# 1 Introduction

*Iterated cryptosystems* are a family of cryptographically strong functions based on iterating a weaker function $n$ times. Each iteration is called a *round* and the cryptosystem is called an *n-round cryptosystem*. The strong functions are called *block ciphers* and the weak functions are called *round functions*. A block cipher is a function that maps n-bits plaintext blocks to n-bit ciphertext blocks. Block ciphers are either symmetric-key or public-key. Here we consider symmetric-key block ciphers.

An early proposal for an iterated cryptosystem was Lucifer [6], which was designed at IBM to resolve the growing need for data security in its products. The *Data Encryption Standard* (DES) [8] is an improved version of Lucifer. It was also developed at IBM and it was adopted by the U.S. National Bureau of Standards - NBS (today this department is called National Institute of Standards and Technology - NIST) as the standard cryptosystem for sensitive but unclassified data (such as financial transactions and email messages). The standard became effective in July 1977. It was reaffirmed in 1983, 1988, 1993 and 1999.

DES has become the most widely used cryptosystem in the world. Thus, a lot of research has focussed on attacking DES. Some techniques were developed that successfully attacked DES (such as *differential cryptanalysis* [2], *linear cryptanalysis* [7] and *exhaustive search* [5]). These techniques required a lot of computational power (mostly hardware). At first these attacks were considered to be impractical due to the lack of such computational power. But with the increased technological progress, especially with the wide use of parallel machines, DES started to become less secure. Thus, in 1999, when DES was last reaffirmed, it was expanded to *triple* DES (TDEA: Triple Data Encryption Algorithm) [1] a more secure version of DES. It is anticipated that TDEA and the *Advance Encryption Standard* (AES)[1] will coexist as FIPS (Federal Information Processing Standards) approved algorithms allowing for a gradual transition to AES. The latter is intended to provide strong cryptographic security for the protection of sensitive information well into the $21^{st}$ century.

In this paper we will describe DES and briefly present TDEA. Furthermore, we will describe an attack (*differential cryptanalysis*) that can be applied not only to DES but to many DES-like iterated cryptosystems (such as Lucifer, FEAL [12] and GDES [11]). This is a chosen plaintext attack which uses only the resultant ciphertexts. The basic tool of the attack is the *ciphertext pair*: a pair of ciphertexts whose plaintexts have particular differences (thus the name differential). The two plaintexts can be chosen at random, as long as they satisfy the difference condition, and the cryptanalyst does not have to know their values. The attack is statistical in nature and can fail in rare instances.

The rest of the paper is organized as follows: In Section 2 we describe DES encryption /decryption. We also give a brief description of TDEA. In Section 3 we define differential cryptanalysis and we illustrate how to attack DES by giving an example.

[1] AES is a new symmetric-based encryption standard under development by NIST

# 2 The Data Encryption Standard (DES)

In Section 2.1 we give the description of DES. In Section 2.2 we summarize the four different DES modes of operation. Finally, in Section 2.3 we briefly describe the triple DES (TDEA).

## 2.1 Description of DES

A complete description of DES can be found in the FIPS Publication 46 [8]. DES encrypts a plaintext bitstring $x$ of length 64 (64 binary digits of 0's and 1's) using a 64-bit[2] key-string $K$, obtaining a ciphertext bitstring of length 64. Decryption is accomplished by using the same key $K$ as for encryption, but with the schedule of addressing the key bits altered so that the decrypting process is the reverse of the encrypting process (we will formally explain this later).

A plaintext to be encrypted is subjected to a fixed *initial permutation* $IP$, then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation $IP^{-1}$. The key-dependent computation is defined in terms of a function $f$, called the *cipher function*, and of a *key schedule* derived from the key $K$. The cipher function $f$ is given in terms of primitive functions which are called *S-Boxes*[3], denoted $S_i$ $(1 \leq i \leq 8)$, and the *permutation function* $P$.

### 2.1.1 Description of DES Encryption

We will first give a high level description of the algorithm of DES encryption, then we will define the cipher function $f$ along with the S-boxes and the permutation function $P$. Finally we will present the key schedule of DES.

Before we give the DES encryption algorithm we give a convenient notation that we will use in this paper: Given two bitstrings $L$ and $R$, $LR$ denotes the bitstring consisting of the bits of $L$ followed by the bits of $R$. Since concatenation is associative, $B_1B_2 \ldots B_8$ for example denotes the bitstring consisting of the bits of $B_1$ followed by the bits of $B_2 \ldots$ followed by the bits of $B_8$.

**I. DES Encryption Algorithm**

The algorithm proceeds in three stages:

1. Given a 64 bit plaintext $x$, a bitstring $x_0$ of length 64 is constructed by permuting the bits of $x$ according to the initial permutation $IP$. That is, $x_0 = IP(x) = L_0R_0$, where $L_0$ consists the first 32 bits of $x_0$ and $R_0$ consists the last 32 bits of $x_0$.

2. Using $L_0$ and $R_0$, the computation proceeds in a 16-iteration loop ($1 \leq i \leq 16$):

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

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{2}$$

[2]Although $K$ is a bitstring of length 64, only 56 bits are randomly generated and used directly by the algorithm. The other 8 bits, which are not used by the algorithm, may be used for error detection. The 8 error detecting bits are set to make the parity of each 8-bit byte of the key odd. Thus, the bits in positions 8,16,24,32,40,48,56 and 64 are defined so that each byte contains an odd number of 1's. A single error can be detected within each group of 8 bits.

[3]They are also called the *selection functions*.

Each $K_i$ is a bitstring of length 48 derived by permuting a selection of bits of $K$. $K_1, K_2, \ldots, K_{16}$ comprise the key schedule which is formally described in Subsection III.
$f$ is the cipher function which operates on two bitstrings, one of length 32 ($R_{i-1}$) and one of length 48 ($K_i$), and produces a bitstring of length 32 ($R_i$). $f$ is formally defined in the next subsection.
$\oplus$ denotes the exclusive-or of two bitstrings (the bit by bit addition modulo 2).
One iteration (round) of the algorithm is depicted in Figure 1.

3. Take the bitstring $R_{16}L_{16}$ (computed in the previous stage of the algorithm) and permute its bits according to the inverse permutation $IP^{-1}$. The result is the ciphertext $y$. That is, $y = IP^{-1}(R_{16}L_{16})$.

The initial permutation $IP$ and its inverse $IP^{-1}$ are given in Table 1. A sketch of the DES encryption is given in Figure 2.

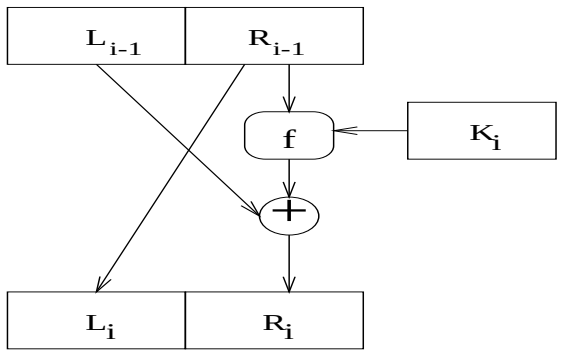


Figure 1: Representation of the $i^{th}$ iteration of the DES algorithm

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

Table 1: The initial permutation $IP$ and its inverse $IP^{-1}$

**Note:** As we mentioned in the Introduction and we can observe from the above algorithm, DES is an iterated cryptosystem. Specifically, since DES operates in 16 rounds, is considered to be a 16-round cryptosystem. The cipher function $f$ is the round function of the cryptosystem. For each $K_i$, $f$ is a bijection on the round of the input.

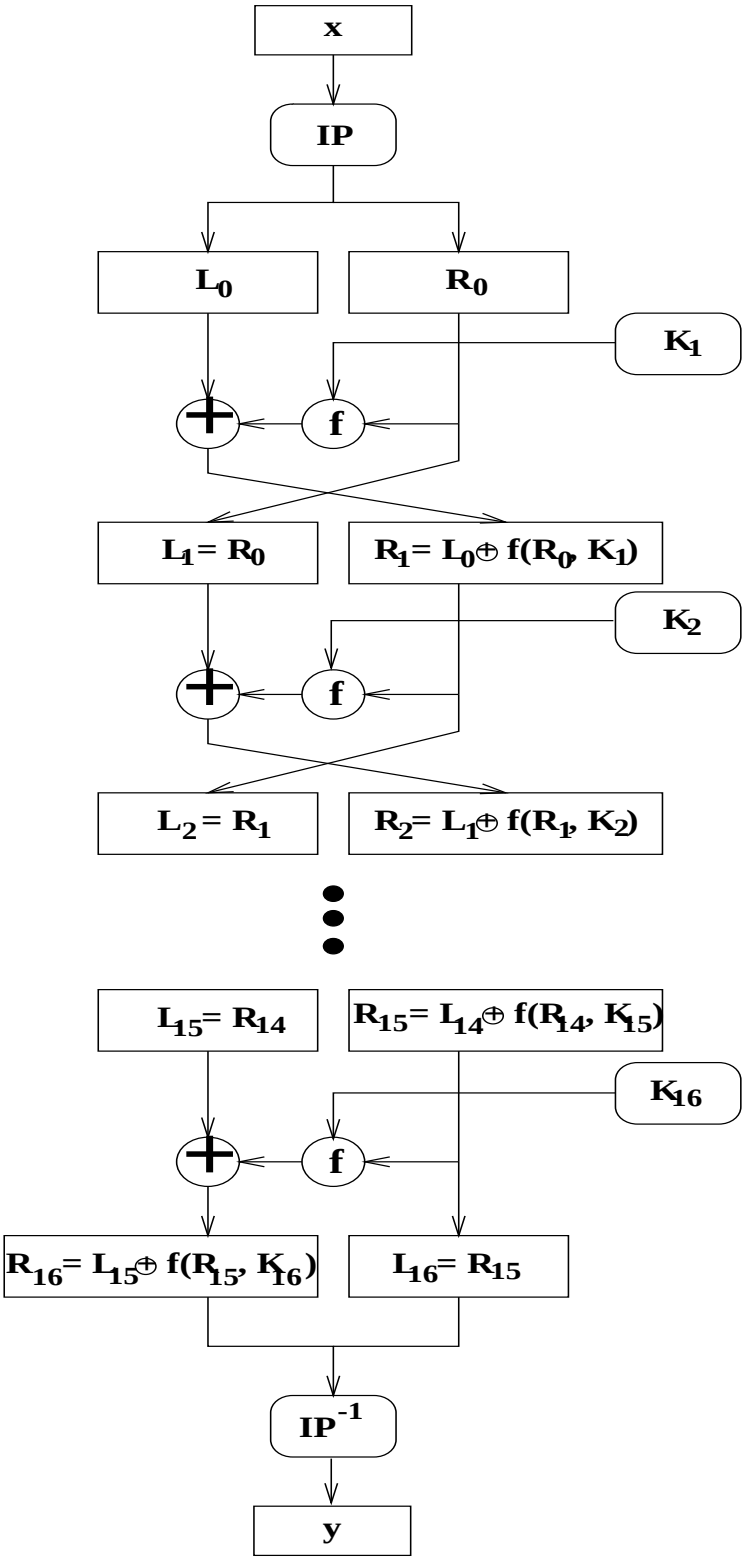


Figure 2: DES Encryption

**II. The Cipher Function $f$**

As we mentioned before, $f$ operates on two bitstrings, one of length 32, called $R$ and one of length 48, called $K$, and produces a bitstring of length 32, called $f(R, K)$.
First, $R$ is expanded from 32 bits to 48 using the fixed expansion function $E$. $E$ is given in Table 2. Basically, $E(R)$ consists of the 32 bits of $R$, permuted in a certain way with 16 of the bits appearing twice. Then, $E(R) \oplus K$ is computed resulting in a bitstring of length 48, call it $B$. Using the notation introduced at the beginning of Section 2.1.1, $B$ can be represented as a concatenation of eight 6-bit strings. That is, $E(R) \oplus K = B = B_1B_2B_3B_4B_5B_6B_7B_8$.
The S-Boxes $S_1, S_2, \ldots, S_8$ are used in the following way: Each $S_i$ is a fixed $4 \times 16$ array whose entries are integers from $0 - 15$. Given a 6-bit string, say $B_j = b_1b_2b_3b_4b_5b_6$, $S_j(B_j)$ is computed as follows: The first and last bits of $B_j$ $(b_1b_6)$ represent in base 2 a number in the range $0 - 3$. Let that number be $r$. $r$ represents a row of $S_j$. The middle bits of $B_j$ $(b_2b_3b_4b_5)$ represent in base 2 a number in the range $0 - 15$. Let that number be $c$. $c$ represents a column of $R_j$. The entry $S_j(r, c)$ is an integer in the range $0 - 15$ which can be represented as a bitstring of length 4. Hence, each $S_j$ is a function that takes as an input two bitstrings of length 2 and 4 and outputs a bitstring of length 4. In this fashion, $C_j = S_j(B_j)$ are computed, where $1 \leq j \leq 8$ (each $C_i$ has length 4). The eight S-boxes are given in Table 3.
Now, consider $S_1$. If $B_1 = 011011$, then $b_1b_6 = 01$, hence $r = 1$ and $b_2b_3b_4b_5 = 1101$, hence $c = 13$. The entry $S_1(1, 13) = 5$ and hence $C_1 = 0101$.
After we compute all $C_i$'s, the resulting 32-bit string $C = C_1C_2C_3C_4C_5C_6C_7C_8$ is permuted according to a fixed permutation $P$. $P$ is given in Table 2. This gives as $f(R, K)$. That is, $f(R, K) = P(C)$. A sketch of the calculation of $f(R, K)$ is given in Figure 3.

| 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 2: The expansion function E and the permutation P

**Note:** The S-boxes are the only part of the cryptosystem that is not linear and thus the security of the cryptosystem crucially depends on their choice. The design criteria of the S-boxes are not completely known. Some rumors were spread arguing that the S-boxes might contain trapdoors which would allow the National Security Agency to decrypt messages while maintaining that DES is secure. However, no evidence became known to the public sustaining this rumor. The design principles of the S-boxes that are known to the public and appeared in [4] are:

- Each row of each S-box is a permutation of the integers $0, \ldots, 15$.

| $S_1$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

| $S_2$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |

| $S_3$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |

| $S_4$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |

| $S_5$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |

| $S_6$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |

| $S_7$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |

| $S_8$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

Table 3: The eight S-boxes

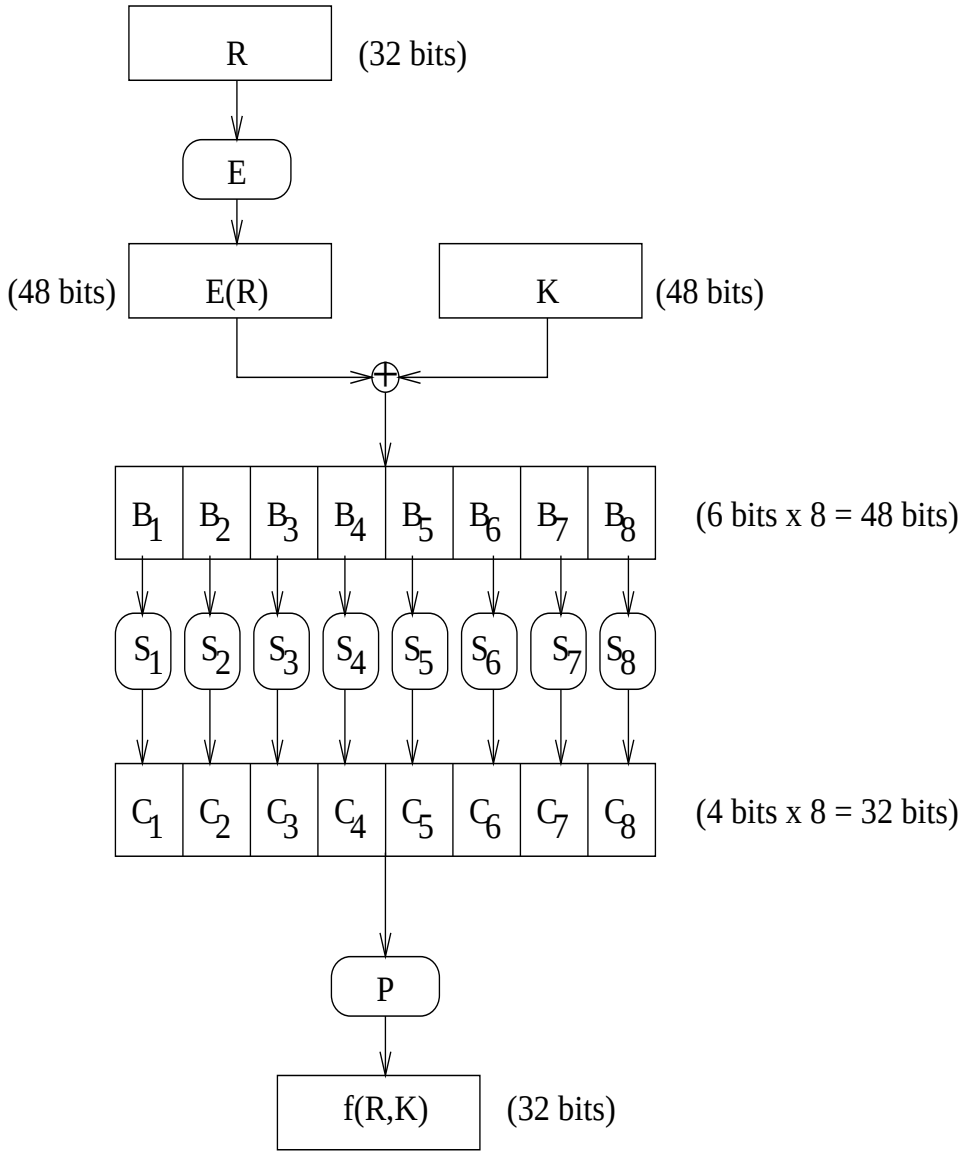


Figure 3: The calculation of $f(R, k)$

- No S-box is a linear or affine function of its input.
- Changing one input bit to an S-box results in changing at least two output bits.
- For any bitstring $x$, $S(x)$ and $S(x \oplus 001100)$ must differ in at least two bits[4].
- For any $x$ and $e, f \in \{0, 1\}$, $S(x) \neq S(x \oplus 11ef00)$.
- The S-boxes were chosen to minimize the differences between the number of 1's and 0's in any S-box output when any single bit is held constant.

Some attempts were made in modifying the DES S-boxes. It was shown that a minimal change of one entry in one of the S-boxes can make DES easier to break. DES with random S-boxes is also very easy to break. This makes the "mysterious" design of the S-boxes even more impressive.

**III. The Key Schedule of DES**

Recall that each $K_i$ ($1 \leq i \leq 16$) used in (2) of the algorithm is a bitstring of length 48 derived by permuting a selection of bits of the key $K$. These $K_i$'s comprise the key schedule of DES. In this section we show how to compute these $K_i$'s. Recall that although the key $K$ is a bitstring of length 64, only 56 bits are used in the algorithm. The other 8 bits are used for parity check (see footnote 3 in the Section 2.1, page 2). The computation of the key schedule is as follows:

- the 56 bits of $K$[5] are permuted according to a fixed permutation PC-1 (see Table 4). Using the notation presented at the beginning of Section 2.1.1, PC-1$(K) = C_0 D_0$, where $C_0$ consists of the first 28 bits of PC-1(K) and $D_0$ consists of the last 28 bits of PC-1$(K)$.
- using $C_0$ and $D_0$, the computation proceeds in a 16-iteration loop ($1 \leq i \leq 16$):

$$\begin{aligned} C_i &= LS_i(C_{i-1}) \\ D_i &= LS_i(D_{i-1}) \\ K_i &= \text{PC-2}(C_i D_i) \end{aligned}$$

  where each $LS_i$ represents a cyclic shift **to the left**[6] of either one or two positions, depending on the value of $i$ :

$$LS_i = \begin{cases} \text{shift one position}, & i = 1, 2, 9, 16 \\ \text{shift two positions}, & \text{otherwise.} \end{cases}$$

  LC-2 is another fixed permutation which reduces the length of $C_i D_i$ ($1 \leq i \leq 16$) from 56 to 48 (see Table 4), i.e., $|K_i| = |\text{PC-2}(C_i D_i)| = 48$, while $|C_i D_i| = 56$.

The key schedule computation is depicted in Figure 4 . Each $K_i$ used in the $i^{th}$ iteration of the DES algorithm can be found in [13] (pages 76-78).

[4] Here $x$ is a bitstring of length 6.

[5] $1^{st}$ to $7^{th}$, $9^{th}$ to $15^{th}$, $17^{th}$ to $23^{rd}$, $25^{th}$ to $31^{st}$, $33^{rd}$ to $39^{th}$, $41^{st}$ to $47^{th}$, $49^{th}$ to $55^{th}$, $57^{th}$ to $63^{rd}$.

[6] By one left shift is meant a rotation of the bits one place to the left and by two left shifts is meant the rotation of the bits two places to the left. For example, if $a = (1, 2, 3, 4, 5, 6, 7, 8)$, then one-position shift to the left results to $a_1 = (8, 1, 2, 3, 4, 5, 6, 7)$ and two-position shift to the left yields $a_2 = (7, 8, 1, 2, 3, 4, 5, 6)$.

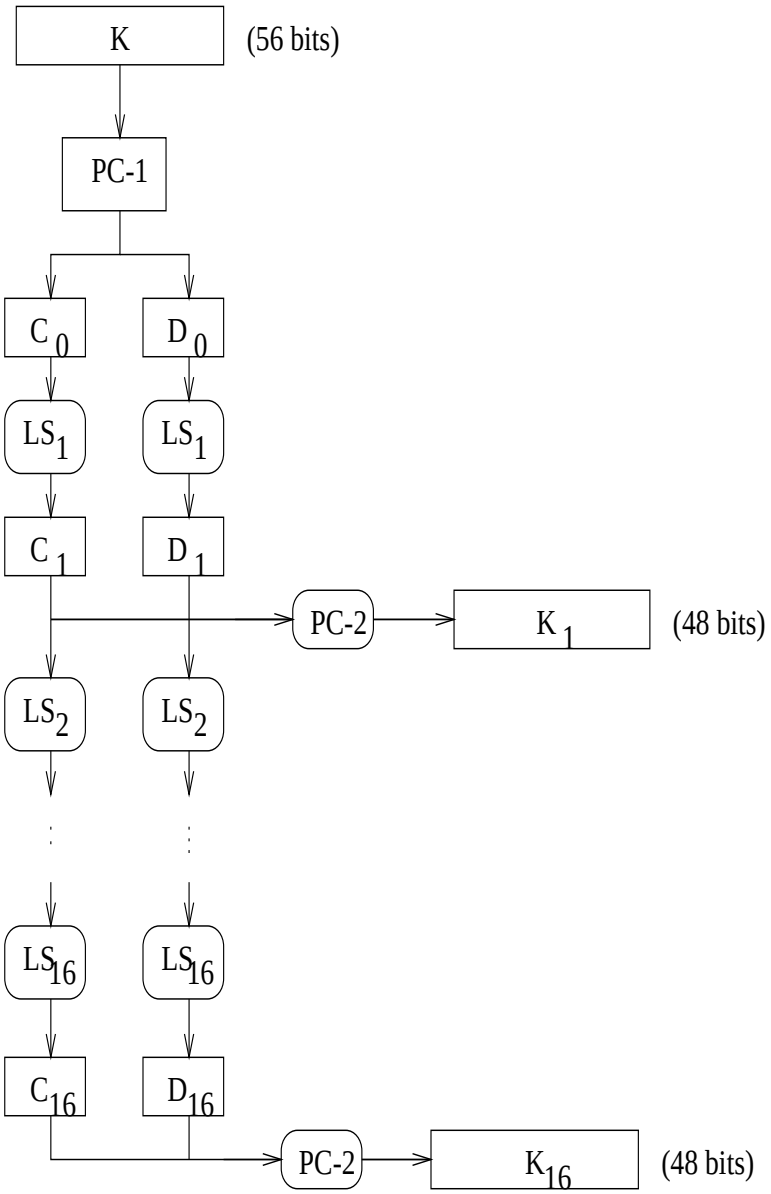


Figure 4: The DES key schedule computation

| PC-1 | | | | | | |
|---|---|---|---|---|---|---|
| 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 |
| 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 |

| PC-2 | | | | | |
|---|---|---|---|---|---|
| 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 4: The permutations PC-1 and PC-2

### 2.1.2 Description of DES Decryption

The decryption is basically done using the same algorithm as encryption, giving as an input the ciphertext $y$, but using the key schedule in the reverse order ($K_{16}, K_{15}, \ldots, K_1$). The resulting output will be the plaintext $x$. Specifically,

1. Given the ciphertext $y = IP^{-1}(R_{16}L_{16})$, then $y_0 = IP(y) = IP(IP^{-1}(R_{16}L_{16})) = R_{16}L_{16}$.
2. Using $R_{16}$ and $L_{16}$, the computation proceeds in a 16-iteration loop starting from $i = 16$ down to $i = 1$:
$$R_{i-1} = L_i$$
$$L_{i-1} = R_i \oplus f(L_i, K_i)$$
3. The resulting bitstring $L_0R_0$ , after is permuted according to the inverse permutation $IP^{-1}$ results to the plaintext $x$. That is, $x = IP^{-1}(L_0R_0)$.

An example of DES encryption/decryption can be found at [13] (pages 79-81).

### 2.1.3 Implementing DES

Someone might think that DES is hard to implement, considering how lengthy its description is. On the contrast, DES can be implemented very efficiently, both in hardware and software. The S-Boxes, the expansion function $E$, the permutations $IP$, $IP^{-1}$ and $P$, and the computation of $K_1, \ldots, K_{16}$ can all be done in constant time by table look up in software, or by hard-wiring them into a circuit in hardware. The only operations that are needed to be performed are the exclusive-or of bitstrings.

Current hardware implementations can attain extremely fast encryption rates. For example, using VLSI chips, you can get over a Gbit/sec throughput. Since 1991, there were more than 45 hardware and software implementations of DES that have been validated by NBS. For more information on DES implementations, the reader is referred to [9].

## 2.2 DES Modes of Operation

In [10] are described four different modes for using the DES algorithm (Section 2.1.1(I)). These four modes are the following:

- Electronic CodeBook mode (ECB): Is a direct application of the DES algorithm to encrypt and decrypt data.
- Cipher Block Chaining mode (CBC): Is an enhanced mode of ECB which chains together blocks of ciphertext.
- Cipher FeedBack mode (CFB): It uses previously generated ciphertext as input to DES to generate pseudorandom outputs which are combined with the plaintext to produce cipher, thereby chaining together the resulting cipher.
- Output FeedBack mode (OFB): Is identical to CFB except that the previous output of DES is used as input in OFB while the previous cipher is used as input in CFB. OFB does not chain the cipher.

More information about the DES modes can be found at [10] and [13] (pages 83-86).

## 2.3 Triple DES (TDEA)

Breaking a DES encrypted ciphertext by trying all possible keys (exhaustive search) has become increasingly more feasible with technology advances. Thus, DES has been extended to triple DES (TDEA: Triple Data Encryption Algorithm). A TDEA key consists of three DES keys, making the exhaustion attack more difficult and impractical. A complete description of TDEA can be found at the American National Standards Institute (ANSI), publication X9.52 [1].

### I. TDEA Encryption/Decryption

We now describe TDEA encryption/decryption. Let $E_K(x)$ and $D_K(y)$ represent the DES encryption and decryption of a plaintext $x$ and its corresponding ciphertext $y$, using the DES key $K$ respectively. Each TDEA encryption/decryption operation is a compound operation of DES encryption and decryption operations:

- TDEA encryption operation: $y = E_{K_3}(D_{K_2}(E_{K_1}(x)))$
- TDEA decryption operation: $x = D_{K_1}(E_{K_2}(D_{K_3}(y)))$

The standard specifies the following key options:

1. $K_1, K_2$ and $K_3$ are independent keys
2. $K_1$ and $K_2$ are independent keys and $K_3 = K_1$
3. $K_1 = K_2 = K_3$

Figure 5 illustrates the TDEA encryption and TDEA decryption.

### II. TDEA Modes of Operation

The X9.52 standard [1] describes seven different modes for using TDEA:

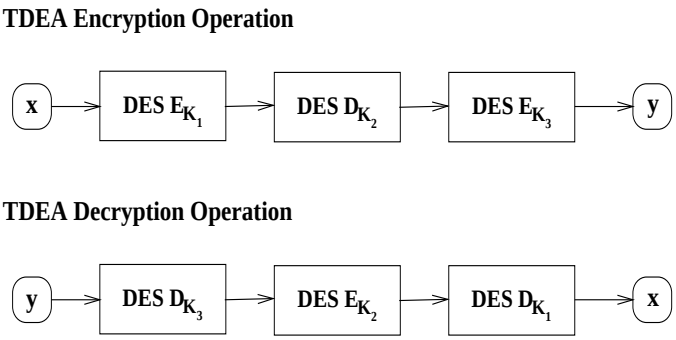


Figure 5: TDEA Encryption/Decryption

- TDEA Electronic CodeBook mode (TECB)
- TDEA Cipher Block Chaining mode (TCBC)
- TDEA Cipher Block Chaining mode - Interleaved (TCBC-I)
- TDEA Cipher FeedBack mode (TCFB)
- TDEA Cipher FeedBack mode - Pipelined (TCFB-P)
- TDEA Output FeedBack mode (TOFB)
- TDEA Output FeedBack mode - Interleaved (TOFB-I)

TECB,TCBC,TCFB and TOBF are based upon ECB,CBC,CFB and OFB respectively obtained by substituting the DES encryption/decryption operation with the TDEA encryption/decryption operation. A TDEA mode of operation is said to be *backward compatible* with its single DES counterpart if, with compatible key options for TDEA operation,

1. an encrypted plaintext computed using a single DES mode can be decrypted by a corresponding TDEA mode;
2. an encrypted plaintext computed using a TDEA mode can be decrypted by a corresponding single DES mode;

An immediate observation is that when using the key option 3 ($K_1 = K_2 = K_3$), TECB, TCBC, TCFB and TOFB modes are backward compatible with single DES modes ECB, CBC, CFB and OFB respectively.

# 3 Differential Cryptanalysis of DES

*Cryptanalysis* is the art of analyzing ciphertext to extract the plaintext or the key. In other words, cryptanalysis is the opposite of cryptography. DES has been subject to cryptanalytic attacks since its adoption in 1977. Practical cryptanalysis of DES focuses on key-recovery.

The most obvious attack is the *exhaustive search*, i.e., breaking a DES encrypted ciphertext by trying all possible keys (the key space is $2^{56}$). Diffie and Hellman [5] suggested exhaustive search of the entire key space on a parallel machine. They estimate that a VLSI chip may be

built which can search one key every microsecond. A machine with a million such chips, all searching in parallel, could search the entire key space in about a day. However, they estimated that such a machine could be build for about $20,000,000 and the cost per solution to be $5000. In 1994, Michael Wiener gave a detailed design of a $1,000,000 machine that does exhaustive key search in about 3.5 hours [14]. More recently, a DES key search machine was actually built, as a cost of $250,000. It finds the key in 56 hours (about 2.5 days). It is just matter of time: soon a cheap machine will be able to break DES very quickly.

Assuming that obtaining enormous numbers of known-plaintext pairs is feasible, *linear cryptanalysis* [7] provides the most powerful attack on DES to date; but it is not considered a threat to DES in practical environments. It is a *known-plaintext attack*: $2^{43}$ plaintext-ciphertext pairs are available to the adversary that uses them to find the key. Linear cryptanalysis improved *differential cryptanalysis*, first introduced by Biham and Shamir [2]. Differential cryptanalysis is a *chosen-plaintext attack*: $2^{47}$ plaintext-ciphertext pairs are available of the adversary's choice, who uses them to find the key. Differential cryptanalysis is less realistic in practice than linear cryptanalysis. Actually, both of them are considered impractical, since it is difficult to obtain $2^{43}$ or more known plaintexts pairs and their corresponding ciphertext pairs.

However, differential cryptanalysis is considered as major breakthrough in cryptanalysis. It is one of the most general cryptanalytic tools to date against modern iterated cryptosystems, including DES, Lucifer [6], GDES (Generalized DES Scheme) [11] and FEAL (Fast Encryption Algorithm) [12]. In Section 3.1 we formally define differential cryptanalysis and we give the main idea behind it. In Section 3.2 we introduce the important concept of the *characteristic* and in Section 3.3 we illustrate differential cryptanalysis by giving an example.

## 3.1 Differential Cryptanalysis

The basic idea of differential cryptanalysis is to first encrypt some plaintext, then make particular changes in that plaintext and encrypt it again. Per [2], a pair of such ciphertexts whose plaintexts have particular differences is called a *ciphertext pair*. Particular ciphertext differences occur more frequently with some key values than others, so when those differences occur, particular keys are indicated. With huge number of tests, false indications will be distributed randomly, but true indications always point at the same key values and so will eventually indicate some part (in some cases in whole) the key. The basic concept can be applied to virtually any sort of statistic which relates ciphertext changes to key values. But because the probabilities involved are generally quite small, success generally depends upon having very substantial amounts of known-plaintext.

We now present the definition of the differential cryptanalysis, as appeared in [2].

**Definition 1** Differential Cryptanalysis *is a method which analyzes the effect of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs. These differences can be used to assign probabilities to the possible keys and to locate the most probable key. This method usually works on many pairs of plaintexts with the same particular difference using only the resultant ciphertext pairs. For DES-like cryptosystems the difference is chosen as a fixed XORed value of two plaintexts.*

To attack the usual 16-round DES, the key is computed using $2^{47}$ chosen plaintexts in $2^{37}$ time. This attack is considered to be impractical, not as much as for its complexity time, but

mainly for the huge number of chosen plaintexts that are needed. However, it is practical, when the number of rounds of encryption is reduced. For instance, the 8-round DES can be broken in only a couple of minutes on an ordinary PC and it uses $2^{14}$ chosen plaintexts. The performance of differential cryptanalysis for various numbers of rounds is summarized in Table 5.

| Rounds | Chosen Plaintexts | Complexity of Analysis |
|---|---|---|
| 8 | $2^{14}$ | $2^{9}$ |
| 9 | $2^{24}$ | $2^{32}$ |
| 10 | $2^{24}$ | $2^{15}$ |
| 11 | $2^{31}$ | $2^{32}$ |
| 12 | $2^{31}$ | $2^{21}$ |
| 13 | $2^{39}$ | $2^{32}$ |
| 14 | $2^{39}$ | $2^{29}$ |
| 15 | $2^{47}$ | $2^{37}$ |
| 16 | $2^{47}$ | $2^{37}$ |

Table 5: Attacks on various numbers of rounds of DES

For the purposes of the attack, DES is considered restricted to $n$ rounds, where $n \leq 16$. The initial permutation $IP$ and its inverse $IP^{-1}$ can be ignored, since they do not affect the cryptanalysis. Thus, $L_0R_0$ is regarded as the plaintext, and $L_nR_n$ as the ciphertext. Note that $L_nR_n$ is not inverted since the generality of the attack is not affected. As we have mentioned before, the attack involves comparing the exclusive-or (x-or) of two plaintexts to the x-or of the corresponding two ciphertexts (cipher pair).

**Definition 2** *Given two bitstrings $X$ and $X^*$ ($|X| = |X^*|$), we denote $X'$ to be the two strings' x-or, i.e., $X' = X \oplus X^*$.*

Following Definition 2, $L_0'R_0' = L_0R_0 \oplus L_0^*R_0^*$ is the x-or of the plaintexts $L_0R_0$ and $L_0^*R_0^*$. In the following definition, $S_j$ denotes an S-box ($1 \leq j \leq 8$).

**Definition 3** *Let $(B_j, B_j^*)$ be an ordered pair of 6-bit strings. We say that the* input x-or *of $S_j$ is $B_j \oplus B_j^*$ and the* output x-or *of $S_j$ is $S_j(B_j) \oplus S_j(B_j^*)$.*

From Definition 3 and from Section 2.1.1(II), it follows that an input x-or is a 6-bit string and an output x-or is a 4-bit string.

**Definition 4** *For any $B_j' \in \{0,1\}^6$, define the set $\Delta(B_j')$ to consist all the ordered pairs $(B_j, B_j^*)$ that have input x-or $B_j'$.*

Note that, $B_j' = B_j \oplus B_j^* \Rightarrow B_j^* = B_j \oplus B_j'$. Thus, $\Delta(B_j') = \{(B_j, B_j \oplus B_j') : B_j \in \{0,1\}^6\}$ **(a)**. $\Delta(B_j')$ contains $2^6 = 64$ pairs. For each such pair, we can compute the output x-or of $S_j$ ($1 \leq j \leq 8$) and write down the resulting distribution. There are 64 output x-ors which are distributed among $2^4 = 16$ possible values (output x-or is a 4-bit string). These distributions most of the time are non-uniform. This non-uniformity is the basis for the attack.

**Definition 5** *Given an input x-or $B'_j$ and an output x-or $C'_j$ $(1 \leq j \leq 8)$, define* $IN_j(B'_j, C'_j) = \{B_j \in \{0,1\}^6 : S_j(B_j) \oplus S_j(B_j \oplus B'_j) = C'_j\}$ *and* $N_j(B'_j, C'_j) = |IN_j(B'_j, C'_j)|$.

Informally speaking, $N_j(B'_j, C'_j)$ counts the number of pairs with the input x-or $B'_j$ which have output x-or equal to $C'_j$ for the S-box $S_j$. The actual pairs can be obtained from the set $IN_j(B'_j, C'_j)$. For each of the eight S-boxes, there are 64 possible input x-ors. Thus, there are 512 distributions which can be computed (easily by a 'small' computer).

We now give an example to help the reader understand better the terms given above.

**Example 1** Consider S-box $S_1$ (given in Table 3) and the input x-or $B'_1 = 110100$. From **(a)**, $\Delta(110100) = \{(000000, 110100), (000001, 110101), \ldots, (111111, 001011)\}$. Now, for each ordered pair of $\Delta(110100)$, we have to compute $C_1$, the output x-or of $S_1$. For example, for the first pair $(000000, 110100)$ we have $C'_1(000000, 110100) = S_1(000000) \oplus S_1(110100) = 1110 \oplus 1001 = 0111$. We do that for all 64 pairs in $\Delta(110100)$ and we get the distribution of output x-ors as shown in Table 6. This distribution consists of the values $N_1(110100, C'_1)$, i.e., $N_1(110100, 0000) = 0, N_1(110100, 0001) = 8, \ldots, N_1(110100, 1111) = 6$. The sets $IN_1(110100, C'_1)$ are listed in Table 7.
Observe that only 8 of the 16 possible output x-ors occur. This example has a very nonuniform distribution. In general, on average, 75-80% of the possible output x-ors actually occur. □

| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
|---|---|---|---|---|---|---|---|
| 0 | 8 | 16 | 6 | 2 | 0 | 0 | 12 |
| 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 6 | 0 | 0 | 0 | 0 | 8 | 0 | 6 |

Table 6: Distribution of $\Delta(110100) \equiv N_1(110100, C'_1)$

From Section 2.1.1 we know that the input to the S-boxes in round $i$ is formed as $B_i = E(R_{i-1}) \oplus K_i$. For simplicity of notation, we write:

$$B = E \oplus J \quad \textbf{(b)}$$

where $B = B_i$, $E = E(R_{i-1})$ and $K_i = J$. The input x-or for all eight S-boxes in round $i$ can be computed as follows:

$$B \oplus B^* = (E \oplus J) \oplus (E^* \oplus J) = E \oplus E^* = E' \quad \textbf{(c)}$$

Hence, the input x-or does not depend on $J$ (some permuted bits of the key $K$).

We write each $B$, $B^*$, $E$, $E^*$ and $J$ as the concatenation of eight 6-bit strings. I.e., $B = B_1B_2B_3B_4B_5B_6B_7B_8$, $B^* = B^*_1B^*_2B^*_3B^*_4B^*_5B^*_6B^*_7B^*_8$, $E = E_1E_2E_3E_4E_5E_6E_7E_8$, $E^* = E^*_1E^*_2E^*_3E^*_4E^*_5E^*_6E^*_7E^*_8$ and $J = J_1J_2J_3J_4J_5J_6J_7J_8$. Let $C'_j = S_j(B_j) \oplus S_j(B^*_j)$ $(1 \leq j \leq 8)$. From **(b), (c)** and Definition 5 it follows that: $E_j \oplus J_j \in IN_j(E'_j, C'_j)$ **(d)**.

**Definition 6** *Let $E_j$ and $E^*$ be 6-bit strings, and $C'_j$ be a 4-bit string. We define $test_j(E_j, E^*_j, C'_j)$ to be the set of all x-ors of $E_j$ with every element of $IN_j(E'_j, C'_j)$. I.e., $test_j(E_j, E^*_j, C'_j) = \{B_j \oplus E_j : B_j \in IN_j(E'_j, C'_j)\}$*

**Theorem 1** *Let $E_j$ and $E^*_j$ be two inputs to the S-box $S_j$, and $C'_j$ be the output x-or for $S_j$. Then, the key bits $J_j$ occur in the set $test_j(E_j, E^*_j, C'_j)$.*

| output x-or | possible inputs |
|---|---|
| 0000 | |
| 0001 | 000011, 001111, 011110, 011111, 101010, 101011, 110111, 111011 |
| 0010 | 000100, 000101, 001110, 010001, 010010, 010100, 011010, 011011<br>100000, 100101, 010110, 101110, 101111, 110000, 110001, 111010 |
| 0011 | 000001, 000010, 010101, 100001, 110101, 110110 |
| 0100 | 010011, 100111 |
| 0101 | |
| 0110 | |
| 0111 | 000000, 001000, 001101, 010111, 011000, 011101, 100011, 101001<br>101100, 110100, 111001, 111100 |
| 1000 | 001001, 001100, 011001, 101101, 111000, 111101 |
| 1001 | |
| 1010 | |
| 1011 | |
| 1100 | |
| 1101 | 000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010 |
| 1110 | |
| 1111 | 000111, 001010, 001011, 110011, 111110, 111111 |

Table 7: Possible inputs for input x-or 110100

**Proof:** Per Definition 6, $test_j(E_j, E_j^*, C_j')$ is the set of all x-ors of $E_j$ with every element of $IN_j(E_j', C_j')$. From **(d)** we know that $E_j \oplus J_j \in IN_j(E_j', C_j')$. Thus, $E_j \oplus (E_j \oplus J_j) \in test_j(E_j, E_j^*, C_j')$. But $E_j \oplus (E_j \oplus J_j) = J_j$. Hence $J_j \in test_j(E_j, E_j^*, C_j')$. □

From the above theorem and from Definitions 5 and 6 it follows that there will be exactly $N_j(E_j', C_j')$ 6-bit strings in $test_j(E_j, E_j^*, C_j')$; the correct value of $J_j$ must be one of these possibilities.

**Example 2** Let $E_1 = 000001$, $E_1^* = 110101$ and $C_1' = 1101$. Hence, $E_1' = 110100$. From Table 6 we get that $N_1(110100, 1101) = 8$. Thus, there will be exactly eight bit-strings in $test_1(000001, 110101, 1101)$. From Table 7 we get that $IN_1(110100, 1101) = \{000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010\}$. When we x-or 000001 with every element of $IN_1(110100, 1101)$, we get: $test_1(000001, 110101, 1101) = \{000111, 010001, 010111, 011101, 100011, 100101, 101001, 110011\}$. □

We have seen that from a triple $E_1, E_1^*, C_1'$ we can obtain a set $test_1$ of possible values for the keybits in $J_1$. If we have a second such triple, then we will obtain a second set $test_1$ of possible values for the keybits in $J_1$. The true value of $J_1$ must be in the intersection of both sets. If we have several such triples, then we can easily determine the key bits in $J_1$. We can do that for all $J_j$'s to get $J$. The question now is how we obtain the triples $E_1, E_1^*, C_1'$. We will see that by investigating an attack on the 6-round DES (Section 3.3). Before we do that, we will study the important concept of the probability of the *characteristic* (Section 3.2).

## 3.2 The Probability of the Characteristic

We now define the notion of the characteristic and later we will see its importance.

**Definition 7** *Consider an integer $n \geq 1$. An $n$-*round characteristic *is a list of the form $L'_0, R'_0, L'_1, R'_1, p_1, L'_2, R'_2, p_2, \ldots, L'_{n-1}, R'_{n-1}, p_{n-1}, L'_n, R'_n, p_n$, which satisfies the following:*

1. *$L'_i = R'_{i-1}$ for $1 \leq i \leq n$.*
2. *Let $L_{i-1}, R_{i-1}, L^*_{i-1}$ and $R^*_{i-1}$ $(1 \leq i \leq n)$ be chosen such that $L_{i-1} \oplus L^*_{i-1} = L'_{i-1}$ and $R_{i-1} \oplus R^*_{i-1} = R'_{i-1}$. By applying one round of DES encryption on $L_{i-1}R_{i-1}$ and $L^*_{i-1}R^*_{i-1}$ we get $L_iR_i$ and $L^*_iR^*_i$ respectively. Then, the probability that $L_i \oplus L^*_i = L'_i$ and $R_i \oplus R^*_i = R'_i$ is precisely $p_i$.*

*The* probability of the characteristic *is defined to be the product $p = p_1 \cdot p_2 \cdot \ldots \cdot p_n$.*

**Note:** Say we choose $L_0, R_0$ and $L^*_0R^*_0$ so that $L_0 \oplus L^*_0 = L'_0$ and $R_0 \oplus R^*_0 = R'_0$ and we apply $n$ rounds of DES encryption. We obtain $L_1, \ldots, L_n$, $R_1, \ldots, R_n$, $L^*_1, \ldots, L^*_n$ and $R^*_1, \ldots, R^*_n$. Then we **cannot** claim that the probability that $L_i \oplus L^*_i = L'_i$ and $R_i \oplus R^*_i = R'_i$ for all $i$ $(1 \leq i \leq n)$ is $p_1 \cdot p_2 \cdot \ldots \cdot p_n$. This is because the 48-bits in the key schedule $K_1, \ldots, K_n$ are not mutually independent. However, we consider $p_1 \cdot p_2 \cdot \ldots \cdot p_n$ to be a fairly accurate estimate of this probability.

The following 1-round characteristic is the basis of the attack on a 3-round DES (See [13], pages 93-97). The probability of this characteristic is 1 (the attack on a 3-round DES is deterministic). For brevity, the numbers are presented in hexadecimal form (for example, $10_{16} = 16_{10} = 1000_2$):

| | | |
|---|---|---|
| $L'_0 =$ anything | $R'_0 = 00000000_{16}$ | |
| $L'_1 = 00000000_{16}$ | $R'_1 = L'_0$ | $p = 1$ |

Let's check why $p = 1$. Since $R'_0 = 00000000_{16}$, that means that $R_0 = R^*_0$. From equation (1) of the DES encryption algorithm, it immediately follows that $L_1 = L^*_1$ and hence $L'_1 = 00000000_{16}$. From equation (2) of the DES encryption algorithm it follows that $R'_1 = R_1 \oplus R^*_1 = (L_0 \oplus f(R_0, K_1)) \oplus (L^*_0 \oplus f(R^*_0, K_1))$. But $f(R_0, K_1) = f(R^*_0, K_1)$ and hence $R'_1 = L_0 \oplus L^*_0 = L'_0$.

The following 1-round characteristic has probability $p = \frac{14}{64} = \frac{1}{4}$.

| | | |
|---|---|---|
| $L'_0 = 00000000_{16}$ | $R'_0 = 60000000_{16}$ | |
| $L'_1 = 60000000_{16}$ | $R'_1 = 00808200_{16}$ | $p = 14/64$ |

The details of this characteristic can be found at [13] (pages 98-99).

The probability of the characteristic used for an attack on an n-round DES ($n > 3$) it gives us the number of chosen plaintexts needed for the attack: $1/p$. Detailed explanations regarding the characteristic used in attacks on the various n-round DES can be found at [2] and [3].

## 3.3 Attack on a 6-round DES

We use the ideas presented in Section 3.1 to describe a probabilistic attack on a 6-round DES. Roughly speaking, the idea is to choose a pair of plaintexts with a specified x-or and then determine the probabilities of a specific sequence of x-ors through the rounds of encryption.

The attack on the 6-round DES is based on the following 3-round characteristic:

| | | |
|---|---|---|
| $L'_0 = 40080000_{16}$ | $R'_0 = 04000000_{16}$ | |
| $L'_1 = 04000000_{16}$ | $R'_1 = 00000000_{16}$ | $p = 1/4$ |
| $L'_2 = 00000000_{16}$ | $R'_2 = 04000000_{16}$ | $p = 1$ |
| $L'_3 = 04000000_{16}$ | $R'_3 = 40080000_{16}$ | $p = 1/4$ |

For the attack, we start with $L_0R_0,\ L_0^*, R_0^*,\ L_6R_6$ and $L_6^*, R_6^*$, chosen in such a way so that $L'_0 = 40080000_{16}$ and $R'_0 = 04000000_{16}$ . We express $R_6$ as follows:

$$\begin{aligned} R_6 &= L_5 \oplus f(R_5, K_6) && \text{(equation (2) of DES algorithm)} \\ &= R_4 \oplus f(R_5, K_6) && (L_5 = R_4 : \text{equation (1) of DES algorithm}) \\ &= L_3 \oplus f(R_3, K_4) \oplus f(R_5, K_6) && (R_4 = L_3 \oplus f(R_3, K_4) : \text{equation (2)}) \end{aligned}$$

Similarly, $R_6^* = L_3^* \oplus f(R_3^*, K_4) \oplus f(R_5^*, K_6)$. Hence,

$$R'_6 = L'_3 \oplus f(R_3, K_4) \oplus f(R_3^*, K_4) \oplus f(R_5, K_6) \oplus f(R_5^*, K_6) \quad (3)$$

Note that $R'_6$ is known, since we know $R_6$ and $R_6^*$. From the characteristic, we estimate that $L'_3 = 04000000_{16}$ and $R'_3 = 40080000_{16}$ with probability $p = \frac{1}{4} \cdot 1 \cdot \frac{1}{4} = \frac{1}{16}$. Recall from Section 3.1 (equation **(c)**) that the input x-or for all eight S-boxes in round 4 is $E(R'_3)$ which is:

| 001000 | 000000 | 000001 | 010000 | 000000 | 000000 | 000000 | 000000 |
|---|---|---|---|---|---|---|---|
| $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ | $S_6$ | $S_7$ | $S_8$ |

Observe that the input x-ors for $S_2,\ S_5,\ S_6,\ S_7$ and $S_8$ are all $000000$ and thus the output x-ors are $0000$ for these five S-boxes in round 4. We can compute the output x-ors of these five S-boxes in round 6 using equation (3). Let $C'_1, C'_2, \ldots, C'_8$ be the 4-bit output x-ors of the S-Boxes in round 6. Recall from Section 2.1.1, that $f(R, K) = P(\text{output of S-boxes})$. Thus, $C'_1C'_2 \ldots C'_8 = P^{-1}(R'_6 \oplus L'_3)$. With probability $1/16$, $C'_2,\ C'_5,\ C'_6,\ C'_7$ and $C'_8$ are the output x-ors of $S_2,\ S_5,\ S_6,\ S_7$ and $S_8$ ,respectively, in round 6. The inputs to these S-boxes in round 6 can be computed to be:

$$E_1E_2E_3E_4E_5E_5E_6E_7E_8 = E(R_5) = E(L_6)$$

$$E_1^*E_2^*E_3^*E_4^*E_5^*E_5^*E_6^*E_7^*E_8^* = E(R_5^*) = E(L_6^*)$$

They can be computed from the ciphertexts, as shown in Figure 6.

Input: $L_0R_0, L_0^*R_0^*, L_6R_6$ and $L_6^*R_6^*$, where $L'_0 = 40080000_{16}$ and $R'_0 = 04000000_{16}$
1. compute $C' = P^{-1}(R'_6 \oplus 04000000_{16})$
2. compute $E = E(L_6)$ and $E^* = E(L_6^*)$
3. for $j \in \{2, 5, 6, 7, 8\}$ do
   compute $test_j(E_j, E_j^*, C'_j)$.

Figure 6: Differential attack on 6-round DES

Proceeding as in Section 3.1, we should be able to find the 30 key bits in $J_2,\ J_5,\ J_6,\ J_7$ and $J_8$. However, the hypothesized output x-or for round 6 is correct only with probability $1/16$.

That means that $15/16$ of the time, we will get just random bits rather than the possible key bits!!! Fortunately our attempts do not stop here. We use the *filtering operation*. Before we explain what is the filtering operation, we define the notions of *right* and *wrong* pairs.

**Definition 8** *Given $L_0' = L_0 \oplus L_0^*$ and $R_0' = R_0 \oplus R_0^*$, the pair of plaintexts $L_0R_0$ and $L_0^*R_0^*$ is said to be a* right pair *with respect to a characteristic if $L_i \oplus L_i^* = L_i'$ and $R_i \oplus R_i^* = R_i'$ for all $i,\ 1 \leq i \leq n$. Otherwise, the pair is said to be a* wrong pair.

For our case, we expect $1/16$ of the pairs to be right pairs and $15/16$ to be wrong pairs with respect to the 3-round characteristic we are using. Recall, that our goal is to compute $E_j, E_j^*$ and $C_j'$ (as in figure 6) and then determine $test_j(E_j, E_j^*, C_j')$, for $j = 2, 5, 6, 7, 8$. If we start with a right pair, then the correct key bits for each $J_j$ are included in $test_j$. Otherwise, the value of $C_j'$ is incorrect and thus $test_j$ will essentially be random. We usually use the filtering operation to identify a wrong pair.

**Filtering Operation:** We check $|test_j|$ for $j = 2, 5, 6, 7, 8$. If for any of the $j's$, $|test_j| = 0$, then we necessarily have a wrong pair.

Given a wrong pair, the probability that $|test_j| = 0$ for some $j$ is about $1/5$. The probability of all five $test_j$'s have positive cardinality is about $0.8^5 \approx 1/3$, so the probability that at least one $test_j$ has zero cardinality is about $2/3$. Thus, the proportion of right pairs that remain after the filtering operation is about $\frac{\frac{1}{16}}{\frac{1}{16} + (\frac{15}{16} \cdot \frac{1}{3})} = \dfrac{1}{6}$.

Now, assume that we have a pair the survives the filtering operation. This does not mean that this pair is necessarily a right pair. We say that the 30-bit string $J_2J_5J_6J_7J_8$ is *suggested* by the pair if $J_j \in test_j$ for all $j = 2, 5, 6, 7, 8$. The number of suggested bitstrings is $\prod_{j \in \{2,5,6,7,8\}} |test_j|$. Note that this number can be large, i.e., greater than $80,000$.

If we tabulate all the suggested bitstrings obtained from the $N$ pairs that were not discarded by the filtering operation, then for every right pair, the correct bitstring $J_2J_5J_6J_7J_8$ will be a suggested bitstring. This correct bitstring will be counted about $3N/16$ times. Incorrect bitstrings occur much less often since they occur essentially at random and there are $2^{30}$ possibilities. Instead of tabulating all the suggested bitstrings, an algorithm is used that requires less space and time (see [13], page 102-103).

So, this method gives us 30 of the 56 key bits. By using a different 3-round characteristic (with the same probability of characteristic), we compute 12 further key bits, namely those in $J_1$ and $J_4$. This new 3-round characteristic is the following:

| | | |
|---|---|---|
| $L_0' = 00200008_{16}$ | $R_0' = 00000400_{16}$ | |
| $L_1' = 00000400_{16}$ | $R_1' = 00000000_{16}$ | $p = 1/4$ |
| $L_2' = 00000000_{16}$ | $R_2' = 00000400_{16}$ | $p = 1$ |
| $L_3' = 00000400_{16}$ | $R_3' = 00200008_{16}$ | $p = 1/4$ |

The 14 remaining unknown key bits can be computed using exhaustive search, since $2^{14} = 16384$ is quite small.

An example that illustrates the attack on a 6-round DES is given in [13] (pages 103 -109). More details about the attacks on the various n-round DES can be found at [2] and [3].

## References


[1] American National Standards Institute, *Triple Data Encryption Algorithm Modes of Operation*, ANSI X9.52, 1998

[2] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *Advances in Cryptology – CRYPTO'90*, Springer-Verlag, 2–21, 1990.

[3] E. Biham and A. Shamir, "Differential Cryptanalysis of the Full 16-round DES," *Advances in Cryptology – CRYPTO'92*, Springer-Verlag, 487–496, 1992.

[4] E. F. Brickell, J .H. Moore and M. R. Purtill, "Structure in the S-Boxes of DES," *Advances in Cryptology –CRYPTO'86*, 3–7, 1986.

[5] W. Diffie and M. E. Hellman, "Exhaustive Cryptanalysis of the NBS DES," *Computer*, Vol. 10, No. 6, 74–84, 1977.

[6] H. Feistel, "Cryptography and Data Security," *Scientific American*, Vol. 228, No. 5, 15–23, 1973.

[7] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," *Advances in Cryptology – EUROCRYPT'93*, 386–397, 1994.

[8] National Bureau of Standards, *Data Encryption Standard*, U.S. Department of Commerce, FIPS pub. 46, January 1977.

[9] National Bureau of Standards, *Guidelines for Implementing and Using the NBS DES*, U.S. Department of Commerce, FIPS pub. 74, April 1981.

[10] National Bureau of Standards, *DES Modes of Operation*, U.S. Department of Commerce, FIPS pub. 81, December 1980.

[11] I. Schaumuller-Bichl, "On the Design and Analysis of New Cipher Systems Related to DES," Technical Report, Linz University, 1983.

[12] A. Shimizu and S. Miyaguchi, "Fast Data Encryption Algorithm FEAL," *Advances in Cryptology – EUROCRYPT'87*, VII 11–14, 1987.

[13] D. R. Stinson, *CRYPTOGRAPHY: Theory and Practice*, CRC Press LLC, 1995.

[14] M. J. Wiener, "Efficient DES Key Search", Technical Report TR-244, Carleton University, 1994.