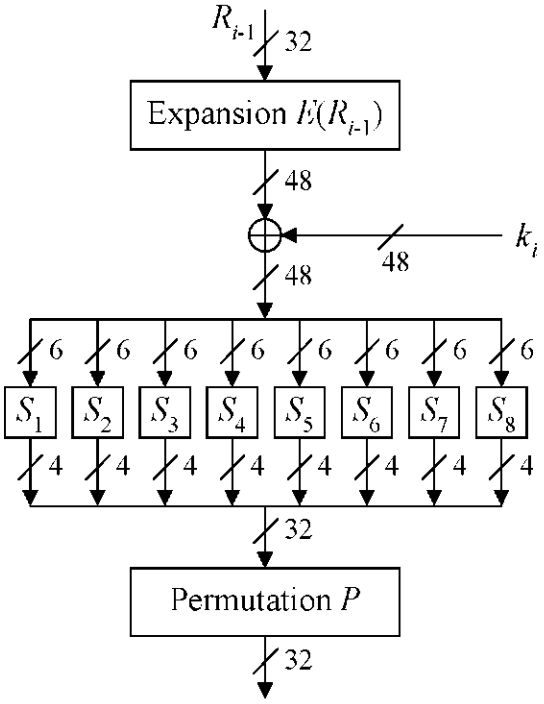

Figure Block diagram of the f-function

Encryption can be expressed by the equations:

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f\left(R_{i-1}, k_i\right) \end{cases}, i = 1,2,...,15;$$
$$\begin{cases} L_{16} = L_{15} \oplus f\left(R_{15}, k_{15}\right) \\ R_{16} = R_{15} \end{cases} \quad (1)$$

Decryption can be expressed by the equations:

$$\begin{cases} L_{i-1} = R_i \\ R_{i-1} = L_i \oplus f\left(R_i, k_i\right) \end{cases}, i = 16,15,...,2;$$
$$\begin{cases} L_0 = L_1 \oplus f\left(R_1, k_1\right) \\ R_0 = R_1 \end{cases} \quad (2)$$