# Differential Cryptanalysis of DES-like Cryptosystems

(Extended Abstract)

*Eli Biham* *Adi Shamir*

*The Weizmann Institute of Science*
*Department of Applied Mathematics*

**Abstract**

The Data Encryption Standard (DES) is the best known and most widely used cryptosystem for civilian applications. It was developed at IBM and adopted by the National Buraeu of Standards in the mid 70's, and has successfully withstood all the attacks published so far in the open literature. In this paper we develop a new type of cryptanalytic attack which can break DES with up to eight rounds in a few minutes on a PC and can break DES with up to 15 rounds faster than an exhaustive search. The new attack can be applied to a variety of DES-like substitution/permutation cryptosystems, and demonstrates the crucial role of the (unpublished) design rules.

## 1 Introduction

*Iterated cryptosystems* are a family of cryptographically strong functions based on iterating a weaker function $n$ times. Each iteration is called

A.J. Menezes and S.A. Vanstone (Eds.): Advances in Cryptology - CRYPTO '90, LNCS 537, pp. 2-21, 1991.

a *round* and the cryptosystem is called an *n round cryptosystem.* The *round function* is a function of the output of the previous round and of a *subkey* which is a key dependent value calculated via a *key scheduling* algorithm. The round function is usually based on S boxes, bit permutations, arithmetic operations and the exclusive-or (denoted by $\oplus$ and XOR) operations. The *S boxes* are nonlinear translation tables mapping a small number of input bits to a small number of output bits. They are usually the only part of the cryptosystem that is not linear and thus the security of the cryptosystem crucially depends on their choice. The bit permutation is used to rearrange the output bits of the S boxes in order to make the input bits of each S box in the following round depend on the output of as many S boxes as possible. The XOR operation is often used to mix the subkey with the data. In most applications the encryption algorithm is assumed to be known and the secrecy of the data depends only on the secrecy of the randomly chosen key.

An early proposal for an iterated cryptosystems was Lucifer[7], which was designed at IBM to resolve the growing need for data security in its products. The round function of Lucifer has a combination of non linear S boxes and a bit permutation. The input bits are divided into groups of four consecutive bits. Each group is translated by a reversible S box giving a four bit result. The output bits of all the S boxes are permuted in order to mix them when they become the input to the following round. In Lucifer only two fixed S boxes ($S_0$ and $S_1$) were chosen. Each S box can be used at any S box location and the choice is key dependent. Decryption is accomplished by running the data backwards using the inverse of each S box.

The Data Encryption Standard (DES) [14] is an improved version of Lucifer. It was developed at IBM and adopted by the U.S. National Bureau of Standards (NBS) as the standard cryptosystem for sensitive but unclassified data (such as financial transactions and email messages). DES has become a well known and widely used cryptosystem. The key size of DES is 56 bits and the block size is 64 bits. This block is divided into two halves of 32 bits each. The main part of the round function is the *F function*, which works on the right half of the data using a subkey of 48 bits and eight (six

bit to four bit) S boxes. The 32 output bits of the $F$ function are XORed with the left half of the data and the two halves are exchanged. The complete specification of the DES algorithm appears in [14]. In this paper the existence of the initial permutation and its inverse are ignored, since they have no cryptanalytic significance.

An extensive cryptanalytic literature on DES was published since its adoption in 1977. Yet, no short-cuts which can reduce the complexity of cryptanalysis to less than half of exhaustive search were ever reported in the open literature.

The 50% reduction[9] (under a chosen plaintext attack) is based on a symmetry under complementation. Cryptanalysis can exploit this symmetry if two plaintext/ciphertext pairs $(P_1, T_1)$ and $(P_2, T_2)$ are available with $P_1 = \bar{P}_2$.

Diffie and Hellman[6] suggested an exhaustive search of the entire key space in one day on a parallel machine. They estimate the cost of this machine to be \$20-million and the cost per solution to be \$5000.

Hellman[8] presented a time memory tradeoff method for a chosen plaintext attack. A special case of this method takes about $2^{38}$ time and $2^{38}$ memory, with a $2^{56}$ preprocessing time. Hellman suggests a special purpose machine which produces 100 solutions per day with an average wait of one day. He estimates that the machine costs about \$4-million and the cost per solution is about \$1–100. The preprocessing is estimated to take 2.3 years on the same machine.

The *Method of Formal Coding* in which the formal expression of each bit in the ciphertext is found as a XOR sum of products of the bits of the plaintext and the key was suggested in [9]. Schaumuller-Bichl[15,16] studied this method and concluded that it requires an enormous amount of computer memory which makes the whole approach impractical.

In 1987 Chaum and Evertse[2] showed that a meet in the middle attack can reduce the key search for DES with a small number of rounds by the

following factors:

| Number of Rounds | Reduction Factor |
|---|---|
| 4 | $2^{19}$ |
| 5 | $2^{9}$ |
| 6 | $2^{2}$ |
| 7 | – |

They also showed that a slightly modified version of DES with seven rounds can be solved with a reduction factor of 2. However, they proved that a meet in the middle attack of this kind is not applicable to DES with eight or more rounds.

In the same year, Donald W. Davies[3] described a known plaintext cryptanalytic attack on DES. Given sufficient data, it could yield 16 linear relationships among key bits, thus reducing the size of a subsequent key search to $2^{40}$. It exploited the correlation between the outputs of adjacent S boxes, due to their inputs being derived from, among other things, a pair of identical bits produced by the bit expansion operation. This correlation could reveal a linear relationship among the four bits of key used to modify these S box input bits. The two 32-bit halves of the DES result (ignoring IP) receive these outputs independently, so each pair of adjacent S boxes could be exploited twice, yielding 16 bits of key information.

The analysis does not require the plaintext $P$ or ciphertext $T$ but uses the quantity $P \oplus T$ and requires a huge number of random inputs. The S box pairs vary in the extent of correlation they produce so that, for example, the pair S7/S8 needs about $10^{17}$ samples but pair S2/S3 needs about $10^{21}$. With about $10^{23}$ samples, all but the pair S3/S4 should give results (i.e., a total of 14 bits of key information). To exploit all pairs the cryptanalyst needs about $10^{26}$ samples. The S boxes do not appear to have been designed to minimize the correlation but they are somewhat better than a random choice in this respect. The large number of samples required makes this analysis much harder than exhaustive search for the full DES, but for an eight round version of DES the sample size of $10^{12}$ or $10^{13}$ (about $2^{40}$) is on the verge of practicality. Therefore, Davies' analysis had penetrated more

rounds than previously reported attacks.

During the last decade several cryptosystems which are variants of DES were suggested. Schaumuller-Bichl suggested three such cryptosystems [15, 17]. Two of them (called C80 and C82) are based on the DES structure with the replacement of the $F$ function by nonreversible functions. The third one, called The Generalized DES Scheme (GDES), is an attempt to speed up DES. GDES has 16 rounds with the original DES $F$ function but with a larger block size which is divided into more than two parts. She claims that GDES increases the encryption speed of DES without decreasing its security.

Another variant is the Fast Data Encryption Algorithm (Feal). Feal was designed to be efficiently implementable on an eight bit microprocessor. Feal has two versions. The first[19], called Feal-4, has four rounds. Feal-4 was broken by Den-Boer[4] using a chosen plaintext attack with 100–10000 encryptions. The creators of Feal reacted by creating a new version, called Feal-8, with eight rounds and additional XORs of the plaintext and the ciphertext with subkeys[18,13]. Both versions were described as cryptographically better than DES in several aspects.

In this paper we describe a new kind of attack that can be applied to many DES-like iterated cryptosystems. This is a chosen plaintext attack which uses only the resultant ciphertexts. The basic tool of the attack is the *ciphertext pair* which is a pair of ciphertexts whose plaintexts have particular differences. 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.

# 2 Introduction to differential cryptanalysis

*Differential cryptanalysis* is a method which analyses 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 the two plaintexts. In this introduction we show how these differences can be analyzed and exploited.

Let us recall how the DES $F$ function behaves in these terms. The $F$ function takes a 32 bit input and a 48 bit key. The input is expanded (by the $E$ expansion) to 48 bits and XORed with the key. The result is fed into the S boxes and the resultant bits are permuted.

Although DES seems to be very non linear in its input bits, when particular combinations of input bits are modified simultaneously, particular intermidiate bits are modified in a usable way with a relatively high probability after several rounds. We describe this property by means of the particular XOR value of the two plaintexts, the particular XOR value of the intermidiate round and the corresponding probability. Two such encryptions are called a pair.

The XOR of pairs is invariant in the XORing of the key and is linear in the $E$ expansion, the $P$ permutation and the XOR operation between the left half of the data and the output of the $F$ function. Therefore, it is very easy to push the knowledge of such a XOR in the input over these operations.

DES contains also S boxes which are non linear tables. Knowledge of the input XOR of a pair cannot guarantee knowledge of its output XOR. However, every input XOR of an S box suggests a probabilistic distribution of the possible output XORs. In this distribution several output XORs have a relatively high probability. Table 1 describes the distribution of the output XORs for several input XORs in S1. The table counts the number of possible pairs that lead to each of the entries. Each input XOR has 64 possible pairs which are divided among the 16 entries in a row. The average entry is thus four, representing a probability of $\frac{4}{64} = \frac{1}{16}$. We see that a zero input XOR has only one possible output XOR, which is zero. Many entries are impossible or have a small probability. However, there are several entries with probability $\frac{1}{4}$ (i.e., 16 out of 64) or close to it.

| Input XOR | Output XOR | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $0_x$ | $1_x$ | $2_x$ | $3_x$ | $4_x$ | $5_x$ | $6_x$ | $7_x$ | $8_x$ | $9_x$ | $A_x$ | $B_x$ | $C_x$ | $D_x$ | $E_x$ | $F_x$ |
| $0_x$ | 64 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $1_x$ | 0 | 0 | 0 | 6 | 0 | 2 | 4 | 4 | 0 | 10 | 12 | 4 | 10 | 6 | 2 | 4 |
| $2_x$ | 0 | 0 | 0 | 8 | 0 | 4 | 4 | 4 | 0 | 6 | 8 | 6 | 12 | 6 | 4 | 2 |
| $3_x$ | 14 | 4 | 2 | 2 | 10 | 6 | 4 | 2 | 6 | 4 | 4 | 0 | 2 | 2 | 2 | 0 |
| $4_x$ | 0 | 0 | 0 | 6 | 0 | 10 | 10 | 6 | 0 | 4 | 6 | 4 | 2 | 8 | 6 | 2 |
| $5_x$ | 4 | 8 | 6 | 2 | 2 | 4 | 4 | 2 | 0 | 4 | 4 | 0 | 12 | 2 | 4 | 6 |
| $6_x$ | 0 | 4 | 2 | 4 | 8 | 2 | 6 | 2 | 8 | 4 | 4 | 2 | 4 | 2 | 0 | 12 |
| $7_x$ | 2 | 4 | 10 | 4 | 0 | 4 | 8 | 4 | 2 | 4 | 8 | 2 | 2 | 2 | 4 | 4 |
| $8_x$ | 0 | 0 | 0 | 12 | 0 | 8 | 8 | 4 | 0 | 6 | 2 | 8 | 8 | 2 | 2 | 4 |
| $9_x$ | 10 | 2 | 4 | 0 | 2 | 4 | 6 | 0 | 2 | 2 | 8 | 0 | 10 | 0 | 2 | 12 |
| $A_x$ | 0 | 8 | 6 | 2 | 2 | 8 | 6 | 0 | 6 | 4 | 6 | 0 | 4 | 0 | 2 | 10 |
| $B_x$ | 2 | 4 | 0 | 10 | 2 | 2 | 4 | 0 | 2 | 6 | 2 | 6 | 6 | 4 | 2 | 12 |
| $C_x$ | 0 | 0 | 0 | 8 | 0 | 6 | 6 | 0 | 0 | 6 | 6 | 4 | 6 | 6 | 14 | 2 |
| $D_x$ | 6 | 6 | 4 | 8 | 4 | 8 | 2 | 6 | 0 | 6 | 4 | 6 | 0 | 2 | 0 | 2 |
| $E_x$ | 0 | 4 | 8 | 8 | 6 | 6 | 4 | 0 | 6 | 6 | 4 | 0 | 0 | 4 | 0 | 8 |
| $F_x$ | 2 | 0 | 2 | 4 | 4 | 6 | 4 | 2 | 4 | 8 | 2 | 2 | 2 | 6 | 8 | 8 |
| | | | | | | | | | ⋮ | | | | | | | |
| $30_x$ | 0 | 4 | 6 | 0 | 12 | 6 | 2 | 2 | 8 | 2 | 4 | 4 | 6 | 2 | 2 | 4 |
| $31_x$ | 4 | 8 | 2 | 10 | 2 | 2 | 2 | 2 | 6 | 0 | 0 | 2 | 2 | 4 | 10 | 8 |
| $32_x$ | 4 | 2 | 6 | 4 | 4 | 2 | 2 | 4 | 6 | 6 | 4 | 8 | 2 | 2 | 8 | 0 |
| $33_x$ | 4 | 4 | 6 | 2 | 10 | 8 | 4 | 2 | 4 | 0 | 2 | 2 | 4 | 6 | 2 | 4 |
| $34_x$ | 0 | 8 | 16 | 6 | 2 | 0 | 0 | 12 | 6 | 0 | 0 | 0 | 0 | 8 | 0 | 6 |
| $35_x$ | 2 | 2 | 4 | 0 | 8 | 0 | 0 | 0 | 14 | 4 | 6 | 8 | 0 | 2 | 14 | 0 |
| $36_x$ | 2 | 6 | 2 | 2 | 8 | 0 | 2 | 2 | 4 | 2 | 6 | 8 | 6 | 4 | 10 | 0 |
| $37_x$ | 2 | 2 | 12 | 4 | 2 | 4 | 4 | 10 | 4 | 4 | 2 | 6 | 0 | 2 | 2 | 4 |
| $38_x$ | 0 | 6 | 2 | 2 | 2 | 0 | 2 | 2 | 4 | 6 | 4 | 4 | 4 | 6 | 10 | 10 |
| $39_x$ | 6 | 2 | 2 | 4 | 12 | 6 | 4 | 8 | 4 | 0 | 2 | 4 | 2 | 4 | 4 | 0 |
| $3A_x$ | 6 | 4 | 6 | 4 | 6 | 8 | 0 | 6 | 2 | 2 | 6 | 2 | 2 | 6 | 4 | 0 |
| $3B_x$ | 2 | 6 | 4 | 0 | 0 | 2 | 4 | 6 | 4 | 6 | 8 | 6 | 4 | 4 | 6 | 2 |
| $3C_x$ | 0 | 10 | 4 | 0 | 12 | 0 | 4 | 2 | 6 | 0 | 4 | 12 | 4 | 4 | 2 | 0 |
| $3D_x$ | 0 | 8 | 6 | 2 | 2 | 6 | 0 | 8 | 4 | 4 | 0 | 4 | 0 | 12 | 4 | 4 |
| $3E_x$ | 4 | 8 | 2 | 2 | 2 | 4 | 4 | 14 | 4 | 2 | 0 | 2 | 0 | 8 | 4 | 4 |
| $3F_x$ | 4 | 8 | 4 | 2 | 4 | 0 | 2 | 4 | 4 | 2 | 4 | 8 | 8 | 6 | 2 | 2 |

Table 1: Partial pairs XOR distribution table of S1(values subsripted with x are in hexadecimal)

We can use this property as a tool to identify key bits. If we can find the output XOR of the $F$ function of the last round, we can filter the set of possible subkeys entering this $F$ function when the pair of ciphertexts is known. Using both ciphertexts it is easy to calculate the input XOR of the $F$ function of the last round and its output XOR. Then the input XOR and output XOR of each S box in the last round are known. In case $k$ input pairs can lead to that entry in the table, exactly $k$ values of the corresponding six subkey bits are possible. Most subkey values are suggested by only few pairs. However, the real value of the subkey bits is suggested by all the pairs and thus can be identified.

The following introductory example breaks a three round DES. The attack uses pairs of ciphertexts whose corresponding plaintext XORs consist of a zero right half and an arbitrary left half. The main part of the algorithm counts the number of pairs which suggest each possible value of the six key bits entering each S box. For each ciphertext pair we do the following: The input XOR of the $F$ function in the first round is zero and thus its output XOR must be zero. The left half of the ciphertext is calculated as the XOR value of the left half of the plaintext, the output of the first round and the output of the third round ($l = L \oplus A \oplus C$, where $L$ is the left half of the plaintext and $l$ is the left half of the ciphertext. See figure 1). Since the plaintext XOR is fixed and the ciphertext XOR is known and the output XOR of the first round is zero, the output XOR of the $F$ function in the third round can be calculated as the XOR of the left half of the plaintext XOR and the left half of the ciphertext XOR. The inputs of the $F$ function in the third round are easily extractable from the ciphertext pair.

The following analysis can be done for each S box in the third round. In this example we only show how to find the six key bits entering S1 in the third round. The input XOR and output XOR of S1 can be easily derived from the input XOR and output XOR of the $F$ function. Let $k$ be the number of possible input pairs to S1 with those input XOR and output XOR. The value of the input bits (denoted by $S_E$) which are XORed with the six key bits (denoted by $S_K$) to make the actual value of the input of the S box (denoted by $S_I$) is extractable from the ciphertext. Therefore, there are exactly $k$ key values suggested by the pair via $S_K = S_E \oplus S_I$. For

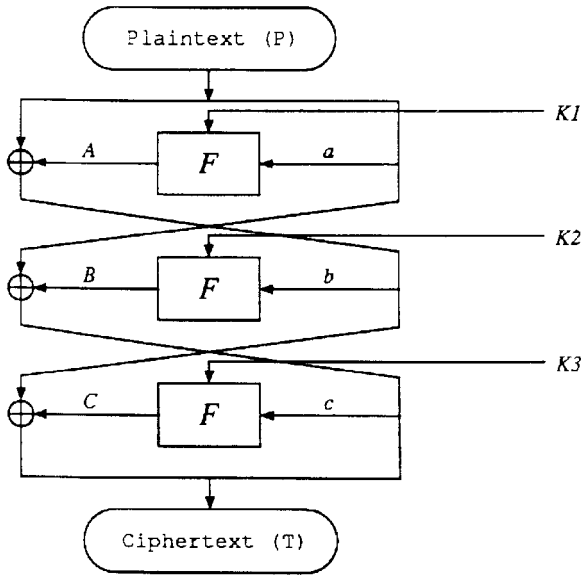



Figure 1: DES with three rounds

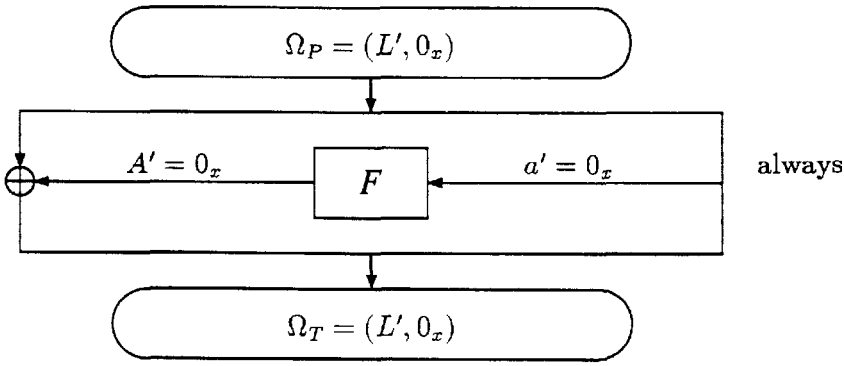
| S box input | | Possible Keys | |
|---|---|---|---|
| 06, | 32 | 07, | 33 |
| 10, | 24 | 11, | 25 |
| 16, | 22 | 17, | 23 |
| 1$C$, | 28 | 1$D$, | 29 |

Table 2: Possible keys for $34 \rightarrow D$ by S1 with input 1, 35 (in hexadecimal)

example, if the ciphertext bits entering S1 in the third round of the two encryptions are $S_E = 1$, $S_E^* = 35$ and the output XOR is $D$ then the value of $S_K$ must appear in table 2. Each line in the table describes two pairs with the same two inputs but with two possible orders. Each pair leads to one key, so each line leads to two keys (which are $S_E \oplus S_I$ and $S_E^* \oplus S_I$). Given many pairs we can count the number of pairs suggesting each possible value of the key bits entering S1. The value which is suggested by all the pairs must be the real value of the key bits.

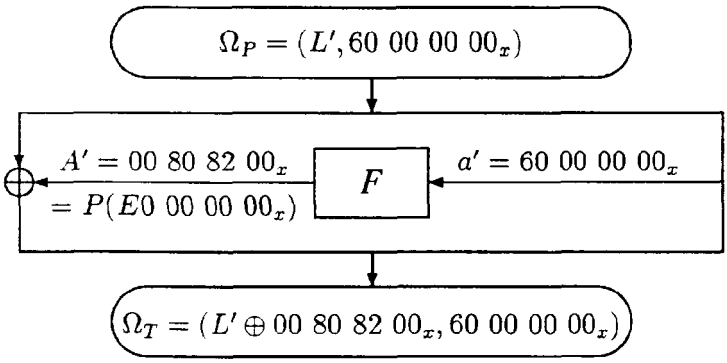
We now describe the basic tool which allows us to push the knowledge of the plaintext XOR to a knowledge of an intermidiate XOR after as many rounds as possible, which is called an $n$ round characteristic. Every $n$ round characteristic has a particular plaintext XOR $\Omega_P$, a particular XOR of the data in the $n^{\text{th}}$ round $\Omega_T$ and a probability $p^{\Omega}$ in which the XOR of the data in the $n^{\text{th}}$ round is $\Omega_T$ when random pairs whose plaintext XOR is $\Omega_P$ are used. Any pair whose plaintext XOR is $\Omega_P$ and whose XOR of the data in the $n^{\text{th}}$ round (using a particular key) is $\Omega_T$ is called a right pair. Any other pair is called a wrong pair. Therefore, the right pairs form a fraction $p^{\Omega}$ of all the possible pairs. In the following examples of characteristics we denote the input and output XORs of the $F$ function in the first round by $a'$ and $A'$ respectively, the input and output XORs of the $F$ function in the second round by $b'$ and $B'$ respectively, etc.

The following one round characteristic has probability 1 (for any $L'$). This is the only one round characteristic with probability greater than $\frac{1}{4}$.




This characteristic has special importance since it plays a role in all the characteristics used to break DES-like cryptosystems.

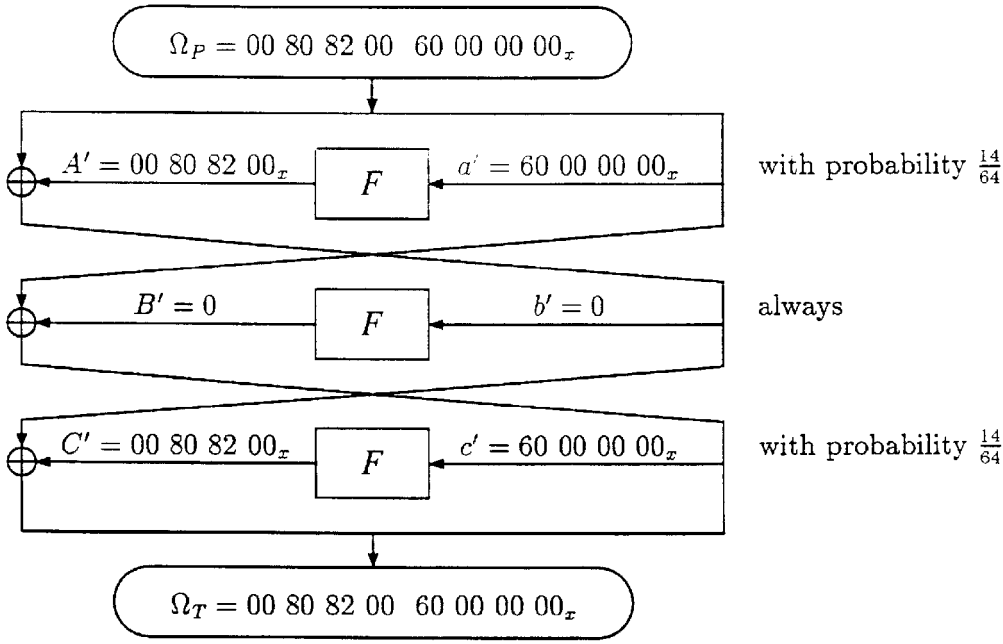
The following one round characteristic has probability $\frac{14}{64}$ (for any $L'$).




with probability $\frac{14}{64}$

The concatenation of two characteristics lets us build a longer characteristic with probability which is close to the product of their probabilities. A concatenation of characteristics $\Omega^1$ and $\Omega^2$ can be done if the swapped value of the two halves of $\Omega^1_T$ equals $\Omega^2_P$. By concatenating the two characteristics described above we can get the following three round characteristic

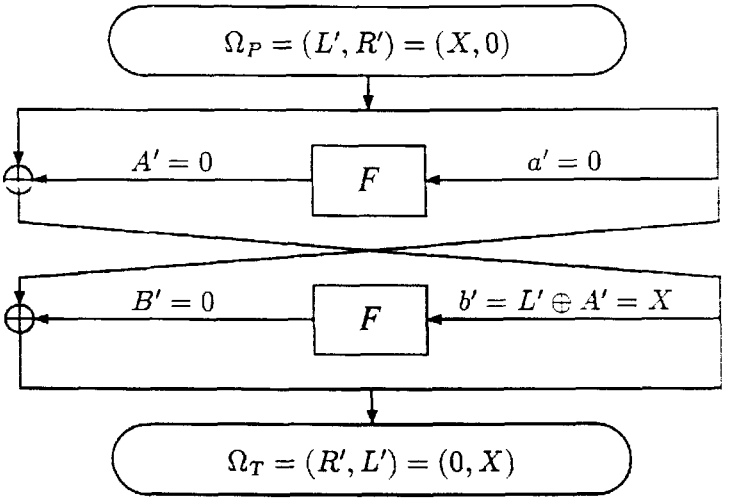
with probability $\left(\frac{14}{64}\right)^2 \approx 0.05$:




Several characteristics can be concatenated to themselves. These characteristics are called iterative characteristics. We can concatenate an iterative characteristic to itself any number of times. The advantage of iterative characteristics is that we can build an $n$ round characteristic for any large $n$ with a fixed reduction rate of the probability for each additional round, while in non iterative characteristics the reduction rate of the probability usually increases due to the avalanche effect.

There are several kinds of iterative characteristics but the simplest ones are the most useful. These characteristics are based on a non zero input XOR to the $F$ function that may cause a zero output XOR (i.e., two different inputs yield the same output). This is possible in the $F$ function of DES if at least three neighboring S boxes differ in the pair (see also at [5,1]). The structure of these characteristics is as follows. The input XOR of the $F$ function is marked by $X$, s.t. there are as many pairs as possible

whose input XOR is $X$ and the output XOR is zero.



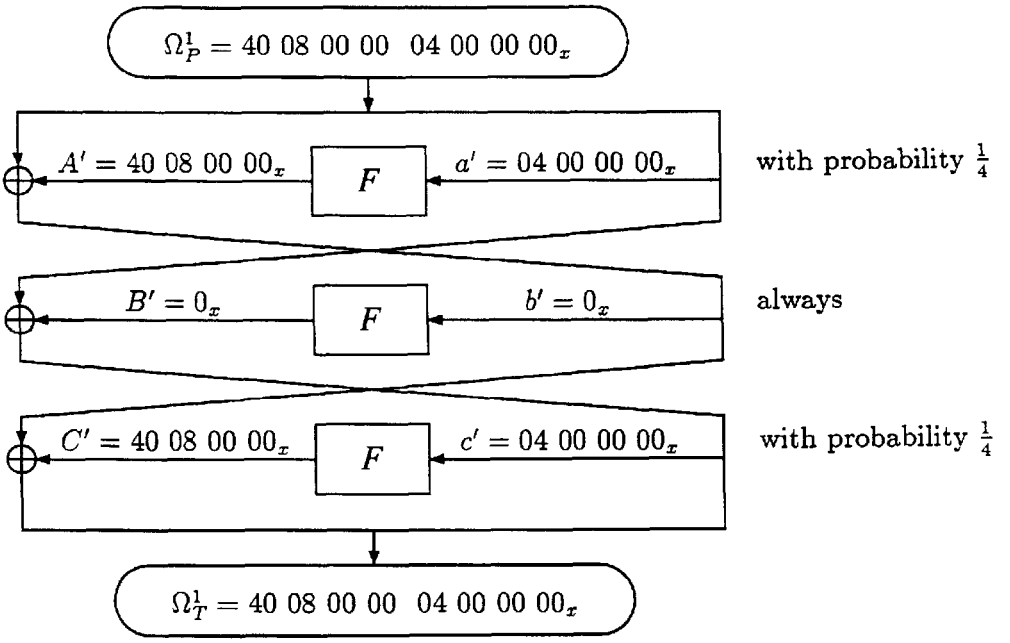

The best such characteristic (with $X$ = 19 60 00 00) has probability about $\frac{1}{234}$. This characteristic is used in the attacks on DES with nine or more rounds.

The statistical behavior of most characteristics does not allow us to look for the intersection of all the keys suggested by the various pairs as we did before, since the intersection is usually empty: the wrong pairs do not necessarily list the right key as a possible value. However, we know that the right key value should result from all the right pairs, which occur with the characteristic's probability. All the other possible key values are fairly randomly distributed: the expected XOR value (which is usually not the real value in the pair) with the known ciphertext pair can cause any key value to be possible, and even the wrong key values suggested by the right pairs are quite random. Consequently, the right key appears with the characteristic's probability (from right pairs) plus other random occurrences (from wrong pairs). To find the key we just have to count the number of occurrences of each of the suggested keys. When the number of pairs is large enough, the right key is likely to be the one that occurs most often.

# 3 Breaking DES with six rounds

The cryptanalysis of DES with six rounds is more complex than the cryptanalysis of the three round version. We use two statistical characteristics with probability $\frac{1}{16}$, and choose the key value that is counted most often. Each one of the two characteristics lets us find the 30 key bits of K6 which are used at the input of five S boxes in the sixth round, but three of the S boxes are common so the total number of key bits found by the two characteristics is 42. The other 14 key bits can be found later by means of exhaustive search or by a more careful counting on the key bits entering the eighth S box in the sixth round.

The first characteristic $\Omega^1$ is:

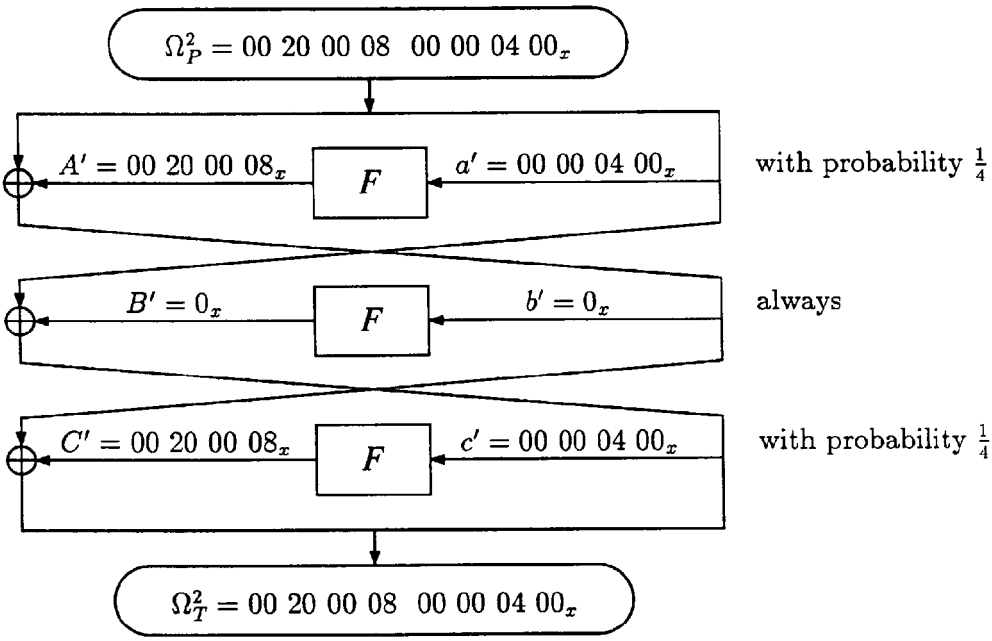



Where in the fourth round $d' = b' \oplus C' = 40\ 08\ 00\ 00_x$.

Five S boxes in the fourth round (S2, S5, ..., S8) have zero input XORs ($S'_{Ed} = 0$) and thus their output XORs are zero ($S'_{Od} = 0$). The

corresponding output XORs in the sixth round can be found by $F' = c' \ominus D' \oplus l'$. Since the right key value is not suggested by all the pairs (due to the probabilistic nature of the characteristic). We should simultaneously count on subkey bits entering several S boxes. The best approach is to count on all the 30 countable subkey bits together, which maximizes the probability that the right key value is the one counted most often. A straightforward implementation of this method requires $2^{30}$ counters, which is impractical on most computers. However, the improved counting procedure described in the full paper achieves exactly the same result with much smaller memory (the program size is about 100K bytes on a personal computer).

The same efficient algorithm is used to find the 30 key bits of S1, S2, S4, S5 and S6 using the second characteristic $\Omega^2$ which is:




Where in the fourth round $d' = b' \oplus C' = 00\ 20\ 00\ 08_x$.

Again, five S boxes in the fourth round (S1, S2, S4, S5 and S6) have zero input XORs. The computed key values of the common S boxes S2, S5 and S6 should be the same in both calculations (otherwise we should ana-

lyze more pairs or consider additional candidate keys with almost maximal counts). If this test is successful, we have probably found 42 bits of K6.

DES has 56 key bits and 14 of them are still missing. We can find them by searching all the $2^{14}$ possibilities for the expected plaintext XOR value of the decrypted ciphertexts.

# 4 Results

We now describe the results we get by this kind of attacks. Due to space limitations, we are not describing the attacks in detail in this extended abstract. Note that the complexities we quote are the number of encryptions needed to create all the necessary pairs while the attacking algorithm itself uses fewer and simpler operations.

DES with six rounds was broken in less than 0.3 seconds on a personal computer using 240 ciphertexts. DES with eight rounds was broken in less than two minutes on a computer by analysing 15000 ciphertexts chosen from a pool of 50000 candidate ciphertexts. DES with up to 15 rounds is breakable faster than exhaustive search. DES with 15 rounds can be broken in about $2^{52}$ steps but DES with 16 rounds still requires $2^{58}$ steps (which is slightly higher than the complexity of an exhaustive search). A summary of the cryptanalytic results on DES with intermidiate number of rounds appears in table 3.

Some researchers have proposed to strengthen DES by making all the subkeys $Ki$ independent (or at least to derive them in a more complicated way from a longer actual key $K$). Our attack can be carried out even in this case. DES with eight rounds with independent subkeys (i.e., with $8 \cdot 48 = 384$ independent bits which are not compatible with the key scheduling algorithm) was broken in less than two minutes using the same ciphertexts as in the case of dependent subkeys. DES with independent subkeys (i.e., with $16 \cdot 48 = 768$ independent bits) is breakable within $2^{61}$ steps. As a result, any modification of the key scheduling algorithm cannot make DES much stronger. The attacks on DES with 9–16 rounds are not

| Rounds | Complexity |
|---|---|
| 4 | $2^4$ |
| 6 | $2^8$ |
| 8 | $2^{16}$ |
| 9 | $2^{26}$ |
| 10 | $2^{35}$ |
| 11 | $2^{36}$ |
| 12 | $2^{43}$ |
| 13 | $2^{44}$ |
| 14 | $2^{51}$ |
| 15 | $2^{52}$ |
| 16 | $2^{58}$ |

Table 3: Summary of the cryptanalysis of DES

influenced by the $P$ permutation and the replacement of the $P$ permutation by any other permutation or function can't make them less successful. On the other hand, the replacement of the order of the eight DES S boxes (without changing their values) can make DES much weaker: DES with 16 rounds with a particular replaced order is breakable in about $2^{46}$ steps. The replacement of the XOR operation by the more complex addition operation makes this cryptosystem much weaker. DES with random S boxes is shown to be very easy to break. Even a minimal change of one entry in one of the DES S boxes can make DES easier to break. GDES is shown to be trivially breakable with six encryptions in less than 0.2 seconds, while GDES with independent subkeys is breakable with 16 encryptions in less than 3 seconds.

This attack is applicable also to a wide variety of DES-like cryptosystems. Lucifer with eight rounds can be broken using less than 60 ciphertexts (30 pairs). The Feal-8 cryptosystem can be broken with less than 2000 ciphertexts (1000 pairs) and the Feal-4 cryptosystem can be broken with just eight ciphertexts and one of their plaintexts. As a reaction to our attack on Feal-8, its creators introduced Feal-N[11], with any even number of rounds N. They suggest the use of Feal-N with 16 and 32 rounds.

Nevertheless, Feal-N can be broken for any N $\leq$ 31 rounds faster than an exhaustive search.

Differential cryptanalytic techniques are applicable to hash functions, in addition to cryptosystems. For example, the following messages hash to the same value in Merkle's Snefru[10] function with two pass:

- `00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000`
- `00000000 f1301600 13dfc53e 4cc3b093 37461661 ccd8b94d 24d9d35f 71471fde 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000`
- `00000000 1d197f00 2abd3f6f cf33f3d1 8674966a 816e5d51 acd9a905 53c1d180 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000`
- `00000000 e98c8300 1e777a47 b5271f34 a04974bb 44cc8b62 be4b0efc 18131756 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000`

and the following two messages hash to the same value in a variant of Miyaguchi's N-Hash[12] function with six rounds:

- `CAECE595 127ABF3C 1ADE09C8 1F9AD8C2`
- `4A8C6595 921A3F3C 1ADE09C8 1F9AD8C2.`

# References


[1] E. F. Brickell, J. H. Moore, M. R. Purtill, *Structure in the S-Boxes of the DES*, Advances in cryptology, proceedings of CRYPTO 86, pp. 3–7, 1986.

[2] David Chaum, Jan-Hendrik Evertse, *Cryptanalysis of DES with a reduced number of rounds, Sequences of linear factors in block ciphers*, technical report, 1987.

[3] D. W. Davies, private communications.

[4] Bert Den Boer, *Cryptanalysis of F.E.A.L.*, Advances in cryptology, proceedings of EUROCRYPT 88, 1988.

[5] Yvo Desmedt, Jean-Jacque Quisquater, Marc Davio, *Dependence of output on input in DES: small avalanche characteristics*, Advances in cryptology, proceedings of CRYPTO 84, pp. 359–376, 1984.

[6] W. Diffie and M. E. Hellman, *Exhaustive cryptanalysis of the NBS Data Encryption Standard*, Computer, Vol. 10, No. 6, pp. 74–84, June 1977.

[7] H. Feistel, *Cryptography and data security*, Scientific american, Vol 228, No. 5, pp. 15–23, May 1973.

[8] M. E. Hellman, *A Cryptanalytic Time-Memory Tradeoff*, IEEE Trans. Inform. Theory, Vol. 26, No. 4, pp. 401–406, July 1980.

[9] M. E. Hellman, R. Merkle, R. Schroppel, L. Washington, W. Diffie, S. Pohlig and P. Schweitzer, *Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard*, Stanford university, September 1976.

[10] Ralph C. Merkle, technical report, March 1990.

[11] Shoji Miyaguchi, *Feal-N specifications.*

[12] S. Miyaguchi, K. Ohta, M. Iwata, *128-bit hash function (N-Hash)*, proceedings of SECURICOM90, March 1990.

[13] Shoji Miyaguchi, Akira Shiraishi, Akihiro Shimizu, *Fast data encryption algorithm Feal-8*, Review of electrical communications laboratories, Vol. 36 No. 4, 1988.

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

[15] Ingrid Schaumuller-Bichl, *Zur Analyse des Data Encryption Standard und Synthese Verwandter Chiffriersysteme*, thesis, May 1981.

[16] Ingrid Schaumuller-Bichl, *Cryptanalysis of the Data Encryption Standard by the Method of Formal Coding*, Cryptologia, proceedings of CRYPTO 82, pp. 235–255, 1982.

[17] Ingrid Schaumuller-Bichl, *On the Design and Analysis of New Cipher Systems Related to the DES*, technical report, 1983.

[18] Akihiro Shimizu, Shoji Miyaguchi, *Fast Data Encryption Algorithm Feal*, Advances in cryptology, proceedings of EUROCRYPT 87, pp. 267, 1987.

[19] Akihiro Shimizu, Shoji Miyaguchi, *Fast Data Encryption Algorithm Feal*, Abstracts of EUROCRYPT 87, Amsterdam, April 1987.