Differential Cryptanalysis and the Theory of S-Box

Chun Tian
University of Bologna
chun.tian@studio.unibo.it

Abstract

In this paper, we present a survey and tutorial on differential cryptanalysis, one of the most significant attacks applicable to symmetric-key block ciphers, together with an introduction on the mathematical theory of S-box, the core of block ciphers. We first demonstrate how differential cryptanalysis is applied on simple ciphers and the Data Encryption Standard (DES) with reduced rounds, then explain how DES's eight S-boxes were designed to be resistant to differential cryptanalysis, on both theory and practical levels. This paper can be used as a quick learning resource on differential cryptanalysis for people who is familiar with the description of DES.

General Terms Cryptography

Keywords DES, Differential Cryptanalysis

1. Introduction

1.1 Motivation

Cryptology[1] is important to this world. I still remember the first scene in an old American movie *A beautiful mind*, which is based on the story of John Nash[2]: in the welcome speech given by his department head when Nash for the first time came to Princeton University: *"Mathematicians won the war, mathematicians broke Japanese's codes, and built the atom bomb. Mathematicians, like you."*. Together with the story of the Cryptanalysis of the Enigma[3] during the Second World War, all these things made Cryptology (as a branch of Mathematics) a fascinating and mysterious area to me.

My first introduction book in Cryptology was the first edition of F. L. Bauer's book [1] (published in 1997, Chinese edition published in 2001) that I read more than 10 years ago. From today's view, especially after I've formally learnt this course, this book now looks more like a survey of ancient and classical Cryptology. Things like DES, RSA and Differential Cryptanalysis have just a few pages without any detail. Instead, I found long discussions in several chapters focusing on the cryptanalysis of ciphertexts from a linguistic approach, using natural language redundancy to decrypt the ciphertext. Unfortunately, these beautiful and interesting contents are not part of modern cryptography any more, thus useless to our current course. However, I did have learnt five important maxims of Cryptology from this book:

1. "One should not underrate the adversary."
2. "Only a cryptanalyst, if anybody, can judge the security of a cryptosystem."
3. "In judging the encryption security of a class of methods, one has to take into account that the adversary knows the class of methods."
4. "Superficial complications can be illusory, for they can provide the cryptographer with a false sense of security."
5. "In judging the encryption security of a class of methods, cryptographic faults and other infringements of security discipline are to be taken into account."

Although above Maxim No. 3, i.e. the *Kerckhoffs's principle*[4] lead us to focus on certain cryptosystem in which all details (except the key) were known, I have to admit that, even after learning this course (with all its contents well understood), I still have little idea on how to conclude out the cryptosystem itself from any number of message-ciphertext pairs, even with some keys.[5]

And above Maxim No. 2 (*Only a cryptanalyst, if anybody, can judge the security of a cryptosystem*) made me think that, to *really* understand the security of a cryptosystem, I must try to learn something on cryptanalysis and actually try to attack some simple cryptosystems, otherwise it's really hard to convince myself that I really have learnt something in this area.

According to [1], cryptography and cryptanalysis are the two faces of cryptology; each depends on the other and each influences the other in an interplay of improvements to strengthen cryptanalytic security on the one side and efforts to mount more efficient attacks on the other side." An course in college on Cryptography usually concerns more about cryptographic methods, less about cryptanalysis. After all the teaching hours are quite limited, and "a course on cryptanalysis is problematic: either it is not conclusive enough, in which case it is useless, or it is conclusive, but touches a sensitive area." So it's reasonable that students who are interested in cryptanalysis must learn (at least the deep part) by themselves. That's *why* I chose this topic.

1.2 Block ciphers and DES

Block ciphers have played a central role in the development of large-scale commercial cryptology. We can use a block cipher to build both a stream cipher and a message authentication code. We can even build a keyless hash function out of a block cipher. Even in an asymmetric cryptosystem, usually the first step is to negotiate a shared key between the message sender and receiver, then still use symmetric cryptosystem (e.g. a block cipher) to actually transfer the actual (long) message, because a block cipher is usually around 50 times faster than an asymmetric algorithm.

[1] Here the word *Cryptology* is to indicate the whole area with two faces: Cryptography and Cryptanalysis.

[2] American mathematician who made fundamental contributions to game theory, differential geometry, and the study of partial differential equations. He shared the 1994 Nobel Memorial Prize in Economic Sciences.

[3] `https://en.wikipedia.org/wiki/Cryptanalysis_of_the_Enigma`

[4] "The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience."

[5] Maybe sometimes the details about a cryptosystem can be retrieved from other channels, e.g. an Enigma machine captured from Germans.

It seems that modern cryptography was built on beautiful theories and actual algorithms designed by genius, although sometimes there're gaps between theoretical and practical parts. First of all, it's a little funny that, the security of our world is protected by several genius-made cryptosystems that we never had formal (or theoretical) proofs for their security. Secondly, we usually got a strong cryptosystem first, then academic world does reverse engineering on its design and then the related works become standard parts of Cryptography textbooks.

The Data Encryption Standard (DES) is one such great example. The importance of DES cannot be overstated and it has been said that "DES trained a generation of cryptographers". [10] When DES was first published in 1975, it was anticipated that the cipher would remain acceptable for ten years. Yet after 30 years, it remains one of the most widely implemented ciphers in the world. After almost 30 years of intensive study, the best known practical attack on DES is still an exhaustive search through its key space. [9] Even today, 40 years since 1975, its strengthened form (Triple-DES) remains US federal standard, and is, without argument, the most trusted block cipher available today.

When reviewing DES algorithm, people may think (at least me) that, they have understood the whole algorithm once they understood the encryption process - *how data blocks go through the network with 16 rounds*. This process looks clear and simple, and it's done all by simple operations like exclusive-or ($\oplus$). However, understanding the encryption process has *nothing to do* with understanding *why DES is so secure*, because the security of DES depends on its S-boxes, which from the view of beginners are just meaningless random numbers. Further speaking, any kind of cryptanalytic attack in which there's no special treatment on S-boxes (i.e. the attacking process can literally apply to all possible S-boxes without changing even parameters), will not succeed under a *better-than-bruce-force* goal.

All these words seems indicating that, before going to any higher level in cryptography, it's very important to understand DES and its attacking techniques first.

1.3 Differential Cryptanalysis

Differential cryptanalysis is one of the most powerful attacks available to the block cipher cryptanalyst. It's invented by Biham and Shamir in 1991 but identified more than 10 years earlier by the designers of DES at IBM. It's the the first known better-than-brute-force attack on DES, although this attack was only a breakthrough from a theoretical standpoint.

Biham and Shamir's first paper [2] in 1991 has only successfully attacked DES with reduced rounds up to 15 (full DES has 16 rounds). This is essentially better than any attacks on DES before their work. One year later they have improved their techniques and successfully attacked Full 16-round DES [3]. Besides showing theoretical attacks, what's really interesting in their work, is the fact they DES had been carefully designed to be resistant to differential cryptanalysis. This conclusion was soon confirmed by Don Coppersmith, one of the original DES designer, in 1994 [5].

Although differential cryptanalysis is not a practical on the original DES, it is practical on many DES variants and some other block ciphers which aimed to replace DES with better security. One notable example is the *Fast Data Encipherment Algorithm* (FEAL). [4, Chapter 6] FEAL aims to replace DES, however, the security of FEAL was not as high as expected by the designers: its first version was easily broken by differential cryptanalysis using only 12 chosen texts. These stories may be well-known to most people who learnt Cryptography, but few actually studied the details. The aim of the rest of this exam paper is to make a better-than-ever tutorial (yet still short) on differential cryptanalysis and its application on DES-like cryptosystems.

It's also worthy to point out that, beside differential cryptanalysis, *linear cryptanalysis* provides the most important general technique for analyzing a block cipher. However, "generally speaking it tends to be far less successfully than differential cryptanalysis, the most prominent exception to this rule of thumb being DES" [10]. Since our ultimate goal is not attacking DES but to be equipped with at least one practical technique in cryptanalysis, the area linear cryptanalysis is completely untouched in scope of this exam paper.

2. Differential Cryptanalysis: the Idea

To motivate the whole idea of differential cryptanalysis, we can consider a perfectly secret encryption sheme called the *one-time pad*. Let m, c, and k be b-bit strings where m represent the message (or plaintext), c the ciphertext, and k the secret key. We can encrypt the message m in the following way:

$$c = m \oplus k$$

where $\oplus$ is the (bitwise) exclusive-or operation. If the key k is chosen at random and used only once, then the cryptanalysis can gain no information about m from observing the ciphertext c. In Shannon's framework this cryptosystem is unconditionally secure under a ciphertext-only attack.

What happens if we use the key twice? Assume we use key k to encrypt m_0 and m_1 to give c_0 and c_1. The an attacker intercepting both ciphertexts can compute

$$c_0 \oplus c_1 = (m_0 \oplus k) \oplus (m_1 \oplus k) = m_0 \oplus m_1.$$

In other words, the cryptanalyst recovers the exclusive-or of two plaintext messages directly from the intercepted ciphertexts. If the mssages contain redundancy, for example, if they represent text from a natural (spoken) language, then the attacker can deduce information about the plaintext from the exclusive-or of the ciphertexts.

This is not particularly sophisticated, but it does highlight the key idea behind differential cryptanalysis. While we might not get much information from considering a single message and ciphertext, we might gain much more by considering pairs of messages and ciphertext. In our simple example the action of the secret key k could be entirely removed by simply manipulating the ciphertexts. In reality things will be far more complex, but looking at the difference between paris of plaintexts as they are encrypted–and choosing a notion of difference that allows us to ignore the action of the key for at least part of the analysis–forms the main idea behind differential cryptanalysis.

In the rest of this section, we will use four simple block ciphers from [10, Chapter 6] to introduce the main features of differential cryptanalysis.

2.1 Differential Behaviour

$$c = S[m \oplus k_0] \oplus k_1$$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

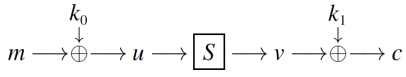


Figure 1. Encryption with CIPHERONE

CIPHERONE is a four-bit block cipher with an eight-bit key and it uses a four-bit look-up box $S[\cdot]$. Encryption with CIPHERONE is

described in Figure 1 where the four-bit message m is encrypted using an eight-bit key $k_0||k_1$ where both k_0 and k_1 are four-bit randomly chosen round keys. $S[\cdot]$ is a four-bit permutation and we assume that the details of $S[\cdot]$ are public (Kerckhoffs' assumption).

Consider the *differential behaviour of* CIPHERONE, i.e. let us trace a difference between two plaintexts that are encrypted with CIPHERONE. Assume that a cryptanalyst knows two message inputs m_0 and m_1 along with the corresponding ciphertexts c_0 and c_1. Since the value of k_0 is secret the cryptanalyst doesn't know the values of u_0 or u_1, which are internal to the cipher. However, the cryptanalyst *does* know the value of the *difference* between these two internal values since

$$u_0 \oplus u_1 = (m_0 \oplus k_0) \oplus (m_1 \oplus k_0) = m_0 \oplus m_1.$$

This can be used to find the secret value of k_1 in the following way:

1. Given two message-ciphertext pairs (m_0, c_0) and (m_1, c_1) the cryptanalyst can compute the value of $u_0 \oplus u_1$.
2. Next the cryptanalyst can guess the value of k_1 and compute the values of v_0 and v_1 directly from c_0 and c_1.
3. Furthermore, since $S[\cdot]$ is publicly known and invertible, the cryptanalyst can compute $S^{-1}[v_0]$ and $S^{-1}[v_1]$.
4. Now, the cryptanalyst cannot compare these value of u_0 and u_1 directly since these values are unknown. However, for the correct value of k_1, the cryptanalyst does know that

   $$u_0 \oplus u_1 = S^{-1}[v_0] \oplus S^{-1}[v_1].$$

   The value $u_0 \oplus u_1$ is known since it is equal to $m_0 \oplus m_1$. So the cryptanalyst can simply try all values t of k_1 one by one and whenever the expected value of $S^{-1}[t \oplus c_0] \oplus S^{-1}[t \oplus c_1]$ is obtained the value t is noted as a candidate for the secret key k_1.
5. If more than one candidate for the key k_1 remains, the attack can be repeated on other messages and ciphertexts.
6. Once k_1 is known, using any single message-ciphertext pair, we can compute out k_0, because

   $$k_0 = m \oplus u = m \oplus S^{-1}[k_1 \oplus c]$$

Although there are certain conditions that ensured our attack worked and it might not be clear why some particular message-ciphertext pairs worked so well. Nevertheless, this this attack on a simple cipher allows us to highlight two important things:

1. Even though we might not know the value of internal variables during the encryption process, we were able to state the difference between variables at certain points during encryption.
2. We were able to recover key information by guessing values to part of the secret key and testing whether some differential condition held. This is the most common method of recovering key material when performing differential cryptanalysis.

2.2 Difference Distribution of S-Boxes

Next we consider the slightly more complex CIPHERTWO given in Figure 2. We assume that a cryptanalyst knows two message inputs m_0 and m_1 along with the ciphertexts c_0 and c_1. An attacker might try to attack CIPHERTWO in much the same way as CIPHERONE though he will soon encounter a problem. As before, we can work backwards and guess the value of k_2 to compute x_0 and x_1. This allows us to get w_0 and w_1, we know $w_0 \oplus w_1$ and hence we know $v_0 \oplus v_1$. Starting with the plaintext and working forwards we know m_0 and m_1 and we also know $u_0 \oplus u_1$ since this is equal to $m_0 \oplus m_1$. But this is of little help since the S-box is non-linear with respect to exclusive-or and this prevents us from deducing the value

$$c = S[S[m \oplus k_0] \oplus k_1] \oplus k_2$$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

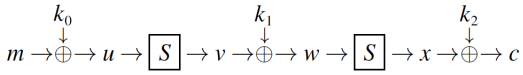


Figure 2. Encryption with CIPHERTWO

of the difference $v_0 \oplus v_1$. Thus an attacker cannot immediately verify a guess for k_2.

i	j	$S[i]$	$S[j]$	$S[i] \oplus S[j]$
0	f	6	b	d
1	e	4	9	d
2	d	c	a	6
3	c	5	8	d
4	b	0	d	d
5	a	7	3	4
6	9	2	f	d
7	8	e	1	f
8	7	1	e	f
9	6	f	2	d
a	5	3	7	4
b	4	d	0	d
c	3	8	5	d
d	2	a	c	6
e	1	9	4	d
f	0	b	6	d

Table 1. Inputs and output relations for i and $j = i \oplus \mathrm{f}$ across $S[\cdot]$.

To move forwards we need to understand the behavior of $S[\cdot]$. Consider two inputs of $S[\cdot]$, i and j, where j is the bitwise complement of i. Another way of saying this is to write $j = i \oplus \mathrm{f}$, where the latter is hexadecimal notation for the all-1 word. For all values of i we can compute j and we can compute the difference between the outputs of $S[i] \oplus S[j]$. The results are given in Table 1. It's important to note $S[i] \oplus S[j]$ has an uneven distribution of values: not all 16 possible values occur and some values occur more frequently than others. In this particular case the output different d occurs in 10 out of 16 cases.

Using this result we can do the following chosen plaintext attack on CIPHERTWO:

1. Pick a random message m_0 and set $m_1 = m_0 \oplus \mathrm{f}$. We can immediately see that $u_0 \oplus u_1 = m_0 \oplus m_1 = \mathrm{f}$.
2. Guess values for k_2 and compute $w_0 \oplus w_1 = v_0 \oplus v_1$.
3. We know that if u_0 takes all values with equal likelihood, and if $u_0 \oplus u_1 = \mathrm{f}$, then the probability that $v_0 \oplus v_1 = S[u_0] \oplus S[u_1] = \mathrm{d}$ is $\frac{10}{16}$. If we assume that an incorrect guess for k_2 leads to garbage with the result that $w_0 \oplus w_1$ "looks random", then attacker can gradually work out the correct value of k_2.
4. Once k_2 is known, for each message-ciphertext pair, we can compute $w = S^{-1}[c \oplus k_2]$. Now the rest of cipher is just a CIPHERONE. We can further attack it by guessing values of k_1.

In practice we can keep a series of counters $T_0, \ldots, T_{|k_2|-1}$, one for each possible value of k_2, which are initialized to 0 at the start of the attack. For any given message-ciphertext pair, we run through all possible value i to k_2 and if we get $v_0 \oplus v_1 = \mathrm{d}$ then we increment the counter T_i by 1. Note that the correct value of k_2 gives $v_0 \oplus v_1 = \mathrm{d}$ with probability $\frac{10}{16}$ while any incorrect value

of the key will give $v_0 \oplus v_1 = \mathrm{d}$ with probability $\frac{1}{16}$. If we use N message-ciphertext pairs, then the counter for the correct value of k_2 (which we might refer to as the *signal*) will be much larger than the counters for the other key values (the *noise*).

2.3 Characteristics

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	6	-	-	-	-	2	-	2	-	-	2	-	4	-
2	-	6	6	-	-	-	-	-	-	2	2	-	-	-	-	-
3	-	-	-	6	-	2	-	-	2	-	-	-	4	-	2	-
4	-	-	-	2	-	2	4	-	-	2	2	2	-	-	2	-
5	-	2	2	-	4	-	-	4	2	-	-	2	-	-	-	-
6	-	-	2	-	4	-	-	2	2	-	2	2	2	-	-	-
7	-	-	-	-	-	4	4	-	2	2	2	2	-	-	-	-
8	-	-	-	-	-	2	-	2	4	-	-	4	-	2	-	2
9	-	2	-	-	-	2	2	2	-	4	2	-	-	-	-	2
a	-	-	-	-	2	2	-	-	-	4	4	-	2	2	-	-
b	-	-	-	2	2	-	2	2	2	-	-	4	-	-	2	-
c	-	4	-	2	-	2	-	-	2	-	-	-	-	-	6	-
d	-	-	-	-	-	-	2	2	-	-	-	-	6	2	-	4
e	-	2	-	4	2	-	-	-	-	-	2	-	-	-	-	6
f	-	-	-	-	2	-	2	-	-	-	-	-	-	10	-	2

Table 2. The difference distribution table for $S[\cdot]$. There is a row for each input difference din and the frequency with which a given output difference d_{out} occurs is given across the row. The entry $(d_{\text{in}}, d_{\text{out}})$ divided by 16 gives the probability that a difference din gives difference d_{out} when taken over all possible pairs with difference d_{in}.

The key to extending our basic attack on CIPHERONE was given in Table 1. This mapped out the possible evolution of the difference f across $S[\cdot]$. In fact, we can do this for all possible input differences and this gives us a *difference distribution table for* $S[\cdot]$. This is present in Table 2. The table gave the total number of input pairs satisfying a given input difference (indexed by row) which leads to a pair of outputs with a given difference (indexed by column).

There are several general properties of the difference distribution table that should be mentioned. First, it should be noted that the sum of all elements in a row is $w^n = 16$; similarity the sum of any column is $2^n = 16$. Also, all element values are even; this results because a pair of input (or output) values represented as (X', X'') has the same ΔX value as the pair (X'', X') since $\Delta X = X' \oplus X'' = X'' \oplus X'$. As well, the input difference of $\Delta X = 0$ must lead to an output difference of $\Delta Y = 0$ for the one-to-one mapping of the S-box. Hence, the top left corner of the table has a value of $2^n = 16$ and all other values in the first row and first column are 0. Finally, if we could construct an ideal S-box, which gives no differential information about the output given the input value, the S-box would have all elements in the table equal to 1 and the probability of occurrence of a particular value for ΔY given a particular value of ΔX would be $1/2^n = 1/16$. However, as the properties discussed above must hold, this is clearly not achievable.

Definition 1. A pair (α, β) for which two inputs with difference α lead to two outputs with difference β is called a (differential) *characteristic* across the operation $S[\cdot]$. it will be denoted by $\alpha \xrightarrow{S} \beta$. A characteristic has a probability associated with it.

Now we introduce the latest version of our cipher, CIPHERTHREE in Fig 3. An attacker trying to mount the attack we used on CIPHERTWO will be able to compute the values y_0 and y_1 for a

$$c = S[S[S[m \oplus k_0] \oplus k_1] \oplus k_2] \oplus k_3$$

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

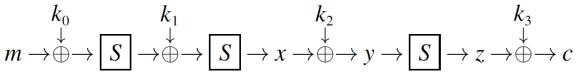


Figure 3. Encryption with CIPHERTHREE

correct guess for the value of k_3. They can readily go on to compute $x_0 \oplus x_1$. To verify whether such a guess for the value of k_3 is correct, the attacker will try to push information about the messages (which he can choose) through two applications of the S-box. Looking at Table 2 we saw that two messages satisfying the difference f will yield two outputs from $S[\cdot]$ with a difference d with a probabilitu of $\frac{10}{16}$.

If k_1 is a randomly chosen four-bit key, then the difference of d in the inputs to the second application of $S[\cdot]$ (a difference directly inherited as the output from the first $S[\cdot]$) will lead to $x_0 \oplus x_1 = \mathrm{c}$ with probability $\frac{6}{16}$. This is obtained by looking at Table 2. We can therefore combine the actions of the two S-boxes by using the characteristic $\mathrm{f} \xrightarrow{S} \mathrm{d}$ over the first $S[\cdot]$ and the characteristic $\mathrm{d} \xrightarrow{S} \mathrm{c}$ over the second $S[\cdot]$. If we assume that these characteristics are independent, then the probability of the two-round characteristic $\mathrm{f} \xrightarrow{S} \mathrm{d} \xrightarrow{S} \mathrm{c}$ is $\frac{10}{16} \times \frac{6}{16}$. Thus an attacker who chooses pairs of messages related by the difference f can expect the difference $y_0 \oplus y_1$ to take the value c with probability $\frac{15}{64}$. Given sufficient message-ciphertext pairs, it should be possible to find the key k_3 just as we did when attacking CIPHERTWO.

2.4 Joining Characteristics

To highlight the last major elements of a differential cryptanalytic attack we need a more sophisticated cipher. This cipher is called CIPHERFOUR and is now beginning to resemble a real-life cipher unlike our previous toy examples.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

i	0	1	2	3	4	5	6	7
$P[i]$	0	4	8	12	1	5	9	13
i	8	9	10	11	12	13	14	15
$P[i]$	2	6	10	14	3	7	11	15

1. Set $u_0 = m$.
2. For $i := 1$ to $r-1$ do:
   a. Combine the round key k_{i-1} with u_{i-1} so that $a_i = u_{i-1} \oplus k_{i-1}$.
   b. Divide a_i into four nibbles $a_i = A_0 || \ldots || A_3$.
   c. Compute $S[A_0] || \ldots || S[A_3]$ where $S[\cdot]$ is defined above.
   d. Write $S[A_0] || \ldots || S[A_3]$ as $y_{15} \ldots y_0 = S[A_0] || \ldots || S[A_3]$.
   e. Permute bit y_i to position j according to the permutation $j = P[i]$ defined above.
   f. Set $u_i = y_{15} || y_{11} || \ldots || y_4 || y_0$.
3. *(Last round)*
   a. Combine the round key k_{r-1} with u_{r-1} so that $a_r = u_{r-1} \oplus k_{r-1}$.
   b. Divide a_r into four nibbles $a_r = A_0 || \ldots || A_3$.
   c. Compute $S[A_0] || \ldots || S[A_3]$ where $S[\cdot]$ is defined above.
   d. Write $S[A_0] || \ldots || S[A_3]$ as $y = y_{15} \ldots y_0 = S[A_0] || \ldots || S[A_3]$.
   e. Output $y \oplus k_r$ as ciphertext.

Figure 4. Encryption with r rounds of CIPHERFOUR

CIPHERFOUR is an iterated cipher with r rounds that operates on 16-bit blocks. Both the message and the ciphertext blocks will be 16 bits long, as will the $r+1$ round keys which are independent

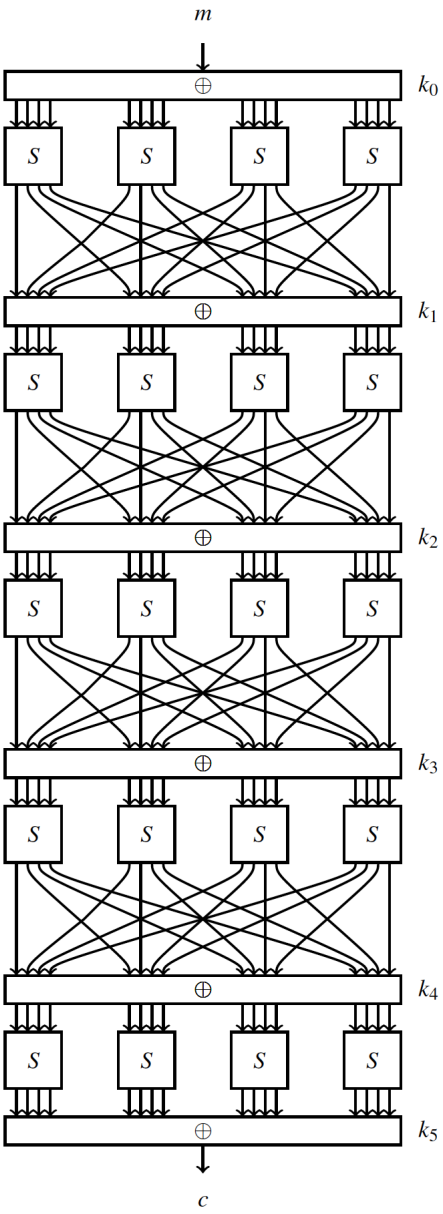


Figure 5. Encryption with five rounds of CIPHERFOUR

and chosen uniformly at random. Encryption with CIPHERFOUR is outlined in Fig 4 and illustrated in Fig 5. It can be shown that each bit of the ciphertext dependes on each bit of the pplaintext after four rounds of encryption and CIPHERFOUR appears to have many of the features of a real-life block cipher.

Let's consider a differential attack on CIPHERFOUR. First note that all components are linear with respect to the exclusive-or operation except the S-box. The S-box is the same as that used in CIPHERTHREE and so the distribution of differences across $S[\cdot]$ can be found in Table 2. If we are trying to mount an attack similar in style to that used on CIPHERTHREE, then the first thing we need to do is to find a characteristic which predicts the difference between two partially encrypted messages after $r-1$ rounds. If such a characteristic can be identified that holds with a sufficiently high probability, then, in principle, we should be able to find the round key k_r.

To build characteristics over several rounds, one normally tries to find one-round characteristics (of high probability) that can be joined together. So with CIPHERFOUR, we will need to identify input differences to four S-boxes for which high-probability output differences exist. Then, since $P[\cdot]$ is linear, one can easily compute the output difference from one round. Let us denote by

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \xrightarrow{S} (\beta_1, \beta_2, \beta_3, \beta_4)$$

the input and output differences to four S-boxes, where α_i is the difference between the values A_i input to the relevant S-box and β_i is the difference between the output values $S[A_i]$. Since the action of $P[\cdot]$ is fixed, we can predict with certainty how the difference will evolve across the permutation

$$(\beta_1, \beta_2, \beta_3, \beta_4) \xrightarrow{P} (\gamma_1, \gamma_2, \gamma_3, \gamma_4)$$

where $\beta_1||\beta_2||\beta_3||\beta_4$ is the 16-bit input difference to $P[\cdot]$ and $\gamma_1||\gamma_2||\gamma_3||\gamma_4$ is the 16-bit output difference from $P[\cdot]$. As a result, over the full round $\mathscr{R}$,

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \xrightarrow{\mathscr{R}} (\gamma_1, \gamma_2, \gamma_3, \gamma_4)$$

which denotes a nibble-based, one-round characteristic for CIPHERFOUR.

The key idea to break CIPHERFOUR, is to find out a *non-trivial iterative characteristic* in which the input difference equals the output difference for one round, thus the characteristic can be concatenated with itself to produce a characteristic over as many rounds as we would like. One possible choice is the following one-round characteristic

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

which holds with probability $\frac{6}{16}$. Thus we have that

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

is a two-round characteristic of probability $(\frac{6}{16})^2$,

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

is a three-round characteristic of probability $(\frac{6}{16})^3$, and

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

is a four-round characteristic of probability $(\frac{6}{16})^4$ and so on. Note that, to estimate the probability of our characteristic we have assumed that its evolution over successive rounds is independent. In this way the cumulative probability can be computed as the product of probabilities of one-round characteristics.

Now we can consider attacking five rounds of CIPHERFOUR. But if we just use the last four round characteristic

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

which has a (conjectured) probability $(\frac{6}{16})^4 \simeq 0.02$, while the probability of any given difference occurring at random is $\frac{1}{16} \simeq 0.06$. The probability of the characteristic we have identified could be too small to effectively distinguish the correct key information.

Fortunately there're at least three other characteristics which has the same start and end differences:

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,0,2) \xrightarrow{\mathscr{R}} (0,0,0,1) \xrightarrow{\mathscr{R}} (0,0,1,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,0,2) \xrightarrow{\mathscr{R}} (0,0,1,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

$$(0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,2,0) \xrightarrow{\mathscr{R}} (0,0,0,2) \xrightarrow{\mathscr{R}} (0,0,1,0) \xrightarrow{\mathscr{R}} (0,0,2,0)$$

Note that an attacker cannot verify whether a specific message pair starting with a given difference actually gives the desired difference after every round (as specified by the characteristic). We say that any pair that follows that characteristic is a *right pair*. A

pair which deviates from the characteristic at some point is called a *wrong pair*.

It's important to note that the attacker is only concerned with the difference in the partially encrypted inputs after four rounds. The differences in the first, second, and third rounds are not used. Indeed, since the attacker does not (yet) know the values of the subkeys in these rounds, he has no way of determining these differences. So in our example, instead of using the four-round characteristic in full detail the attacker is really only interested in the probability that the following happens:

$$(0,0,2,0) \xrightarrow{\mathscr{R}} ? \xrightarrow{\mathscr{R}} ? \xrightarrow{\mathscr{R}} ? \xrightarrow{\mathscr{R}} (0,0,2,0)$$

where ? indicates an unknown (and irrelevant) value. Such a structure is called a *differential* and a differential may contain many characteristics, all starting and ending with the same difference. In our particular case, we don't know the exact probability that this differential holds, but we do know that at least four characteristics in it all hold with a probability of $(\frac{6}{16})^4$. This suggests that the probability that the differential holds will be at least as large as the sum of the probabilities of the four characteristics, i.e. $4 \times (\frac{6}{16})^4 \simeq 0.08$, now it's higher enough to be identified.

2.5 Filtering

The basic aim of differential cryptanalysis is to identify a statistically unusual distribution in the differences that occur. This is the signal we are trying to detect, and it can be masked by many other pairs that are not following the anticipated characteristic or differential. It can therefore be very desirable to eliminate such wrong pairs from consideration as soon as possible. This makes searching for the signal that much easier.

It is often possible to identify wrong pairs–pairs that have not followed a characteristic–by looking at the ciphertext alone. To illustrate this technique of *filtering* we continue with CIPHERFOUR. We now examine closely how differences propagate through the 5th (and final) round of the cipher when the plaintext pair is a right pair. In the 5th round we have

$$(0,0,2,0) \xrightarrow{\mathscr{S}} (0,0,h,0),$$

where h can take any of the values in $\{1,2,9,\mathrm{a}\}$. Since $P[\cdot]$ is not used in the last round, these four difference values must become part of the ciphertext after five rounds. Therefore, for each message pair, the attacker can inspect the corresponding ciphertext pair and immediately determine whether a pair is a wrong pair or, potentially, a right pair. In our example, 12 bits of the difference in the ciphertexts must be zero, and the remain four bits can take only four specific values. The process of discarding wrong pairs is called *filtering* and a good filtering technique is essential for the success of many differential attacks. Experiments on random chosen keys in CIPHERFOUR have shown that, on average about 11% of all message-ciphertext pairs will survive the filtering process.

2.6 Recovering Key Information

We can now put all the pieces together and construct a key recovery attack on CIPHERFOUR. In our attack we will only recover four bits of the key k_5 used in the last round. These bits correspond to the four key bits that affect the S-box in the fifth encryption round, for which the output difference is nonzero. We call these key bits the *target* key bits.

From previous results, we believe that the four-round differential

$$(0,0,2,0) \xrightarrow{\mathscr{R}} ? \xrightarrow{\mathscr{R}} ? \xrightarrow{\mathscr{R}} ? \xrightarrow{\mathscr{R}} (0,0,2,0)$$

holds with a probability of 0.08. And we assume that a pair survives the filtering process with probability 0.11. Suppose that the cryptanalyst receives the encryption of t message pairs which satisfy the starting difference $(0,0,2,0)$. From the ciphertext difference alone, the attacker can filter out pairs that pairs that are certainly wrong. If the plaintexts are randomly chosen, one expects $t \times 0.11$ pairs to survive filtering. Among these we will certainly have the pairs that satisfy the differential and there will be $t \times 0.08$ such pairs.

Now we know that as we try out each value to the target bits, the correct value will yield the expected difference for pairs that satisfy the differential. This means that pairs satisfying the differential will always suggest the right value of the target bits. Thus, with t pairs of texts in the attack, the correct value of the target bits will be suggested $t \times 0.08$ times. On the other side, we assume that target bits with an incorrect value give the expected difference (as specified by the differential) with a probability of $\frac{1}{16}$. Then we expect a pair not satisfying the differential but still passing the filter to suggest one (incorrect) value of the target bits on average. Thus, over t chosen message pairs, we wound expect roughly $t \times (0.11 - 0.08) = t \times 0.03$ incorrect values for the target bits to be suggested. By using sufficiently many messages we expect the correct value of the target bits to be the most suggested value.

Our attack finds 4 key bits from a five-round version of CIPHERFOUR. To find other key bits one can implement a similar differential attack using other differential. It might also be possible to determine some key bits in the first round of the cipher. Once all 16 bits of the first subkey or the last subkey are determined, the attacker can peel off one round of the cipher and continue his attack on a weaker versions. In a differential attack the most difficult part is to find *any* secret key bits. Once some key bits have been found, it is rarely hard to recover the rest.

3. Differential Cryptanalysis on DES Variants

3.1 Notations and Definitions

To apply differential cryptanalysis on DES, we first introduce the following notations:

- *The numbers:* An hexadecimal number n is denoted with the subscript x as n_x (e.g., $10_x = 16$). Decimal numbers are denoted without subscripts.
- *The plaintext:* The plaintext is denoted by P. In the discussion on DES, we ignore the existence of the initial permutation of DES, and thus P is the value after the initial permutation which is entered directly into the first round. In differential cryptanalytic attacks the plaintexts are used in pairs. The other plaintext in the pair is denoted by P^*. The left and the right halves of the plaintext P are denoted by P_L and P_R respectively (i.e., $P = (P_L, P_R)$).
- *The ciphertext:* The ciphertext is denoted by T. Since we ignore the existence of the initial permutation of DES, T is the value before the inverse initial permutation IP^{-1}. The ciphertext of the second plaintext P^* is denoted by T^*. The left and the right halves of the ciphertext T are denoted by T_L and T_R respectively (i.e., $T = (T_L, T_R)$).
- *The difference:* At any intermediate point during the encryption of pairs of plaintexts, if X denotes a value during the encryption of the first plaintext, X^* denotes the corresponding value during the encryption of the second plaintext. The difference of these values is denoted by X'. For DES-like cryptosystems we define $X' = X \oplus X^*$.
- *The inputs and the outputs of the F function:* The 32-bit inputs of the F function in the various rounds are denoted by the lowercase letters $a, b, \ldots, j$. The corresponding 32-bit outputs

of the F function in the various rounds are denoted by the uppercase letters $A, B, \ldots, J$.

- The initial permutation: The initial permutation of DES is denoted by $\mathrm{IP}(X)$. In differential cryptanalysis the existence of the initial permutation IP and the inverse initial permutation IP^{-1} of DES are ignored, since they have no cryptanalytic significance in our attack.
- *The subkeys:* The F function of each round has a unique key dependent input, called the *subkey*. DES iterates the round-function 16 times and uses 16 subkeys, named $K1, K2, \ldots, K16$. All the bits of the subkeys are chosen by the key scheduling algorithm of DES by duplicating each bit of the 56-bit key into about 14 out of the 16 48-bit subkeys.
- *The P permutation:* The P permutation of DES is denoted by $P(X)$.
- *The S-boxes:* The S-boxes of DES are S1, S2, ..., S8. The input of the S-box Si in the round whose input letter is X ($X \in \{a, \ldots, j\}$) is denoted by Si_{IX}. The corresponding output of Si is denoted by Si_{OX}. The value of the six bits of the subkey entering the S-box Si after they are XORed with the expanded data is denoted by Si_{KX} and the value of the six input bits of the expanded data ($E(X)$) which are XORed with Si_{KX} to form Si_{IX} is denoted by Si_{EX}. (See Figure 6)

Definition 2. An *n-round characteristic* is a tuple $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ where Ω_P and Ω_T are m bit numbers and Ω_Λ is a list of n elements $\Omega_\Lambda = (\Lambda_1, \Lambda_2, \ldots, \Lambda_n)$, each of which is a pair of the form $\Lambda_i = (\lambda_I^i, \lambda_O^i)$ where λ_I^i and λ_O^i are $m/2$ bit numbers and m is the block size of the cryptosystem. A characteristic satisfies the following requirements:

$$\begin{aligned}\lambda_I^1 &= \text{the right half of } \Omega_P\\ \lambda_I^2 &= \text{the left half of } \Omega_P \oplus \lambda_O^1\\ \lambda_I^n &= \text{the right half of } \Omega_T\\ \lambda_I^{n-1} &= \text{the left half of } \Omega_T \oplus \lambda_O^n\end{aligned}$$

and for every i such that $2 \le i \le n-1$:

$$\lambda_O^i = \lambda_I^{i-1} \oplus \lambda_I^{i+1}$$

Definition 3. A *right pair with respect to an n-round characteristic $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ and an independent key K* is a pair for which $P' = \Omega_P$ and for each round i of the first n rounds of the encryption of the pair using the independent key K the input difference of the i^{th} round equals λ_I^i and the output difference of the F function equals λ_O^i. Every pair which is not a right pair with respect to the characteristic and the independent key is called a *wrong pair with respect to the characteristic and the independent key*. We refer them shortly by *right pair* and *wrong pair*.

Definition 4. An n-round characteristic $\Omega^1 = (\Omega_P^1, \Omega_\Lambda^1, \Omega_T^1)$ can be concatenated with an m-round characteristic $\Omega^2 = (\Omega_P^2, \Omega_\Lambda^2, \Omega_T^2)$ if Ω_T^1 equals the swapped value of the two halves of Ω_P^2. The *concatenation* of the characteristic Ω^1 and Ω^2 (if they can be concatenated) is the characteristic $\Omega = (\Omega_P^1, \Omega_\Lambda, \Omega_T^2)$ where Ω_Λ is the concatenation of the lists Ω_Λ^1 and Ω_Λ^2.

The following definitions and theorem deal with the probability of characteristics:

Definition 5. Round i of a characteristic Ω has probability p_i^Ω if $\lambda_I^i \to \lambda_O^i$ with probability p_i^Ω by the F function.

Definition 6. An n-round characteristic Ω has probability p^Ω if p^Ω is the product of the probabilities of its n rounds:

$$p^\Omega = \prod_{i=1}^{n} p_i^\Omega.$$

Note that by Definition 5 and 6 the probability of a characteristic Ω which is the concatenation of the characteristic Ω^1 with the characteristic Ω^2 is the product of their probabilities: $p^\Omega = p^{\Omega^1} \cdot p^{\Omega^2}$. As a result, every n-round characteristic can be described as the concatenation of n one-round characteristics with probability which is the product of the one-round characteristic with probability which is the product of the one-round probabilities.

The following theorem was proven by Biham and Shamir in [4]:

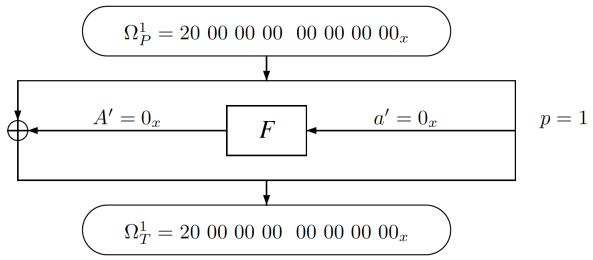
Theorem 7. *The formally defined probabilities of a characteristic $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ is the actual probability that any fixed plaintext pair satisfying $P' = \Omega_P$ is a right pair when random independent keys are used.*

Definition 8. A characteristic $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ is called an *interactive characteristic* is it can be concatenated with itself.

Wen can concatenate an interactive characteristic to itself any number of times and can get characteristic with an arbitrary number of rounds. 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. In the next subsections we briefly explain that how Biham and Shamir actually attacked DES and its variants using above results.

3.2 Attacks on DES Reduced to Four Rounds

In this attack we used the following one-round characteristic Ω^1 with probability 1, where in the second round (if added) $b' = 20\ 00\ 00\ 00_x$:



In the first round the characteristic has $a' = 0 \to A' = 0$ with probability 1. The single bit difference between the two plaintexts starts to play a role in the second round in S1. Since the inputs to S1 differ only in one bit, at least two output bits must differ. Typically such two bits enter three S-boxes in the third round ($c' = a' \oplus B' = B'$), where there is a difference of one bit in each S-box input. The 28 outputs difference bits of S2, ..., S8 in B' must be all zeros, since their input differences are zero. The value of D' can be derived from a', B' and T_L' by the equation

$$D' = a' \oplus B' \oplus T_L' = B' \oplus T_L'. \tag{9}$$

When the ciphertext pair values T and T^* are known then d and d^* are known to be their right halves (by $d = T_R$). Since a', T_L' and the 28 bits of B' are known, the corresponding 28 bits of D' are known as well by Equation 9. These 28 bits are the output differences of the S-boxes S2, ..., S8. Thus, we know the values S_{Ed}, S_{Ed}^* and S_{Od}' of seven S-boxes in the fourth round.

Given four encrypted pairs they used a separated counting procedure for each one of the seven S-boxes in the fourth round. They

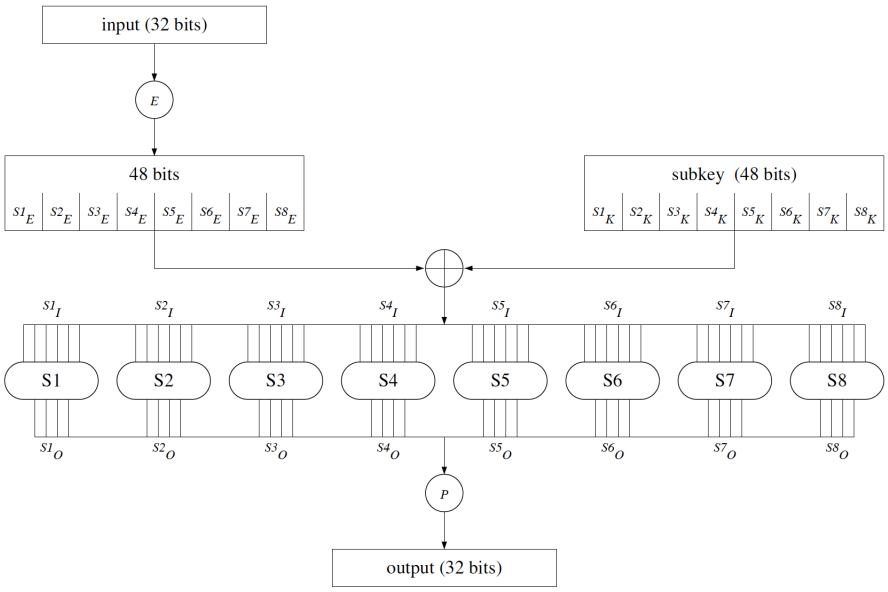


Figure 6. The F function of DES

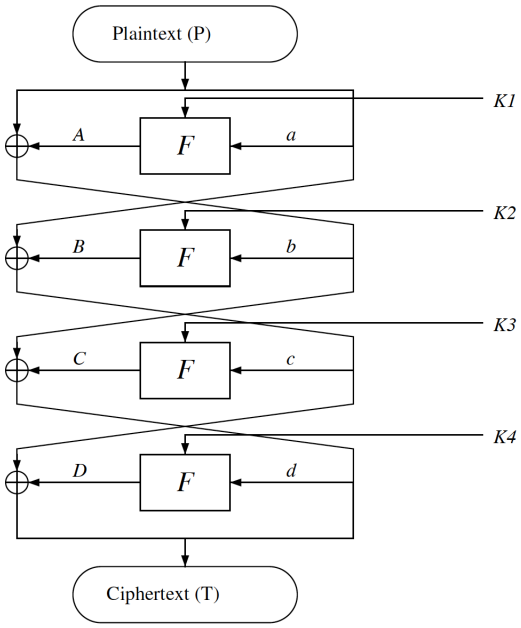


Figure 7. DES reduced to four rounds.

try all the 64 possible values of S_{Kd} and check whether

$$S(S_{Ed} \oplus S_{Kd}) \oplus S(S^*_{Ed} \oplus S_{Kd}) = S'_{Od}.$$

For each key they count the number of pairs for which the test succeeds. The right key value is suggested by all the pairs since they use a characteristic with probability 1, for which all the pairs are right pairs. The other 63 key values may occur in some of the pairs. It is unlikely that a value occurs in all the pairs, which have various values of S'_E and S'_O.

So far they have found $7 \cdot 6 = 42$ bits of the subkey of the last round ($K4$). If the subkeys are calculated via the key scheduling algorithm of DES, these are 42 actual key bits out of the 56 key bits, and thus 14 key bits are still missing. One can now try all the 2^{14} possibilities of the missing bits and decrypt the given ciphertexts using the resulting keys. The right key should satisfy the known plaintext difference value for all the pairs, but the other $2^{14} - 1$ values have only probability 2^{-64} to satisfy this condition.

3.3 Attacks on DES Reduced to Eight Rounds

DES reduced to eight rounds can be broken using about 25000 ciphertext pairs for which the plaintext difference is

$$P' = 40\ 5C\ 00\ 00\ 04\ 00\ 00\ 00_x.$$

The method finds 30 bits of K8. 18 additional key bits can be found using similar manipulations on the pairs. The remaining eight key bits can be found using exhaustive search. See Figure 8 for the details of the characteristic.

This characteristic has probability $\frac{1}{10485.76}$. The input difference in the 6th round of a right pair is $f' = 40\ 5C\ 00\ 00_x$. Consequently, for five S-boxes $S'_{Ef} = S'_{If} = 0$ and $S'_{Of} = 0$.

In right pairs, the five S-boxes S2, S5, S6, S7 and S8 satisfy $S'_{Ef} = S'_{If} = 0$ and $S'_{Of} = 0$. By the formula $H' = T'_L \oplus g' = T'_L \oplus e' \oplus F'$ we can find the output differences of the corresponding S-boxes in the 8th round. The input data of the 8th is known from the ciphertexts. Therefore, we can se the counting method to find the 30 subkey bits entering the five S-boxes at the 8th round. The signal to noise ratio of this counting scheme is $S/N = \frac{2^{30}}{4^5 \cdot 10485.76} = 100$.

Counting on 30 subkey bits demands a huge memory of 2^{30} counter. We can reduce the amount of memory by counting on fewer subkey bits entering fewer S-boxes. The remaining S-boxes can be used for identification of some of the wrong pairs (in which $S'_{Eh} \nrightarrow S'_{Oh}$). In counting schemes that count on a reduced number of bits we can choose the reduced set of countable S-boxes arbitrarily. In this particular case we can choose the reduced set in a way which maximizes the characteristic's probability and the signal to noise ratio by using a slightly modified characteristic which ignores output bits that are not counted anyway. The slightly modified characteristic is similar to the original one except that in the 5th round only one bit of $S2'_{Oe}$ is fixed and all the combinations of the other

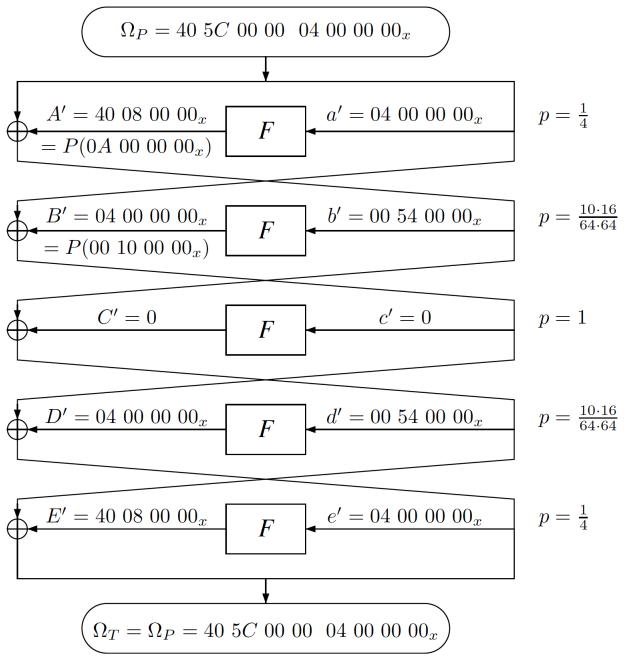


Figure 8. The five-round characteristic used for attacking DES reduced to eight rounds

three are allowed:

$$e' = 04\ 00\ 00\ 00_x \rightarrow E' = P(0\mathrm{W}\ 00\ 00\ 00_x) = \mathrm{X0\ 0Y\ Z0}\ 00_x,$$

where $W \in \{0, 1, 2, 3, 8, 9, \mathrm{A}, \mathrm{B}\}, X \in \{0, 4\}, Y \in \{0, 8\}$ and $Z \in \{0, 4\}$. Therefore at the 6th round $f' = \mathrm{X0\ 5V\ Z0}\ 00_x$, where $V = Y \oplus 4$. The only possible combination in which $Z = 0$ is $04\ 00\ 00\ 00_x \rightarrow 40\ 08\ 00\ 00_x$ which has probability $\frac{16}{64}$. All the other combinations (in which $Z = 4$) have an overall probability $\frac{20}{64}$. We cannot count on the subkey bits $S5_{Kh}$ but it is still advisable to check the possibility of $S5'_{Eh} \rightarrow S5'_{Oh}$ which is satisfied by 80% of the pairs. Therefore, the probability of $e' \rightarrow E'$ is $\frac{16}{64} + 0.8\frac{20}{64} = \frac{32}{64} = \frac{1}{2}$. The probability of the five-round modified characteristic is $\frac{16 \cdot 10 \cdot 16}{64^3} \cdot \frac{16 \cdot 10 \cdot 32}{64^3} \simeq \frac{1}{5243}$. The signal to noise ratio of a counting scheme which counts on 18 subkey bits entering three S-boxes out of S2, S6, S7 and S8 is $S/N = \frac{2^{18}}{4^3 \cdot 0.8^2 \cdot 5243} \simeq 1.2$. This 18-bit counting scheme needs 150000 pairs and has an average of about 24 counts for any wrong key value and about 53 counts for the right key value ($53 = 24 + \frac{150000}{5243} = 24 + 29$).

Below is a summary of this cryptanalytic method, which can be easily implemented on a personal computer:

1. Set up an array of $2^{18} = 256K$ single-byte counters which is initialized by zeros. The array corresponds to the 2^{18} values of the 18 key bits of K8 entering S6, S7 and S8.
2. Preprocess the possible values of S_I that satisfy each $S'_I \rightarrow S'_O$ for the eight S-boxes into a table. This table is used to speed up the program.
3. For each ciphertext pair do:
   (a) Assume $h' = T'_R, H' = T'_L$ and $h = T_R$. Calculate $S'_{Eh} = S'_{Ih}$ and S'_{Oh} for S2, S5, S6, S7 and S8 by h' and H'. Calculate S_{Eh} for S6, S7 and S8 by h.
   (b) For each one of the S-boxes S2, S5, S6, S7 and S8, check if $S'_{Ih} \nrightarrow S'_{Oh}$. If $S'_{Ih} \nrightarrow S'_{Oh}$ for at least one of the S-boxes then discard the pair as a wrong pair.
   (c) For each one of the S-boxes S6, S7 and S8: fetch from the preprocessed table all the values of S_{Ih} which are possible for $S'_{Ih} \nrightarrow S'_{Oh}$. For each possible value calculate $S_{Kh} = S_{Ih} \oplus S_{Eh}$. Increment by one the counters corresponding to all the possible 18-bit concatenations of the one six-bit value suggested for $S6_{Kh}$, one six-bit value suggested for $S7_{Kh}$ and one six-bit value suggested for $S8_{Kh}$.
4. Find the entry in the array that contains the maximal count. The entry index is likely to be real value of $S6_{Kh}$, $S7_{Kh}$ and $S8_{Kh}$ which is the value of the 18 bits (number $31, \ldots, 48$) of K8.

For the method of finding more key bits, please refer to [4]. As we said before, once some key bits have been found, it is rarely hard to recover the rest.

Noticed that, the attacking process heavily depends on some special differentials which is highly related to the values inside each S-boxes. S-box is inevitable to have differentials. However, finding a circle differential with high probability is not trivially easy. But the key is, no matter how long time an attacker spent on find such a differential, even 10 years, once it's found, the target cryptosystem is then breakable. In another words, the time spent on finding differentials is irrelevant to the time spent on finding a key on a number of plaintext-cipher pairs. It's a one-time job.

3.4 Differential Cryptanalysis on Modified DES Variants

DES is extremely well designed, not only in its overall structure but also in the choice on its P permutation and S-boxes, and many other details. However, this fact was only really understood when applying differential cryptanalysis on modified variants of DES. This was the one of the main contribution of [4], beside the recovering of differential cryptanalysis. Here we just briefly review the results with little details.

First of all, the choice of the P permutation has a major influence on the existence of high probability characteristics. Many modifications of the P permutation would weaken the variants of DES. An extreme case is when the P permutation is replaced by the identity permutation (or eliminated). In this case, the two middle output bits of each S-box would enter as the two middle (private) bits of the same S-box in the following round, and this would give rise to an three-round iterative characteristic on difference $00\ 00\ 00\ 00\ 00\ 60\ 00\ 00_x$. This attack requires up to 2^{20} pairs. On the other side, attacks based on characteristics in which the output differences of all the F functions are zero, are not influenced by the choice of the P permutation. Therefore, all the attacks based on the iterative characteristic are independent of the choice of the P permutation and thus the replacement of the P permutation by any other permutation cannot make DES stronger.

The DES cryptosystem specifies a certain order of the eight S-boxes. Even a modification of the order of the S-boxes can make the cryptosystem much weaker. Consider for example the case in which S1, S7 and S4 are brought together in this order (without loss of generality, in the first three S-boxes entries) and the other S-boxes are set in any order. Then there is a similar two-round iterative characteristic, denoted by $\phi^* = 1\mathrm{D}\ 40\ 00\ 00_x$ whose probability is about $\frac{1}{73}$. Such DES variants can be attacked using about 2^46 chosen plaintexts, which means much weaker than the original DES.

In a random S-box there is a very high probability (about 0.998) that there are two different inputs that differ in the two middle input bits of an S-box (which do not affect the neighboring S-boxes) which have the same output. In this case there is an iterative characteristic with high probability, for S1 it's $60\ 00\ 00\ 00\ 00\ 00\ 00\ 00_x$.

The security of DES can be devastated even by minor modifications of the S-boxes. With a single modification in one entry of one of the original S-boxes of DES we can force this S-box to have two inputs which differ only in one private input bit of the S-box and have the same output. For example, such a modification may set the value of $S1(4)$ to be equal to $S1(0)$ on. Then, the two inputs 0 and 4 have the same output, and thus the probability of $4 \to 0$ by this S-box is $\frac{1}{32}$. A two-round iterative characteristic based on this property has probability $\frac{1}{32}$ and is 20 00 00 00 00 00 00 00_x, only 2^{37} pairs are required to attack the 16-round modified DES.

Finally, a cryptosystem similar to DES in which the E expansion is eliminated and the S-boxes map four bits to four bits is quite weak. Even the cryptosystems that use permutations derived from the original S-boxes are easily attacked. We omit the details.

4. The Theory of S-Boxes

4.1 Design Criteria of DES S-Boxes

In the rest of this paper we will focus on S-boxes, since the security of DES depends on the S-boxes because these are the only nonlinear components of the cipher.

After the publication of differential cryptanalysis [2] it became clear that both the S-boxes as well as the expansion E and permutation P were designed to increase the resistance of DES to this attack. This was confirmed by D. Coopersmith, one of the original DES designers at IBM, in [5]. Although the design criteria finally revealed by D. Coppersmith, one of the original DES designer, is not directly based on the characteristics of the differential distribution table of S-boxes, with differential cryptanalysis kept in mind, it's not hard to imagine each of these design criteria have made cryptanalysis harder.

Here we quote all the design criteria on DES S-boxes:

(S-1) Each S-box has six bits of input and four bits of output. (This was the largest size that we could accommodate and still fit all of DES onto a single chip in 1974 technology.)

(S-2) No output bit of an S-box should be too close to a linear function of the input bits. (That is, if we select any output bit position and any subset of the six input bit positions, the fraction of inputs for which this output bit equals the XOR of these input bits should not be close to 0 or 1, but rather should be near 1/2.)

(S-3) If we fix the leftmost and rightmost input bits of the S-box and vary the four middle bits, each possible 4-bit output is attained once as the middle four input bits range over their 16 possibilities.

(S-4) If two inputs to an S-box differ in exactly one bit, the outputs must differ in at least two bits. (That is, if $|\Delta I_{i,j}| = 1$, then $|\Delta O_{i,j}| \geq 2$, where $|x|$ is the number of 1-bits in the quantity x.)

(S-5) If two inputs to an S-box differ in the two middle bits exactly, the outputs must differ in at least two bits. (If $\Delta I_{i,j} = 001100$, then $|\Delta O_{i,j}| \geq 2$.)

(S-6) If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must not be the same. (If $|\Delta I_{i,j}| =$ 11xy00, where x and y are arbitrary bits, then $\Delta O_{i,j} \neq 0$.)

(S-7) For any nonzero 6-bit difference between inputs, $\Delta I_{i,j}$, no more than eight of the 32 pairs of inputs exhibiting $\Delta I_{i,j}$ may result in the same output difference $\Delta O_{i,j}$.

(S-8) Similar to (S-7), but with stronger restrictions in the case $\Delta O_{i,j} = 0$, for the case of three active S-boxes on round i.

In addition, the following criteria for the permutation P are also highly related to S-boxes:

(P-1) The four output bits from each S-box at round i are distributed so that two of them affect (provide input for) "middle bits" of S-boxes at round $i+1$ (the two middle bits of input to an S-box, not shared with adjacent S-boxes), and the other two affect "end bits" (the two left-hand bits or the two right-hand bits, which are shared with adjacent S-boxes).

(P-2) The four output bits from each S-box affect six different S-boxes; no two affect the same S-box. (Remember that each "end bit" affects two adjacent S-boxes.)

(P-3) For two S-boxes j, k, if an output bit from S_j affects a middle bit of S_k, then an output bit from S_k cannot affect a middle bit of S_j. This implies that in the case $j = k$, an output bit from S_j must not affect a middle bit of S_j.

Among those design criteria on S-boxes, (S-1), i.e. the choice of relative small S-boxes, seems related to hardware limitations in 1974. Since DES was designed for hardware implementation, it was deemed important that the cipher components was chosen so that the number of logical circuits would be kept at a minimum. However, choosing relative small S-boxes (6-to-4) has some other benefits. According to page 25 of [10], "early in the design process a pool of possible S-boxes was generated according to the design criteria developed thus far and these boxes were then ranked according to how efficiently they could be implemented. As the process evolved and more constraints were put on the S-boxes, the pool of boxes shrunk and the complexity of the most efficient ones increased. In the end there were three S-boxes with the same most-efficient implementation and seven S-boxes with the same second most-efficient implementation. *It seems that the eight S-boxes for DES were chosen from this set of ten*". The rareness of candidate S-boxes also exposed the fact that designing a secure block cipher is so difficult.

One interesting coincidence is that, it's possible to use such a pool only when the S-boxes are small. To further understand DES S-boxes, especially how large the above mentioned "pool" was, we need some knowledge from mathematics, i.e. the theory of boolean functions.

4.2 S-Box and Boolean Functions

Boolean functions are $\{0,1\}$-valued functions of a finite number of $\{0,1\}$-valued variables [7]:

Definition 10. A Boolean function of n variables is a *function on $\mathcal{B}^n$ into $\mathcal{B}$, where $\mathcal{B}$ is the set $\{0,1\}$, n is a positive integer; and $\mathcal{B}^n$ denotes the n-fold cartesian product of the set $\mathcal{B}$ with itself. A point $X^* = (x_1, x_2, \ldots, x_n) \in \mathcal{B}^n$ is a true point (respectively, false point) of Boolean function f if $f(X^*) = 1$ (respectively, $f(X^*) = 0$). We denote by $\mathbb{1}_n$ the function that takes constant value 1 on $\mathcal{B}^n$ and by 0_n the function that takes constant value 0 on $\mathcal{B}^n$.*

There're deep connections between Boolean functions and Cryptography [6]:

Definition 11. Let n and m be two positive integers. The functions from $\mathcal{B}^n$ to $\mathcal{B}^m$ are called (n,m)*-functions*. Such function F being given, the Boolean functions $f_1, \ldots, f_m$ defined, at every $x \in \mathcal{B}^n$, by $F(x) = (f_1(x), \ldots, f_m(x))$, are called the *coordinate functions* of F. When the numbers m and n are not specified, (n,m)-functions are called *multioutput Boolean functions*, *vectorial Boolean functions*, or *S-boxes*.

Therefore S-boxes can be designed by appropriate composition of nonlinear Boolean functions. However, n is rarely large. For

reason of efficiency, the S-boxes used in most block ciphers are concatenations of sub S-boxes on at most eight variables.

n	4	5	6	7	8
$\lvert\mathcal{BF}_n\rvert$	2^{16}	2^{32}	2^{64}	2^{128}	2^{256}
$\approx$	$6 \cdot 10^4$	$4 \cdot 10^9$	10^{19}	10^{38}	10^{77}

Table 3. Number of n-variable Boolean functions

Despite the fact that Boolean functions are currently used in cryptography with low numbers of variables, determining and studying those Boolean functions satisfying the desired conditions is not feasible through and exhaustive computer investigation: the number $|\mathcal{BF}_n| = 2^{2^n}$ of n-variable Boolean functions is too large when $n \geq 6$. In Table 3, we give the values of this number for n ranging between 4 and 8. Assume that visiting an n-variable Boolean function, and determining whether it has the desired properties, requires one nanosecond (10^{-9} seconds). Then it would take millions of hours to visit all functions in six variables, and about 100 billion times the age of the universe to visit all those in seven variables. The number of eight-variable Boolean functions approximately equals the number of atoms in the whole universe! We see that trying to find functions satisfying the desired conditions by simply picking up functions at random is also impossible for these values of n, because visiting a nonnegligible part of all Boolean functions in seven or more variables is not feasible, even when parallelizing. Thus the study of Boolean functions for constructing or studying codes or ciphers is essentially mathematical.

The S-boxes of DES are $(6, 4)$-functions, thus it sounds like the original designers have create a pool of 6-variable Boolean functions, which is huge (and impossible for computers in 1970s). However, although Coppersmith didn't reveal this part at all, we have reasons to believe that, the actual pool created in IBM was made of 4-variable Boolean functions, which are totally just $2^{2^4} = 65536$ choices (for single output bit). This is because DES was based on Lucifer, a substitution/permutation cryptosystem designed by IBM prior to the design of DES. From the differential cryptanalysis on CIPHERTWO and some DES variants, we can see that, if a block cipher is non-invertible, it's more difficult to perform differential cryptanalytic attacks. By extending existing $(4, 4)$-functions (S-boxes of Lucifer) into $(6, 4)$-functions (S-boxes of DES), the attacking difficulties on DES have greatly increased.

On the other side, AES S-box is a $(8, 8)$-function. It's clearly generated by mathematical methods [10, Chapter 8], It's absolutely impossible to search "in the whole universe" for a candidate.

4.3 S-Box Properties

Soon after the publication of DES it was realized that the DES S-boxes had been carefully chosen. Later, with the advent of differential cryptanalysis, we found out just how inspired their design had been. In this final part of this paper, we provide a quick overview of some of the properties that have been proposed, and we find it easy to classify them into two groups:

1. There are properties that reflect the propagation of difference that is induced by an S-box. These include the *strict avalanche criterion (SAC)* which stipulates that if we complement any single input bit to an S-box, then any of the output bits should flip with probability $\frac{1}{2}$.
2. There are properties that reflect the algebraic structure of an S-box. These include issues such as the *algebraic degree*, the *non-linear degree*, and what has been termed the *IO degree*.

Most, if not all of these properties, can be explained and manipulated using combinatorial techniques and the study of Boolean functions.

Let $f(x)$ be a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ that maps n-bit inputs to a single bit of output. We will denote the binary representation of an input x by $x_{n-1}x_{n-2}\ldots x_1x_0$. It is well known that we can write the output of the function as a (unique) multivariate expression in terms of the n input bits:

$$f(x) = \sum_{b\in\{0,1\}^n} A_f(b) x_{n-1}^{b_{n-1}} x_{n-2}^{b_{n-2}} \cdots x_1^{b_1} x_0^{b_0}.$$

Here $b_{n-1}b_{n-2}\ldots b_1b_0$ denotes the binary representation of b and determines which input bits are involved in a particular summand. The sum itself is computed modulo 2 and A_f is another Boolean function. If we consider this representation we see that the constant term of this summation is given by $A_f(0,\ldots,0)$ and the coefficient to the term $x_{n-1}^{b_{n-1}} x_{n-2}^{b_{n-2}} \cdots x_1^{b_1} x_0^{b_0}$ is given by $A_f(b_{n-1}b_{n-2}\ldots b_1b_0)$.

This representation of the Boolean function $f(x)$ is known as the *algebraic normal form*. An important quantity is the maximum Hamming weight of $b \in \mathcal{B}^n$ for which $A_f(b) = 1$. This is known as the (algebraic) degree of f and will be denoted by $\mathrm{AD}(f)$. The algebraic degree gives a measure of how many input bits might simultaneously have an impact on the value of the output. Intuitively we might expect a high algebraic degree to be a desirable attribute for an S-box, but things are rarely so clear-cut and trade-offs may well come into play. Here is an example of the algebraic normal form.

Let $f\colon \mathcal{B}^3 \to \mathcal{B}$ be a Boolean function defined by the following look-up table:

x	0	1	2	3	4	5	6	7
$f(x)$	1	0	1	0	1	0	1	1

The algebraic normal form can be derived as $f(x) = x_2x_1x_0 + x_0 + 1$ and so the algebraic degree $\mathrm{AD}(f)$ is 3.

We can extend these ideas to a general (n, m)-function as instantiated by an S-box S. One can view such a function as the concatenation of m Boolean functions, so that the evaluation of $S[x]$ with input x is given by the evaluation of m component n-bit Boolean functions f_{m-1} to f_0 with

$$S[x] = f_{m-1}(x)||\ldots f_1(x)||f_0(x).$$

These component Boolean functions f_i are called the *coordinate functions* of the S-box.

We now have a very natural definition for the *algebraic degree* of an S-box (or any (n, m)-function f), and we define the *algebraic degree* of the S-box to be the maximum algebraic degree of the coordinate functions of the S-box. Again, it seems that a high algebraic degree would be a very desirable property since it implies that at least one of the output bits depends on many input bits.

Related to the algebraic degree is another quantity called the *non-linear degree*. It is also called the *non-linear order* by some authors. Given an (n, m)-function f with $f\colon \mathcal{B}^n \to \mathcal{B}^m$, the non-linear degree of f, denoted by $\mathrm{NL}(f)$, is the *minimum* value of the algebraic degree of all linear combinations of the coordinate functions of f. This is a more demanding criterion than the algebraic degree itself and a high value to this quantity would again, at least intuitively, appear to be a desirable attribute. Loosely, it implies that not only are individual output bits from the S-box complicated, but so are some simple simple algebraic combinations.

The *input-output degree* of IO degree is of interest for resisting certain kinds of algebraic attacks. In many ways, it provides a natural extension of the ideas embodied in the algebraic degree.

Suppose we have an (n, m)-function f, then we can represent all the input bits as $x_{n-1} \dots x_0$ and all the output bits as $y_{m-1} \dots y_0$. We can then consider deriving a multivariate equation that holds over all inputs and has the following form:

$$\sum_{b \in \mathcal{B}^{n+m}} B_f(b) y_{m-1}^{b_{n+m-1}} y_{m-2}^{b_{n+m-2}} \cdots y_1^{b_{n+1}} y_0^{b_n}$$
$$x_{n-1}^{b_{n-1}} x_{n-2}^{b_{n-2}} \cdots x_1^{b_1} x_0^{b_0} = 0.$$

The sum is computed modulo 2 and B_f is a Boolean function on $(m + n)$-bit inputs that sets each coefficient for the 2^{n+m} terms in the summation. If we define d to be equal to the maximum Hamming weight of $b \in \mathcal{B}^{m+n}$ for which $B_f(b) = 1$, then d is termed the *degree* of the multivariate expression.

This is an exact analogy of the algebraic degree, but involves all the input and output bits for a non-Boolean function. However, in contrast to the case of the algebraic degree the sum considered here is not unique, and so we imagine searching over all multivariate expressions of the form above and keeping the lowest value of all *degrees*. This is called the *IO degree* of the function f.

It will not be a surprise that the quantities we have introduced are related. For an $(n.m)$-function f the IO degree is less than or equal to the non-linear degree, which in turn is less or equal to the algebraic degree. More concisely,

$$\mathrm{AD}(f) \geq \mathrm{NL}(f) \geq \mathrm{IO}(f).$$

4.4 Upper Bounds and Choosing S-Boxes

Since the goal is to choose S-boxes with highest possible value for the three degrees introduced, it's important to know their upper bounds.

In the special case of (n, n)-functions, for $n > 1$, the highest possible non-linear degree is $n - 1$ while the highest possible algebraic degree is n. Unlike in the algebraic and non-linear degrees, bounds on the IO degree depend closely on the size of the input and output. We omit the details, instead just give Table for the maximum values of IO degree for different values of n and m that appear in common S-box designs.

One school of thought suggests that we construct an S-box from a (simple) mathematical function with known properties. In this way we know exactly what we are getting. The downside is that the box will inevitably contain some structure, though if said structure cannot be exploited then it is hard to see what risks it might pose. The most frequently used mathematical functions in S-boxes have been the power functions over a Galois field. The S-box in the AES is constructed from an affine transformation (over $\mathrm{GF}(2)$) of the power mapping $f(x) = x^{-1} = x^{254}$ in $\mathrm{GF}(2^8)$. In this way it is guaranteed that the AES S-box has the highest possible algebraic and non-linear degree over $\mathrm{GF}(2^8)$ for a bijection, namely 7.

The opposing design approach is to choose an S-box at random, or rather to generate S-boxes repeatedly in a pseudorandom and verifiable way until we find an S-box that satisfies all the criteria that we seek. However, as we explained before, such approach won't work for S-boxes with more than five variables.

In addition to the security that is offered by an S-box, we are likely to be interested in the efficiency of its implementation, particularly in hardware. DES is such an example, its S-boxes are constructed from smaller four-bit S-boxes, which can lead to compact implementations.

5. Conclusion and Future Directions

In this paper, we have presented the fundamental concepts of differential cryptanalysis as applied to both simple toy ciphers and DES variants with reduced rounds. We also discussed the design criteria of DES S-boxes, revealed by original DES designers, together with the general theory of S-boxes as Boolean functions. Although a lot of things are still missing for people who is going to attack new ciphers, e.g., the method of finding good characteristics, but the author believe it's enough to enter the area of cryptanalysis by reading this work.

There're two possible future directions (for people who wants to learn more):

1. Go to linear cryptanlysis, by reading [10, Chapter 7] and [8, Section 3], another big topic which provided the current start-of-the-art attacking technique on DES.
2. Go deeper into mathematical theories on differential cryptanalysis and S-boxes, by reading [10, Chapter 8] and [6, Chapter 8 and 9].

There're always infinite things to learn. (but we have to finally stop at somewhere)

References

[1] F. L. Bauer. *Decrypted Secrets: Methods and Maxims of Cryptology*. Springer Science & Business Media, 4 edition, 2013.

[2] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[3] E. Biham and A. Shamir. Differential Cryptanalysis of the Full 16-round DES. In *Advances in Cryptology — CRYPTO' 92*, pages 487–496. Springer Berlin Heidelberg, Berlin, Heidelberg, Aug. 1992.

[4] E. Biham and A. Shamir. Differential Cryptanalysis of the Data Encryption Standard. pages 1–188, Dec. 2009.

[5] D. Coopersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, 38(3):243–250, May 1994.

[6] Y. Crama and P. L. Hammer. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Cambridge University Press, June 2010.

[7] Y. Crama and P. L. Hammer. *Boolean Functions*. Theory, Algorithms, and Applications. Cambridge University Press, May 2011.

[8] H. M. Heys. A Tutorial on Linear and Differential Cryptanalysis. *Cryptologia*, 26(3):189–221, July 2002.

[9] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, Nov. 2014.

[10] L. R. Knudsen and M. J. B. Robshaw. *The Block Cipher Companion*. Springer Publishing Company, Incorporated, Oct. 2011.