Linear and S-Box Pairs Cryptanalysis of the Data Encryption Standard

Fauzan Mirza

Third Year Undergraduate Project

October 1996 – April 1997

Royal Holloway
University of London

Department of Computer Science
Egham, Surrey TW20 0EX, England

Chapter 1

Introduction

Cryptography is mostly concerned with keeping messages secret. The idea is that a *plaintext* message (the message to be kept secret) is transformed to a *ciphertext* message (the coded message) by means of a secret *key*. Nobody then should be able to obtain the plaintext from the ciphertext without knowing the key. The method of transforming plaintext to ciphertext is called *encryption*, and the method itself is called the encryption *algorithm* (or *cipher system*).

Cryptography plays an important rôle in modern data security. One of the first advances in cryptography was due to the publication of the Data Encryption Standard (DES).

DES was developed by IBM in the early 1970's, and is currently the most widely-used data encryption algorithm. It is used by many commercial and private organisations. For example, the banking industry uses DES for encryption of transactions [16]. There is an obvious motivation for finding a weakness in DES; some method of being able to either yield the plaintext without the key, or better yet, being able to determine the key. However, for over 12 years since its standardisation in 1977, nobody had found a method of breaking DES.

Now, 20 years later, there are three published methods of finding the key in less time than naïve exhaustive key-search. Out of these three, one is a *chosen-plaintext* attack which means that the attacker must be able to, somehow, control the values of the plaintexts that are encrypted, as well as obtain their corresponding ciphertext. The other two are *known-plaintext* attacks which means that the attacker merely needs to know the values of the plaintexts and corresponding ciphertexts.

In this report, we provide descriptions of DES and the two known-plaintext attacks on DES. We will show how DES can be implemented, in the 'C' programming language, and optimised to run efficiently on 32-bit computers. We also implement two algorithms from each of the two attacks (a simple algorithm, and an advanced algorithm), and provide experimental proof of the effectiveness of these attacks.

1.1 Report Structure

In chapter 2, we give a detailed description of the DES cipher, explaining its operation and how each of its components affects its security. The chapter closes with a description and analysis of a weak DES-like system (demonstrating an important point concerning the security of any cipher system), and an algorithm for an exhaustive attack on DES.

In chapter 3, we describe the most recently developed attack on DES, known as "linear cryptanalysis". We explain the actual weakness and the principle of the attack, followed by an example of how the attack is developed. The chapter describes three algorithms, each algorithm offers an improvement in complexity.

In chapter 4, we describe the "S-box pairs cryptanalysis" method. We explain the intrinsic design flaw in DES, and how it provides the basis for the attack. Two algorithms (the original attack and an improved attack) are described in detail.

In chapter 5, we show how we designed and implemented the DES software (including descriptions of our optimisations to make the algorithm significantly faster), the linear cryptanalysis attacks, and the S-box pairs attacks. The source code for our software is given in appendix B.

In chapter 6, we present the results of our testing the software. We use our DES software to show a weakness in DES. The testing of the linear cryptanalysis and S-box pairs attacks are explained as a series of experiments in which we recover a part of the keys. A summary of the results concludes the chapter.

Finally, we give our assessment of this project in chapter 7, describing our approach to this project from planning to completion.

1.2 Terminology and Notation

When we say "known-plaintext", we are referring to the plaintext *and* its corresponding ciphertext.

There is a fair amount of mathematics (particularly in binary) discussed in this report, and so it is important that we introduce our terminology and notation here.

We define the term *MSB* to mean the "most significant bit" of a binary string. Similarly, we will use *LSB* to mean the "least significant bit" of a binary string.

When we say that an n-bit binary string is *MSB-first* it means that the bits of the string are labelled such that the MSB is labelled '1', through to the LSB being labelled 'n'. If an n-bit binary string is *LSB-first*, it means that the MSB is labelled '$n-1$', through to the LSB being labelled '0'. Note that the LSB-first definition is more conventional when talking about the positions of bits in software data types (integers, and so on). The MSB-first definition is used mainly when discussing the technical details of the Data Encryption Standard (because the official specification of DES [13] describes it assuming an MSB-first definition).

In the following explanation of the notation, we will use A and B to denote any n-bit binary string, and all other uppercase letters to denote arbitrary-length binary strings.

Logical shifts A logical right shift of T by u bits is denoted by: $T \gg u$. A logical left shift of T by u bits is denoted by: $T \ll u$. This is a primitive operation in the 'C' programming language.

Concatenation We denote concatenation of binary strings by enclosing the list of strings, in the required order, within parenthesis. For example, if $T = (U, V)$ then T is the string U followed by the string V. Also, $111001110 = (11, 1001, 110)$.

Complementation If $T = \overline{U}$ then T is the bitwise complement of U. The complement of U is the string obtained by inverting all its bits. This is a primitive operation in the 'C' programming language.

Exclusive-OR The logical exclusive-OR, or XOR, of strings, T and U, is denoted by $T \oplus U$. XOR has the property that $T \oplus T$ is the all-'0' binary string and $T \oplus \overline{T}$ is the all-'1' binary string, for all values of T. This is a primitive operation in the 'C' programming language. XOR is equivalent to bitwise addition modulo 2. XOR is a *linear* operation on binary strings.

Logical AND The logical AND operation of strings, T and U, is denoted by TU. AND causes the result to be zero in all places, except those places where both bits are '1'. This is a primitive operation in the 'C' programming language. AND is equivalent to bitwise multiplication modulo 2. AND is not a linear operation on binary strings.

Bit Selection We denote the i-th bit, in LSB-first order, of T as $T[i]$. For example, $A[0]$ is the LSB of A, and $A[n-1]$ is the MSB of A.

Selective Parity The notation $T[i, j, \ldots, k]$ is a shortcut for $T[i] \oplus T[j] \oplus \cdots \oplus T[k]$. This is the XOR of specified bits. For example, if $U = (10110)$, then $U[0, 1, 3] = 1$.

Dot Product The notation $A \cdot B$ is a shortcut for $A[n-1]B[n-1] \oplus A[n-2]B[n-2] \oplus \cdots \oplus A[0]B[0]$. Note that this is equivalent of calculating the parity of the bitwise AND of A and B.

Chapter 2

Data Encryption Standard

2.1 Description

DES is a block cipher that operates on 64-bit data blocks using a 56-bit key [1, 13]. It is a symmetric (or secret-key) cipher, which means that the key used for encryption will also be required for decryption. The DES algorithm is composed of two main functions:

- ◇ The key schedule, which prepares 16 48-bit subkeys. A subkey is a selection of bits from the 56-bit key.
- ◇ The cipher function, which scrambles the data block. The data block passes through 16 iterations (rounds) of the cipher function, and a different 48-bit subkey is applied in each round.

The 64-bit plaintext $\mathcal{P}$ is permuted according to a bijective mapping called the Initial Permutation (IP). The permuted 64-bit data block is split into two 32-bit halves. The left half is called L_0, and the right half is called R_0 Thus,

$$(L_0, R_0) = IP(\mathcal{P}).$$

The following transformations are executed 16 times, starting with $i = 1$.

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus F(R_{i-1}, K_i), \end{aligned} \tag{2.1}$$

where F is the cipher function, described in section 2.1.2. Equation 2.1 is called the *round function*, and is illustrated by figure 2.1. DES specifies 16 rounds, but we will be using a *reduced-round* DES, which uses n rounds, where $3 \leq n < 16$.

R_{16} and L_{16} are concatenated and permuted according to the Final Permutation The Final Permutation is the inverse of the Initial Permutation and is known as IP^{-1}. The output of IP^{-1} is the 64-bit ciphertext block,

$$\mathcal{C} = IP^{-1}((R_{16}, L_{16})).$$

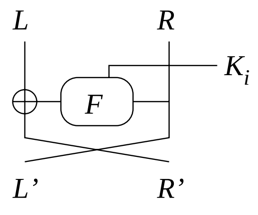

Figure 2.1 The round function

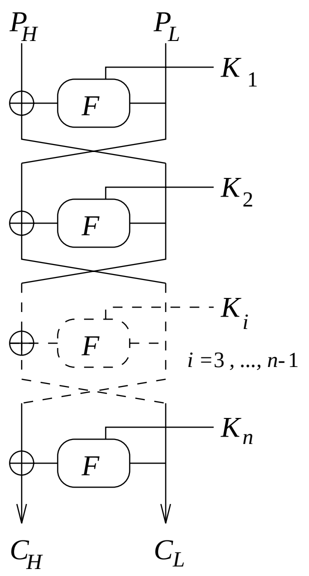

Figure 2.2 n-round DES encryption (the Feistel network)

Figure 2.2 shows an overview of n-round DES.

The algorithm to decrypt a data block is the same as for encryption, except that the subkeys are applied in the reverse order. This useful property is called the *symmetry* of DES, and is due to the iterative structure (this type of block cipher structure is called a *Feistel network*).

IP and IP^{-1} are at the start and end of of the ciphering transformation, respectively, and neither depends on the data nor the key. This has two consequences:

1. Permuting bits in software is a time-consuming process. There is a significant speed difference between software implementations of DES that use IP and IP^{-1} and those that do not use IP and IP^{-1} when used for mass data encryption. In hardware, these permutations are easy to implement and the speed difference is not as significant.

2. IP and IP^{-1} do nothing to deter a known-plaintext attack on the DES cipher, since the attacker could prepare for these transformations by applying IP to each plaintext and ciphertext block (assuming that the plaintext is input to the DES cipher) before using them for the attack.

For these reasons, we shall ignore the existence of these permutations in the remainder of this report. Thus we assume that

$$\mathcal{P} = (L_0, R_0),$$

and that

$$\begin{aligned} \mathcal{C} &= (L_n, R_n) \qquad \text{for odd } n, \\ \mathcal{C} &= (R_n, L_n) \qquad \text{for even } n. \end{aligned}$$

To simplify notation, we shall denote the left half of $\mathcal{P}$ by $\mathcal{P}_H$ and the right half by $\mathcal{P}_L$. Similarly, the left half of $\mathcal{C}$ will be denoted by $\mathcal{C}_H$, and the right half denoted by $\mathcal{C}_L$.

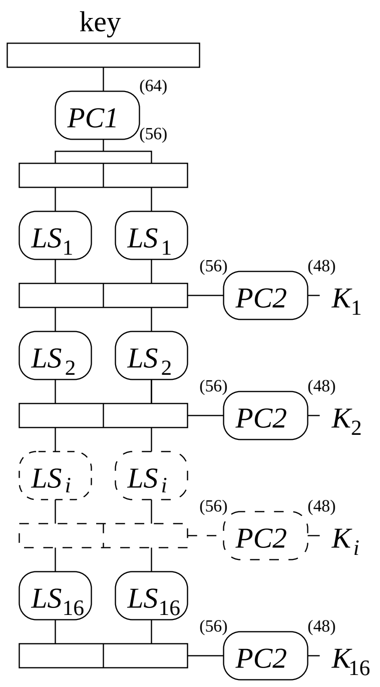

Figure 2.3 Key Schedule

2.1.1 Key Schedule

DES specifies a 64-bit key, but the cipher function only uses key bits selected from a 56-bit key. The additional 8 bits are used for parity checking.

We begin with a 64-bit key (with parity bits at every 8th bit, counting from the MSB), and reduce it to a 56-bit key through a permutation called Permuted Choice-1 ($PC1$). $PC1$ discards the parity check bits.

The 56-bit *effective key* $\mathcal{K}$ is split into two 28-bit halves, C_0 and D_0. Let $i = 1$, then repeat the following process until K_{16} has been calculated:

◇ Calculate

$$\begin{aligned} C_i &= LS(C_{i-1}, i), \\ D_i &= LS(D_{i-1}, i). \end{aligned}$$

The function LS returns the result of performing circular left shifts on C_{i-1}. The number of circular left shifts depends on the value of i, and is given by table A.2.

◇ Let $K_i = PC2((C_i, D_i))$, where $PC2$ is Permuted Choice-2, a 56-bit to 48-bit permutation.

Clearly, $PC1$ does not have any marked effect on the DES cipher, and since it is inconvenient, in further discussion we will ignore $PC1$ and assume a 56-bit key.

Each subkey K_i is 48 bits. In the next section, we shall see that in the context in which the subkeys are used, it is useful to split the subkey into 8×6 bits, and denote each 6-bit block by K_i^j where i is the subkey number, and j is the position of the 6-bit block. Thus,

$$K_i = (K_i^1, K_i^2, K_i^3, K_i^4, K_i^5, K_i^6, K_i^7, K_i^8).$$

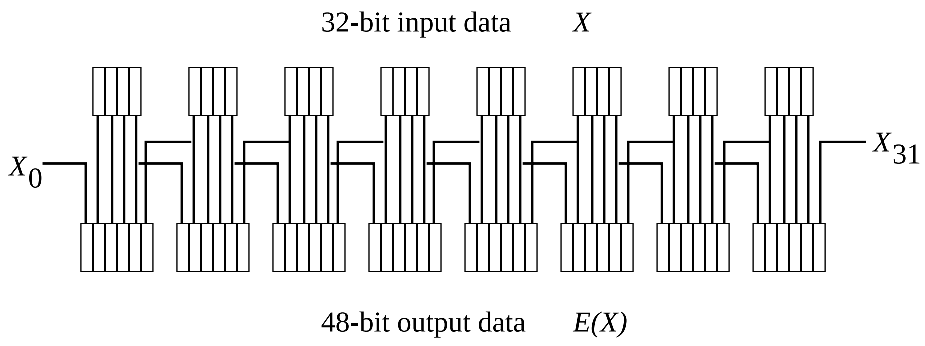

Figure 2.4 Expansion Permutation (E)

Table A.8 shows the 56-bit to 48-bit permutation for each subkey (LSB-first). Since the attacks on DES recover subkey bits, table A.8 is useful for determining which effective key bits are shared by the subkeys.

2.1.2 Cipher Function

The cipher function F produces a 32-bit output data block from a 32-bit input data block X and 48-bit subkey K:

$$F(X, K) \overset{\text{def}}{=} P(S(E(X) \oplus K)). \tag{2.2}$$

The 32-bit data is expanded according to an expansion table E, or E-box. The E-box describes a 32-bit to 48-bit permutation of X.

The data is split into eight 4-bit blocks. Each 4-bit block prepends its LSB to the beginning of the next 4-bit block, and appends its MSB to the end of its preceeding block. For example:

X	0101	1110	0111	0010	1010	1100	0111	0011
$E(X)$	**1**0101**1**	**1**1110**0**	**0**0111**0**	**1**0010**1**	**0**1010**1**	**0**1100**0**	**0**0111**0**	**1**0011**0**

The result of the expansion, a 48-bit block $E(X)$, is XORed to the 48-bit subkey K, producing the 48-bit input to the selection functions (more commonly known as the *substitution boxes*, or S-boxes).

There are eight S-boxes, each described by a 4×16 table. Each row of an S-box is a permutation of the integers from 0 to 15. Hence, each S-box represents a 6-bit to 4-bit mapping. The S-boxes are non-linear and provide most of the security of the DES cipher.

To perform the substitution, the 48-bit input is split into eight 6-bit blocks (one for each S-box). The first 6-bit block (bits 42–47) are input to S-box S_1, the second 6-bit block (bits 36–41) are input to S-box S_2, and so on. Note that, the key input to S-box S_i is K_r^i (where r is the round). The output of each S-box is determined by the following rules:

- ◇ The two outermost bits of the input (bits 0,5) are concatenated to determine the row of the S-box (an integer in the range 0 to 3).
- ◇ The middle four bits of the input (bits 1,2,3,4) determine the column of the S-box (an integer in the range 0 to 15).
- ◇ The 4-bit integer at the row and column of the S-box is the output (i.e., the substitution).

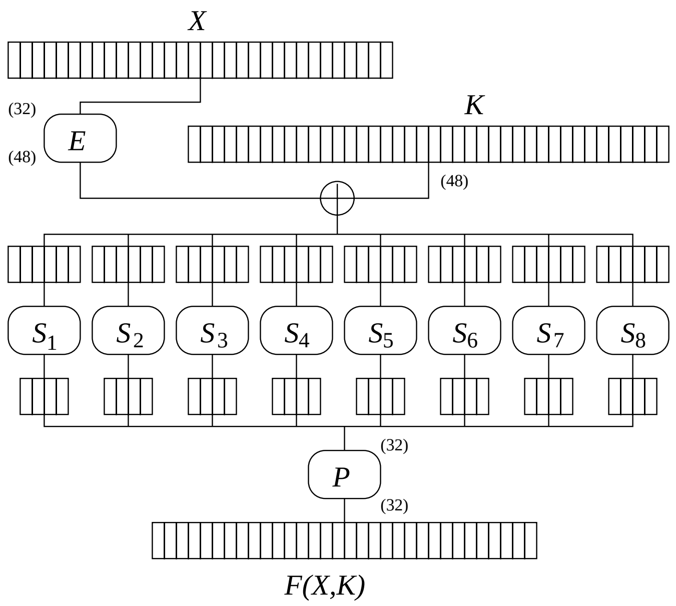

Figure 2.5 Cipher Function (F)

Since each S-box has a 4-bit output, the result is a 32-bit block.

Finally, the 32-bit output from the S-boxes is permuted according to the permutation table P, or P-box (a bijective mapping). The result after the permutation P is the output of the cipher function F.

2.2 Security Overview

Since its adoption, in 1977, as the US standard for data encryption, DES has been under intensive analysis. The main reason for this is that IBM were requested by the NSA (National Security Agency) to withhold from publishing the design criteria. Many people believed this to mean that NSA had convinced IBM to install a "trapdoor" into DES that would enable them to decrypt messages much faster than anticipated.

In fact, DES has shown to be very resistant to most methods of attack to which other, more recent, block ciphers have succumbed [16]. Much of the security of DES lies in the design of the cipher function.

The dependence of the ciphertext bits on the plaintext and key bits is called the *avalanche*. It has been shown that DES needs only 5 rounds to achieve complete avalanche [9] (i.e., after five rounds, every bit of the ciphertext depends on every bit of the plaintext and every bit of the key).

The purpose of the expansion permutation is clear: it expands the data by repeating bits such that the repeated bits will enter into neighbouring S-boxes. This increases the input/output dependence.

2.2.1 Insecurity of Linear S-Boxes

The most interesting part of DES are its S-boxes. All the operations in DES, except for the S-boxes, are linear. If the S-boxes were also linear then we would be able to express the operation of DES as a linear equation of the plaintext, ciphertext and key. We can demonstrate this by simulating the execution of 3-round DES:

$$\begin{aligned} L_1 &= R_0, \\ R_1 &= L_0 \oplus F(R_0, K_1), \\ L_2 &= R_1, \\ R_2 &= L_1 \oplus F(R_1, K_2), \\ L_3 &= R_2, \\ R_3 &= L_2 \oplus F(R_2, K_3). \end{aligned} \tag{2.3}$$

Expanding equation 2.3, we get

$$\begin{aligned} L_3 &= R_0 \oplus F(L_0 \oplus F(R_0, K_1), K_2), \\ R_3 &= L_0 \oplus F(R_0, K_1) \oplus F(R_0 \oplus F(L_0 \oplus F(R_0, K_1), K_2), K_3). \end{aligned} \tag{2.4}$$

We are unable to simplify the equations because the linear (permutation) and non-linear (substitution) operations in the cipher function do not commute. The DES equation becomes increasingly complex as more rounds are added. Compare equation 2.4 with the equation obtained from 3-round DES with linear S-boxes:

$$\begin{aligned} L_3 &= R_0 \oplus P(S(E(L_0))) \oplus P(S(E(R_0))) \oplus P(S(K_1)) \oplus P(S(K_2)), \\ R_3 &= L_0 \oplus P(S(E(L_0))) \oplus P(S(E(R_0))) \oplus P(S(K_2)) \oplus P(S(K_3)). \end{aligned}$$

The final linear equation always remains simple since some terms introduced from previous rounds cancel out (i.e., additional rounds would not increase security if the round function is linear).

We can calculate the S-boxes that give the result of the cumulative substitutions of all n rounds (because the S-boxes are linear). We let the inverse S-boxes S^{-1} be the inverse of these "n-round S-boxes". Using just one plaintext/ciphertext pair we obtain:

$$K_1 \oplus K_3 = S^{-1}(P^{-1}(L_0 \oplus R_0 \oplus L_3 \oplus R_3)). \tag{2.5}$$

As the S-boxes are 6-bit to 4-bit mappings, we cannot solve them uniquely. The following attack exploits the shared data bits between S-boxes to reduce the complexity of exhaustive key-search. The attack finds 256 possibilities for 48 bits of the key. For each of these possibilities, we try possibilities for the remaining 8 bits, by exhaustive search. We assume that the n-round S-boxes will have the same type of tabular structure as the DES S-boxes (namely that each row is a permutation of the integers in $\mathbf{Z}_2^4$) and also that they can be expressed as a linear (or affine) function, such that the coefficient of the input is a matrix with no linearly dependent rows or columns. The latter condition ensures that any four columns of the S-box matrix form an invertible (*non-singular*) matrix. If this condition does not hold, we may still be able to perform this type of attack, but the details become more complicated and we will not discuss them here.

A similar attack, applicable to 2-round DES is given in [8]; it cannot be extended to more than 2-round DES because the complexity of solving the non-linear S-boxes becomes too large (the trick used in the following discussion to "chain S-box inputs" cannot be applied to non-linear S-boxes).

We use equation 2.5 as the basis for the attack on "linear DES". Inverting linear S-box S_1 would give 4 linear equations (each equation corresponds to an output value in a row of the S-box, as each row is a permutation). Since we require 6 bits of S-box input, we fix the two LSBs of the S-box input and evaluate the other four using the S_1^{-1} linear expressions[1]. The two bits which we fixed would be shared by

[1] We obtain a unique solution for the other four bits *iff* the linear expressions required for the S-box inversion can be expressed as a non-singular matrix. As we have fixed two bits, the linear equations will correspond to a 4×4 matrix; if we can transform this matrix to the identity matrix (using linear algebra), then it is non-singular.

linear S-box S_2 because of the E-box, so we can automatically evaluate all six input bits to S_2 from the linear equations obtained by S_2^{-1}. Two bits are shared between the inputs of S_2 and S_3, and so we can evaluate the input to S_3, and so on. Thus, to invert all the S-boxes we only need to try possibilities for 2 input bits to S_1, so we would have 4 linear expressions (1 correct, 3 incorrect), each representing 48 key bits.

By guessing the value of those key bits which appear in either K_1 or K_n, but not both, we can recover 48 bits of the "key". In case of 3-round linear DES, there are only 6 key bits not shared by K_1 and K_3 (determined from table A.8).

The remaining 8 key bits can be recovered by reduced exhaustive key search. Thus the complexity of an attack on DES if it had linear S-boxes (satisfying the conditions described above) would be $O(2^{16})$. Although we have put restrictions on the structure of the S-boxes to make this analysis easier, the point is that linear ciphers, even DES-like linear ciphers, are solvable by known-plaintext attacks.

The S-boxes possess some peculiar structural properties [8], which only served to reinforce the belief that IBM had chosen the S-boxes to satisfy a particular (secret) purpose.

The permutation P distributes the data bits so that they will not enter the same S-boxes in the following round [4]. This speeds up the avalanche effect.

2.2.2 Design Criteria

The S-boxes were indeed designed to satisfy a particular purpose. At the time of its development, the IBM researchers had discovered a method of finding weaknesses in S-boxes that could allow a cipher to be broken in a chosen-plaintext attack. The IBM team called it the "T attack", but the researchers who rediscovered it[2] in 1990, called it *differential cryptanalysis* [3].

In 1994, IBM published the design criteria for the S-boxes. Out of the eight criterion listed, only one addresses linearity:

> "No output bit of an S-box should be too close to a linear function of the input bits." [4]

Another S-box design criterion specifies the size of the input and output, and the other six serve to make differential cryptanalysis more difficult.

2.2.3 Complementation Property

The data and subkey are combined using XOR before being input to the S-boxes (equation 2.1). Neither the expansion E nor the key-schedule (which generates K) apply any substitution; they only permute bits. Consequently $E(\overline{X}) \oplus \overline{K} \equiv E(X) \oplus K$ and hence $F(\overline{X}, \overline{K}) \equiv F(X, K)$.

One round of DES is defined as

$$\begin{aligned} L_i &= R_{i-1}, \\ R_i &= L_{i-1} \oplus F(R_{i-1}, K_i). \end{aligned}$$

If the inputs are complemented, so are the outputs:

$$\begin{aligned} \overline{L_i} &= \overline{R}_{i-1}, \\ \overline{R_i} &= \overline{L}_{i-1} \oplus F(\overline{R}_{i-1}, \overline{K}_i). \end{aligned}$$

[2] Actually, the technique of differential cryptanalysis was first published as an attack on another block cipher called FEAL [12].

This holds for all rounds and so, by induction:

$$\text{If} \quad \mathcal{C} = \text{DES}(\mathcal{P}, \mathcal{K}) \quad \text{then} \quad \overline{\mathcal{C}} = \text{DES}(\overline{\mathcal{P}}, \overline{\mathcal{K}}).$$

The following table shows the results of using DES to encrypt complemented plaintexts with complemented keys. The two ciphertexts, $\mathcal{C}$ and $\mathcal{C}'$, are distinct.

DES	$\mathcal{K}$	$\overline{\mathcal{K}}$
$\mathcal{P}$	$\mathcal{C}$	$\overline{\mathcal{C}'}$
$\overline{\mathcal{P}}$	$\mathcal{C}'$	$\overline{\mathcal{C}}$

In a chosen-plaintext attack, where the attacker can choose some plaintexts to be encrypted under the unknown key, the *complementation property* can half the computational complexity of an exhaustive key search. The results in the above table are the basis of the following algorithm:

Algorithm 2.2.1 Exhaustive Key Search (Complementation Property)

1. Obtain the ciphertext $\mathcal{C}$ corresponding to a randomly chosen (or known) plaintext $\mathcal{P}$.
2. Obtain the ciphertext $\mathcal{C}'$ corresponding to $\overline{\mathcal{P}}$, the complement of the previously chosen plaintext $\mathcal{P}$.
3. Perform a reduced exhaustive key search:
   For each key $\mathcal{K} \in \mathbf{Z}_2^{55}$ (all possible 55-bit integers), do the following:
   - ◇ Let $\mathcal{C}_{\mathcal{K}} = \text{DES}(\mathcal{P}, \mathcal{K})$. Note that the MSB of the 56-bit key is zero.
   - ◇ If $\mathcal{C}_{\mathcal{K}} = \mathcal{C}$ then the key is $\mathcal{K}$. Stop.
   - ◇ If $\mathcal{C}_{\mathcal{K}} = \overline{\mathcal{C}'}$ then the key is $\overline{\mathcal{K}}$. Stop.

Although this is a chosen-plaintext attack (since we require one plaintext to be the complement of another), there is a greater than even chance that among any 2^{32} random known-plaintexts, we will find a pair, $\mathcal{P}$ and $\mathcal{P}'$, such that $\mathcal{P}' = \overline{\mathcal{P}}$ [8].

If we know the value of some bits of the key $\mathcal{K}$ then we cannot exploit the complementation property (even if we only know the value of only one key bit), since $\overline{\mathcal{K}}$ will also have the known key bits complemented — and we know that this cannot be the key.

However, if we know a linear expression consisting of two or more key bits (as opposed to the value of the key bits), then we can use the complementation property. We implement the linear expression in the reduced exhaustive key search algorithm to fix the value of one key bit based on the value of other key bits.

Chapter 3

Linear Cryptanalysis

The security of DES essentially lies in the S-boxes. The first published analysis of DES presented a number of ideas on design and analysis of the DES S-boxes [8]. In particular, Hellman *et al.* demonstrated the threat of linear S-boxes, including "hidden" linearity (linearity in subsets of the inputs and outputs of an S-box), and also pointed out that even S-boxes that were close to linear (in a probabilistic sense) could be a problem.

Whilst the S-boxes are non-linear, they exhibit properties that are very close to linear. Shamir noted that, for each S-box, there is a high correlation between input bit 4 and the XOR of the four output bits (i.e., a linear relation between S-box input and output bits that holds with high probability) [17].

The observations by Hellman *et al.* and Shamir provide the basis for a known-plaintext attack, developed by Matsui, known as *linear cryptanalysis* [10]. Linear cryptanalysis can break 16-round DES with 2^{43} plaintexts and complexity $O(2^{43})$ [11]. It is currently the most effective attack against DES.

3.1 Description

The principle behind linear cryptanalysis is we find linear approximations to the non-linear S-boxes, then use these linear expressions to construct a *linear path* through the Feistel network, so that the result will be a linear expression for the DES cipher that relates the plaintext, ciphertext and key.

There are two stages to linear cryptanalysis: (1) finding a suitable linear approximation for the DES cipher, and (2) applying the known-plaintext attack algorithm. First we give an outline of how we find linear approximations to the DES cipher:

1. Find linear approximations to the S-boxes that relates a subset of the S-box inputs to a subset of its outputs. We find the probability of each approximation holding true.

2. Extend the approximations to the cipher function F, and thus formulate linear expression for each approximation.

3. Construct a linear path (an approximation of the DES cipher) by compounding linear expressions for the cipher function F. We can calculate the probability for the DES approximation from the probabilities of the S-box approximations.

4. Calculate the number of plaintexts required to obtain the value of the DES linear approximation expression. The number of plaintexts depends on the probability that the DES approximation holds, as well as our required degree of success.

We will describe these steps in greater detail.

3.1.1 Linear Approximation of S-Boxes

One of the first demonstrations of linearity in the DES S-boxes was provided in [15]. Rueppel shows an application of the Walsh transform in finding the best linear approximation to non-linear boolean functions, and provides an example of the best linear approximation to DES S-boxes S_2 and S_5 (incidentally, the best linear approximations found by Rueppel are the same linear correlation noted by Shamir).

Let F be the non-linear function to be approximated, n and m are the size (in bits) of the input and output to F respectively, α is a n-bit vector over which all possible inputs will be projected, and β is an m-bit vector over which the outputs will be projected. Matsui used a measure of linearity very similar to the Walsh transform:

$$N_F(\alpha, \beta) = \sum_{x=0}^{2^n-1} (-1)^{x \cdot \alpha} F(x) \cdot \beta. \tag{3.1}$$

For comparison, the Walsh transform is computed by:

$$S_F(\alpha) = \sum_{x=0}^{2^n-1} (-1)^{x \cdot \alpha} F(x). \tag{3.2}$$

The further away the result of equation 3.1 is from zero, the greater the linearity. We will now put equation 3.1 into the context of measuring the linearity of DES S-boxes:

$$N_{S_n}(\alpha, \beta) = \sum_{x=0}^{63} (-1)^{x \cdot \alpha} S_n(x) \cdot \beta. \tag{3.3}$$

Provided that $\alpha \neq 0$, the result is an even integer in the range -32 to 32 (the result will be even because the S-boxes are *0-1 balanced* — there are an equal number of '0' bits as '1' bits). The result will be less than zero if the correlation holds with high probability (greater than 1/2), zero if there is no correlation, and greater than zero if the correlation holds with low probability (less than 1/2).

We define function $NS_n(\alpha, \beta) = 32 - N_{S_n}$, which counts the number of times that the result of the two projections coincide (an integer in the range 0 to 64). Matsui gives the following, equivalent, definition of this function [10]:

$$NS_n(\alpha, \beta) \stackrel{\text{def}}{=} \#\{x | 0 \leq x < 64, (\bigoplus_{s=0}^{5}(x[s] \cdot \alpha[s])) = (\bigoplus_{t=0}^{3}(S_n(x)[t] \cdot \beta[t]))\}. \tag{3.4}$$

The probability that the linear approximation expression associated with the pair (α, β) holds is $NS_n(\alpha, \beta)/64$.

For example, $NS_6(16, 7) = 18$, therefore the probability that input bit 4 (value 16) of S_6 coincides with output bits 0, 1, and 2 (value 7) is $18/64 = 0.28125$.

Testing all S-boxes with all values of α and β reveals that the best approximation (i.e., the one where $|N_{S_n}(\alpha, \beta)|$ is maximal) comes from $N_{S_5}(16, 15) = -20$. This has a coincidence probability of $12/64 = 0.1875$.

We can now derive a linear expression representing one round of DES (i.e., a linear expression for the cipher function F). To find the linear expression we note that the input bit(s) from the subkey must be in the same position as the input bit(s) of the data block after the E-box expansion, and that the output bit(s) are permuted through the P-box.

For $NS_6(16,7)$, we obtain the one-round linear approximation:

$$X[11] \oplus F(X,K)[3,21,13] = K[16], \tag{3.5}$$

where X is the input data block, K is the round subkey, and $F(X,K)$ is the output data block. The one-round expression holds with the same probability as the linear approximation from which it was derived, e.g., equation 3.5 holds with probability 0.28125 for random X and fixed K.

3.1.2 n-round Linear Approximations

Now we extend the approximation to a number of rounds. We do this by compounding linear expressions. For example, to extend equation 3.5 which is a one-round linear approximation to 3-round DES, we start by expressing it for the first round.

If $\mathcal{P}_H$ is the left half of the plaintext block and $\mathcal{P}_L$ is the right half of the plaintext (that enters the cipher function F), then we have:

$$F(\mathcal{P}_L, K_1)[3,21,13] \equiv \mathcal{P}_H[3,21,13] \oplus X_2[3,21,13], \tag{3.6}$$

where X_2 is the data block as it enters F in the second round. Substituting equation 3.6 in equation 3.5, we obtain the linear approximation for the first round:

$$\mathcal{P}_L[11] \oplus \mathcal{P}_H[3,21,13] \oplus X_2[3,21,13] = K_1[16]. \tag{3.7}$$

Similarly, we apply the linear approximation (equation 3.5) to the final round:

$$\mathcal{C}_L[11] \oplus \mathcal{C}_H[3,21,13] \oplus X_2[3,21,13] = K_3[16], \tag{3.8}$$

where $\mathcal{C}_L$ and $\mathcal{C}_H$ are the right and left halves of the ciphertext respectively. Adding equations 3.7 and 3.8 cancels the (unknown) X_2 term giving a linear approximation to 3-round DES:

$$\mathcal{P}_L[11] \oplus \mathcal{P}_H[3,21,13] \oplus \mathcal{C}_L[11] \oplus \mathcal{C}_H[3,21,13] = K_1[16] \oplus K_3[16]. \tag{3.9}$$

The probability that equation 3.9 holds for random plaintext $\mathcal{P}$ and its corresponding ciphertext $\mathcal{C}$ is $(18/64)^2 + (1-18/64)^2 = 0.5957$. This is because it will hold *if and only if* either both one-round approximations hold, or if neither hold. The probability can also be calculated using equation 3.10, which evaluates the probability that the binary sum of n independent random variables X_i $(1 \le i \le n)$ will equal zero [10]:

$$\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2}\right). \tag{3.10}$$

Matsui estimates the success rate of the algorithm and the number of plaintexts required to acheive a particular degree of success. The success rate is based on the maximum likelihood method[1] and is derived by approximating binary distribution with normal distribution. Table 3.1, given in [10], lists the relationship between the number of plaintexts N and the success rate.

We need $|p-1/2|^{-2}$ plaintexts to acheive a 97.7% success rate, i.e., $N = |0.5957-1/2|^{-2} \approx 110$ plaintexts. We use algorithm 3.1.1 to recover the value of the linear approximation (equation 3.9) and, hence, obtain a linear expression for a key bit.

[1]An explanation of the principles of a maximum likelihood method will be given in chapter 5.

N	$\frac{1}{4}\lvert p-1/2\rvert^{-2}$	$\frac{1}{2}\lvert p-1/2\rvert^{-2}$	$\lvert p-1/2\rvert^{-2}$	$2\lvert p-1/2\rvert^{-2}$
Success Rate	84.1%	92.1%	97.7%	99.8%

Table 3.1 Success rate of linear cryptanalysis (algorithm 3.1.1)

Algorithm 3.1.1 Linear Cryptanalysis (1 key bit)

1. Initialise counter $T = 0$.
2. For each plaintext $\mathcal{P}_i$ and corresponding ciphertext $\mathcal{C}_i$, where $0 \le i < N$, do the following:
   - ◇ Let t be the evaluation of the left-hand side of the DES linear approximation, e.g., evaluate LHS of equation 3.9:
   $$t \leftarrow \mathcal{P}_L[11] \oplus \mathcal{P}_H[3,21,13] \oplus \mathcal{C}_L[11] \oplus \mathcal{C}_H[3,21,13].$$
   - ◇ If $t = 0$, increment counter T.
3. If the probability of the DES linear approximation is greater than 1/2 (e.g., $p = 0.5957$):
   - ◇ If $T > N/2$ then guess that the value of the right-hand side of the DES linear approximation is 0, e.g., the RHS of equation 3.9: $K_1[16] \oplus K_3[16] = 0$.
   - ◇ If $T \le N/2$ then guess that the value of the right-hand side of the DES linear approximation is 1, e.g., the RHS of equation 3.9: $K_1[16] \oplus K_3[16] = 1$.
4. If the probability of the DES linear approximation is less than than 1/2:
   - ◇ If $T > N/2$ then guess that the value of the right-hand side of the DES linear approximation is 1.
   - ◇ If $T \le N/2$ then guess that the value of the right-hand side of the DES linear approximation is 0.

The success rate of this algorithm increases when N or $|p - 1/2|$ increases. Note that in this example, the probability is not maximal.

The value of the linear equation for the key bit would be used in a reduced exhaustive key search algorithm. Having recovered a linear expression for one key bit, we would have only 55 key bits to search. If we can use the complementation property then we have to search only 54 key bits (see section 2.2.3).

For linear expressions for a larger number of rounds (except for 5- and 11-round DES), in order to obtain the best approximation, we would build the DES linear approximation using different S-box approximations (in our 3-round DES example, we used the same approximation for the first and final rounds, thus making the construction symmetrical). Matsui gives a table of the best linear approximation expressions for DES upto 16 rounds ("best" means optimal probability) which was found by computer search [10].

We can use the symmetry of DES to obtain another linear expression for a key bit, with little additional overhead. In those instances where the n-round DES linear approximation construction is not symmetrical (i.e., we do not use an S-box approximation in round $(n - i + 1)$ if it is used in the ith round) we can run algorithm 3.1.1 again, swapping the rôles of the plaintext and ciphertext. The result would be the value of the "reverse" key linear equation, which is similar to the original expression, except that all subkeys K_i are replaced with K_{n-i+1}. Note that in an implementation, we would not actually run the algorithm twice: it would be more efficient to use two counters T and U, one for each expression, in the data analysis phase of the algorithm.

3.1.3 (n–1)-round Linear Approximations

An improvement over the basic algorithm allows us to recover more key bits at a time. We do this by using a linear expression that holds for n-rounds with the probability of an (n–1)-round approximation. The only way to construct such an expression is to apply the DES linear approximation expression to the first n rounds only, and leave the final round unapproximated. The final (non-linear) round will be part of the resulting expression, but we will show how to deal with this with an example for 8-round DES.

The best expression for 8-round DES, using the best 7-round linear approximation (holding with probability $0.5 + 1.95 \times 2^{-10}$), is [10]:

$$\begin{aligned}\mathcal{P}_H[7,18,24] \oplus \mathcal{P}_L[12,16] \oplus \mathcal{C}_H[15] \oplus \mathcal{C}_L[7,18,24,29] \oplus F(\mathcal{C}_L, K_8)[15] \\ = K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22].\end{aligned} \tag{3.11}$$

Note that we are only interested in one bit of the term involving the cipher function F (bit 15). The value of this bit will be determined by one S-box, and hence will be affected by only six subkey bits. So although equation 3.11 contains the 48-bit subkey K_8, we need only try possibilities for the six subkey bits that influence $F(\mathcal{C}_L, K_8)[15]$, namely $K_8^1 = (K_8[47], \ldots, K_8[42])$.

Thus we require counters U_i for each possible value for the 6 subkey bits, and proceed with the linear cryptanalysis algorithm. We expect the maximum likelihood method to reveal which of the $2^6 = 64$ possible results is the correct one. Not only will we obtain the linear expression for a key bit (the right-hand side of the DES linear approximation), but we also find out the value of the 6 subkey bits. Thus we should recover a total of 7 key bits (including a linear expression for a key bit).

Also note that, in equation 3.11, we need only 7 bits of information from the plaintext/ciphertext. These 7 *text bits* include a one bit result of an XOR of plaintext and ciphertext bits, and six bits input to the S-boxes. Since we shall need to evaluate part of the cipher function F in order to calculate the left-hand side of equation 3.11, it is efficient to separate the data collection and data analysis into two distinct phases, and dedicate the first phase (*data collection phase*) to collecting the text bits from each plaintext/ciphertext pair. The collected data can then be processed in the second phase (also known as the *key counting phase*). By "collecting" the text bits, it suffices to increment a counter, U_x, for each plaintext/ciphertext (where x is a 7-bit string representing the 7 text bits extracted from the plaintext/ciphertext pair). For example, we may store the value of

$$\mathcal{P}_H[7,18,24] \oplus \mathcal{P}_L[12,16] \oplus \mathcal{C}_H[15] \oplus \mathcal{C}_L[7,18,24,29],$$

into the LSB of x, and the six bits

$$\mathcal{C}_L[0], \mathcal{C}_L[31], \ldots, \mathcal{C}_L[27]$$

into the 6 MSBs of x, then increment U_x.

Algorithm 3.1.2 Linear Cryptanalysis (7 key bits)

1. **Data Counting Phase:**
   Initialise T_i, where $0 \leq i < 128$, to zero.

2. For each plaintext $\mathcal{P}_i$ and corresponding ciphertext $\mathcal{C}_i$, where $0 \leq i < N$, do the following:

   ◇ Let t hold the text bits used in the left-hand side of the DES approximation equation. For example, from equation 3.11:

   $$t \leftarrow (\mathcal{C}_L[0], \mathcal{C}_L[31], \mathcal{C}_L[30], \mathcal{C}_L[29], \mathcal{C}_L[28], \mathcal{C}_L[27], \\ \mathcal{P}_H[7,18,24] \oplus \mathcal{P}_L[12,16] \oplus \mathcal{C}_H[15] \oplus \mathcal{C}_L[7,18,24,29])$$

   ◇ Increment counter T_t.

3. **Key Counting Phase:**
   For each $s \in \mathbf{Z}_2^6$, do the following:

   ◇ Initialise U_s to zero.

   ◇ For each $t \in \mathbf{Z}_2^7$, let $U_s = U_s + T_t$ if the left-hand side of the DES linear approximation, with $F(\text{relevant\_6\_bits\_of\_}t, s)$ for the final round, is zero. For example, from equation 3.11: if

   $$t[0] \oplus F((t[6], t[5], t[4], t[3], t[2], t[1]), (s[5], s[4], s[3], s[2], s[1], s[0]))[15] = 0,$$

   then let $U_s = U_s + T_t$.
   Note: $F(X, K)[15] \equiv S_1(X \oplus K)[2]$.

4. Let U_{max} be the maximum U_i counter, and let U_{min} be the minimum U_i counter.

5. If the probability of the DES linear approximation is greater than 1/2 (e.g., $p = 0.5 + 1.95 \times 2^{-10}$):

   ◇ If $|U_{max} - N/2| > |U_{min} - N/2|$ then guess that the value of the right-hand side of the DES linear approximation is 0 and the 6 key bits affecting $F(X, K)$ in the last round are given by max, e.g., the RHS of equation 3.11): $K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] = 0$, and max represents the six bits entering S_1 in the last round.

   ◇ If $|U_{max} - N/2| \leq |U_{min} - N/2|$ then guess that the value of the right-hand side of the DES linear approximation is 1 and the 6 key bits affecting $F(X, K)$ in the last round are given by min, e.g., the RHS of equation 3.11): $K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] = 1$, and min represents the six bits entering S_1 in the last round.

6. If the probability of the DES linear approximation is less than than 1/2:

   ◇ If $|U_{max} - N/2| > |U_{min} - N/2|$ then guess that the value of the right-hand side of the DES linear approximation is 1 and the 6 key bits affecting $F(X, K)$ in the last round are given by max.

   ◇ If $|U_{max} - N/2| \leq |U_{min} - N/2|$ then guess that the value of the right-hand side of the DES linear approximation is 0 and the 6 key bits affecting $F(X, K)$ in the last round are given by min.

The complexity of the data counting phase is $O(N)$ and the complexity of the key counting phase is $O(2^{13})$. For a 97% success rate, we need around 2^{20} known-plaintexts.

Using algorithm 3.1.2 we recover 7 key bits (including one linear expression for a key bit), but we can recover another 7 key bits because of the symmetry of DES: Apply the 7-round approximation to the second to final rounds, and include

$F(\mathcal{P}_L, K_1)[15]$ to recover $K_1^1 = (K_1[47], \ldots, K_1[42])$. So by running the algorithm twice we recover 14 key bits. The remaining $(56-14) = 42$ key bits can be recovered by reduced exhaustive search. Because we recover the value of 12 key bits, we would no longer be able to exploit the complementation property in the reduced exhaustive key search.

3.1.4 (n–2)-round Linear Approximations

In the year following the publication of his linear cryptanalysis attack on DES, Matsui improved it further by suggesting that the linear approximation cover the (n–2) 'middle' rounds of the cipher, and to use counters for the outer two rounds [11].

This attack will recover one linear expression for a key bit, six bits of the subkey applied in the first round, and six bits of the subkey applied in the final round: a total of 13 key bits. Once again, the algorithm can be applied in "reverse" (provided that the linear approximation is not symmetrical) to recover another 13 bits, so that we have a total of 24 key bits plus linear expressions for 2 key bits. The remaining $(56 - 26) = 30$ bits can be found using reduced exhaustive key search.

The n-round DES linear approximation is based on the best approximation of (n–2)-round DES. Thus, we will need much less known-plaintext to achieve a high success rate than the n-round and (n–1)-round algorithms. The best 6-round linear approximation for 8-round DES is

$$\begin{aligned} &\mathcal{P}_H[7,18,24] \oplus \mathcal{C}_H[15] \oplus \mathcal{C}_L[7,18,24,29] \\ &\oplus F(\mathcal{P}_L, K_1)[7,18,24] \oplus F(\mathcal{C}_L, K_8)[15] \\ &= K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22]. \end{aligned} \tag{3.12}$$

Equation 3.12 holds with probability $1/2 - 1.95 \times 2^{-9}$ [11].

The (n–2)-round attack is very similar to algorithm 3.1.2; there are only a few modifications. The main difference is that we will require an additional 2^6 text counters and key counters (because we are counting for another unapproximated S-box). Thus we have a total of $2^{13} = 8192$ text counters T_i (where i holds the one bit XOR of some plaintext and ciphertext bits, 6-bit plaintext input to the S-box in the first round, and 6-bit ciphertext input to the S-box in the final round) and $2^{12} = 4096$ key counters U_j (where j holds the 6 key bits input to the S-box in the first round, and also the 6 key bits input to the S-box in the final round). The S-boxes used in the first and final round need not be the same.

Chapter 4

S-Box Pairs Cryptanalysis

Recall that the expansion permutation E causes certain data bits to be repeated (hence expansion). The principle of repeating bits to increase avalanche is sound but when coupled with the linear fashion in which the key is added, it allows a statistical relationship between the plaintext, ciphertext and key to be observed.

This property was first reported by Davies [7]. A few years later, the "potential weakness" was expanded to a known-plaintext attack by Davies and Murphy [6].

The basic attack hinges on the fact that the distribution of the outputs from a pair of S-boxes that share two data bits is non-uniform (the shared data bits are due to the E-box). This is the *S-box pairs cryptanalysis*. An extension of this is an *S-box triplets cryptanalysis*, which examines the non-uniform distribution from three adjacent S-boxes (sharing four data bits).

For DES, the triplets analysis has no major advantages over a pairs analysis [6]. Actually, DES appears to be optimised against the S-box pairs cryptanalysis too, but the attack is a feasible threat for reduced-round DES.

4.1 Description

There are two stages to the S-box pairs cryptanalysis: (1) finding the S-box pairs output distribution table for as many rounds as necessary, and (2) applying the known-plaintext attack algorithm. We first show how to obtain the S-box pairs distribution for one round of DES (i.e., just the cipher function), then show how to compute the S-box pair output distribution tables for more rounds.

4.1.1 S-Box Pair Output Distribution Table

For this example, we shall focus on the S-box pair output distribution table for S-boxes S_1 and S_2.

Let I and J be the 6-bit inputs to S_1 and S_2 respectively. The inputs to the cipher function are the 32-bit data, X, and a 48-bit subkey K. The E-box causes pairs of bits to be repeated such that they enter into neighbouring S-boxes. From figure 4.1, we can see that

$$\begin{aligned} I[1] &= X[28] \oplus K[43], \\ I[0] &= X[27] \oplus K[42], \\ J[5] &= X[28] \oplus K[41], \\ J[4] &= X[27] \oplus K[40]. \end{aligned}$$

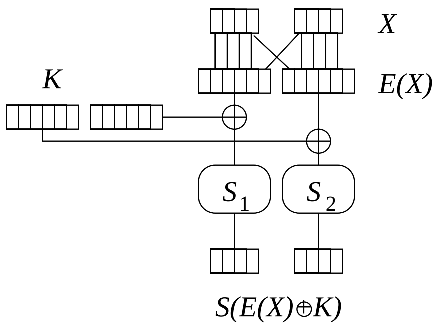

Figure 4.1 Data bits shared between a pair of S-boxes

Hence the XOR of four consecutive bits that enter a pair of adjacent S-boxes gives us a linear equation whose terms are key bits:

$$k = K[40, 41, 42, 43].$$

Knowledge of k would enable us to calculate the value of one key bit in a reduced exhaustive key search. The outputs from the S-box pair leak information about k, and we exploit this in the known-plaintext attack (to obtain the value of k).

We need linear equations obtained from each of the two shared data bits. Let $s = I[0] \oplus J[4]$ and $t = I[1] \oplus J[5]$, so

$$\begin{aligned} s &= K[40, 42], \\ t &= K[41, 43]. \end{aligned} \tag{4.1}$$

Note that $k = s \oplus t$. The value of the pair (s, t) is independent of the data bits. Fixing the value of (s, t), we count occurences of each 8-bit output of the S-box pair S_1 and S_2 over all possible 12-bit inputs. The function D_1 describes the number of times the output (X, Y) of a pair of neighbouring S-boxes, S_p and S_q, occurs over all possible inputs (I, J) for given values of s and t:

$$D_1(X, Y, s, t) \stackrel{\text{def}}{=} \#\{I, J \in \mathbf{Z}_2^6 | I[1] \oplus J[5] = s, I[0] \oplus J[4] = t, S_p = X, S_q = Y\}.$$

Since we fix the value of two key bits, there will be a total of $2^{12-2} = 1024$ possible inputs to the S-box pair. So for a uniform distribution each of the 2^8 possible outputs should occur exactly 4 times.

We write the result of D_1 for all possible outputs (X, Y) in an *S-box pair output distribution table*. Table 4.1 is the S-box pair output distribution table for S_1 and S_2 with $(s, t) = (0, 0)$. It shows the number of times each 8-bit output, from S_1 and S_2, occurs over all possible inputs given that bits 40 and 42 of the subkey are the same, and that bits 41 and 43 of the subkey are the same (equation 4.1). The S-box pair output distribution table is non-uniform and depends on the value of (s, t). Specifically, this is the "leak" which we exploit. For example, $D_1(8, D, 0, 0) = 5$ for S-box pair S_1 and S_2, which means that there is a probability of 5/1024 that the 8-bit joint output of S-boxes S_1 and S_2 will be $8D$ (hexadecimal) if bits 40 and 42 of the subkey are the same, and bits 41 and 43 of the subkey are the same.

Knowledge of the S-box pair output distribution table allows us to find the value of k for 1-round or 2-round DES using ciphertext only. Simply checking whether either '2A' or 'AA' have occured as the output of S-boxes S_1 and S_2 is sufficient (for example, if '2A' occurs then we know $k = 1$, since it cannot occur for $k = 0$). Of course, we would use weaknesses in all the S-box pair output distribution tables, not just S_1 and S_2. In fact, the most uniform distribution are due to the pair S_3 and S_4, and the most non-uniform distribution are due to the pair S_7 and S_8.

	S_2															
S_1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
1	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3
2	2	2	4	6	4	4	6	4	6	4	0	4	4	2	6	6
3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5
6	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
7	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3
8	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3
9	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
A	6	6	4	2	4	4	2	4	2	4	8	4	4	6	2	2
B	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5
C	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3
D	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5
E	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5
F	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Table 4.1 Joint output distribution of S-boxes S_1 and S_2

In any case, 2-round DES is not particularly secure, so we need to be able to calculate S-box pair output distribution tables for more rounds. We define a couple of functions in order to separate the X and Y components. Each of these functions operates on one S-box of a pair:

$$\begin{aligned} e(X,i,j) &\stackrel{\text{def}}{=} \#\{I \in \mathbf{Z}_2^6 | I[1]=i, I[0]=j, S_p = X\}, \\ f(Y,i,j) &\stackrel{\text{def}}{=} \#\{J \in \mathbf{Z}_2^6 | J[5]=i, J[4]=j, S_q = Y\}, \end{aligned} \tag{4.2}$$

so e and f count the number of times an S-box output has a particular value over all possible inputs if 2 (common) bits of the 6 input bits are fixed (unlike D_1 in which we count the values after fixing the key bits, rather than the S-box inputs). We define the following functions:

$$\begin{aligned} e_1(X) &= e(X,0,0) - e(X,0,1), \\ f_1(Y) &= f(Y,0,0) - f(Y,1,0), \\ d_1(X,Y) &= e_1(X) f_1(Y). \end{aligned}$$

Note that the value of d_1 neither depends on any data bits, nor any key bits. It depends only on the structure of the S-boxes. Finally, we can give an expression for D_1 in terms of the above functions:

$$D_1(X,Y,s,t) = 4 + (-1)^{s \oplus t} d_1(X,Y). \tag{4.3}$$

Equation 4.3 tells us that the output distribution table depends on the value of the key bits, and that there are two possible distribution tables (one for each of the possible values of $s \oplus t$). In particular, given a pair of S-boxes and an output distribution table, we can easily determine the value of $s \oplus t$.

For n-round DES, we are interested in the S-box pair output distribution table calculated by $D_{n/2}$, since each half of the plaintext passes through $\frac{n}{2}$ S-boxes (if n is odd, then $\mathcal{P}_H$ passes through $\frac{n}{2}+1$ S-boxes).

Equation 4.3 holds for one pair of S-boxes, but we can generalise it to the XOR of n pairs of S-boxes (i.e., applicable to $2n$ rounds):

$$D_n(X,Y,k) = 2^{10n-8} + (-1)^k d_n(X,Y),$$

where k is the XOR of the $4n$ key bits that affect the shared S-box input bits for n pairs of S-boxes, and $d_n(X,Y) = e_n(X)f_n(Y)$. We calculate e_n and f_n by discrete convolutions:

$$\begin{aligned} e_n(X) &= \sum_x d_\alpha(x) d_\beta(x \oplus X), \\ f_n(Y) &= \sum_y d_\alpha(y) d_\beta(y \oplus Y), \end{aligned}$$

where $\alpha + \beta = n$ (e.g., for $n = 4$, $\alpha = 2$, $\beta = 2$).

4.1.2 n-round Cryptanalysis

We can calculate the S-box pair output distribution tables for any number of rounds of DES, using the function D_n. However, D_n calculates the distribution for the *XOR of n pairs of S-boxes*. We now describe how we obtain the XOR of the S-boxes, and then how to perform the known-plaintext attack.

By substituting the first equation of the round function (equation 2.1) into the second, we get

$$R_{i+1} = R_{i-1} \oplus F(R_i, K_i), \qquad 0 \le i < n, \tag{4.4}$$

where the plaintext $\mathcal{P} = (R_{-1}, R_0)$, and the ciphertext $\mathcal{C} = (R_n, R_{n-1})$. Equation 4.4 can be expanded to get the following two expressions [6]:

$$\mathcal{P}_L \oplus \mathcal{C}_H = \bigoplus_{i=1}^{n/2} F(R_{2i-1}, K_{2i}), \tag{4.5}$$

$$\mathcal{P}_H \oplus \mathcal{C}_L = \bigoplus_{i=1}^{n/2} F(R_{2(i-1)}, K_{2i-1}). \tag{4.6}$$

So we can obtain 2 equations for the XOR of $n/2$ cipher functions given one plaintext/ciphertext pair. Equation 4.5 represents the XOR of the even-numbered rounds (2,4,6,...), and equation 4.6 represents the XOR of the odd-numbered rounds (1,3,5,...).

The final operation of the cipher function — the permutation P — is linear. Hence, if we apply P^{-1} to the XOR of the cipher functions, we obtain the XOR of the S-boxes [6]. This is the actual data required for the attack.

The known-plaintext attack is simple. We obtain the XOR of $n/2$ S-boxes for the even rounds, by

$$Q = P^{-1}(\mathcal{P}_L \oplus \mathcal{C}_H),$$

or, for the odd rounds,

$$Q = P^{-1}(\mathcal{P}_H \oplus \mathcal{C}_L),$$

and let $T_{X,Y}$ be the number of times that the XOR of outputs of the S-box pair is (X,Y). For example, if we are using the S-box pair distribution for S-boxes S_1 and S_2 then let x equal the 4-bit integer at $Q[31], \ldots, Q[28]$ and let y equal the 4-bit integer at $Q[27], \ldots, Q[24]$. We then increment the distribution counter $T_{x,y}$.

To decide whether k is 0 or 1, we evaluate the *indicator*

$$I = \sum_{X,Y} T_{X,Y} \, d_n(X,Y), \tag{4.7}$$

X/Y	S_1	S_2	S_2	S_3	S_3	S_4	S_4	S_5
0	568	384	1344	232	72	168	1088	152
1	-336	256	-768	0	32	0	0	-80
2	144	-320	-320	-192	32	-160	0	80
3	-304	-256	-256	-32	56	0	0	-24
4	-16	-256	-256	0	-32	-160	0	-112
5	-208	-256	768	160	0	0	-1024	112
6	336	192	-768	-32	0	160	64	-48
7	-176	256	256	-128	-32	0	0	48
8	-336	-192	-256	-32	0	0	0	80
9	176	-256	-256	192	-32	-160	-64	-80
A	-368	256	256	0	-32	0	-1024	80
B	464	256	256	-160	0	160	0	-80
C	-144	192	256	128	-32	0	0	-112
D	304	256	-704	-32	0	160	0	48
E	-176	-256	-256	-104	0	0	0	-112
F	72	-256	704	0	-32	-168	960	48

X/Y	S_5	S_6	S_6	S_7	S_7	S_8	S_8	S_1
0	232	376	376	408	536	1408	232	200
1	-160	200	-336	-304	-336	-1408	128	128
2	-224	-336	336	336	-368	1280	-128	0
3	160	-176	-304	-368	496	-1280	-160	0
4	96	-368	-208	144	-304	-1280	-32	0
5	-160	-208	176	-176	176	1280	0	0
6	-104	336	-176	144	144	-1280	0	-192
7	160	176	144	-176	-336	1280	-32	-136
8	0	-176	-368	-208	-208	-1280	0	-160
9	0	-336	336	176	16	1280	-32	-160
A	0	144	-336	-144	40	-1280	-32	32
B	0	304	304	232	-176	1280	0	-32
C	0	176	200	-336	304	1280	-192	-32
D	0	336	-176	304	-176	-1280	-104	32
E	0	-144	176	-336	-144	1280	160	160
F	0	-304	-144	304	336	-1280	192	160

Table 4.2 $e_3(X)$ and $f_3(Y)$ values for each pair of S-boxes

and if $I > 0$ then $k = 0$, otherwise $k = 1$ [6].

The number of plaintexts required for this attack to achieve a 97.7% success rate is given by

$$N = \frac{2^{20n-6}}{\sum_{X,Y} d_n(X,Y)^2}.$$

We will now illustrate how we would apply the S-box pairs attack to 6-round DES, using the best S-box pair S_7 and S_8.

Only the S-box pair output distribution tables for $n = 3$ S-box pairs are required, since each half of the 64-bit data is passed through the S-boxes exactly three times.

Table 4.2 to is used evaluate N, the number of plaintexts required for a 97.7% success rate:

S-Box Pair	S_1/S_2	S_2/S_3	S_3/S_4	S_4/S_5	S_5/S_6	S_6/S_7	S_7/S_8	S_8/S_1
N	$2^{25.2}$	$2^{24.9}$	$2^{31.4}$	$2^{26.0}$	$2^{27.0}$	$2^{25.4}$	$2^{21.8}$	$2^{28.6}$

We then proceed with the following algorithm which recovers the value to a linear expression for a key bit:

Algorithm 4.1.1 S-Box Pairs Attack (1 bit)

```
1. Initialise $T_i$, where $0 \leq i < 256$, to zero.
2. For each plaintext $\mathcal{P}_i$ and corresponding ciphertext $\mathcal{C}_i$, where $0 \leq i < N$, do the following:
    ◇ Let $Q \leftarrow P^{-1}(\mathcal{P}_H \oplus \mathcal{C}_L)$, where $P^{-1}$ is the inverse of the DES permutation $P$.
    ◇ Increment $T_j$, where $j$ is the 8-bit integer obtained from $(Q[7], \ldots, Q[0])$.
3. Calculate $I = \sum_i T_i d_3(X,Y)$, where $d_3(X,Y) = e_3(X) f_3(Y)$, and $X$ and $Y$ are the most significant and least significant 4-bit halves of $i$ respectively.
4. If $I > 0$ then $k = 0$, otherwise $k = 1$, where
        $k = K_1[4,5,6,7] \oplus K_3[4,5,6,7] \oplus K_5[4,5,6,7].$
```

The success rate of this algorithm is 97.7%. Algorithm 4.1.1 only evaluates the odd rounds. We can simultaneously evaluate the even rounds in the above algorithm, giving the value of another expression for a key bit. There is little additional overhead.

Algorithm 4.1.2 S-Box Pairs Attack (2 bits)

```
1. Initialise counters $T_{O,i}$ and $T_{E,i}$, where $0 \leq i < 256$, to zero.
2. For each plaintext $\mathcal{P}_i$ and corresponding ciphertext $\mathcal{C}_i$, where $0 \leq i < N$, do the following:
    ◇ Let $Q \leftarrow P^{-1}(\mathcal{P}_H \oplus \mathcal{C}_L)$.
    ◇ Increment $T_{O,j}$, where $j$ is the 8-bit integer obtained from $(Q[7], \ldots, Q[0])$.
    ◇ Let $Q \leftarrow P^{-1}(\mathcal{P}_L \oplus \mathcal{C}_H)$.
    ◇ Increment $T_{E,j}$, where $j$ is the 8-bit integer obtained from $(Q[7], \ldots, Q[0])$.
3. Calculate $I_O = \sum_i T_{O,i} d_3(X,Y)$, where $d_3(X,Y) = e_3(X) f_3(Y)$, and $X$ and $Y$ are the most significant and least significant 4-bit halves of $i$ respectively.
4. Calculate $I_E = \sum_i T_{E,i} d_3(X,Y)$ (as described in the previous step).
5. If $I_O > 0$ then $k_O = 0$, otherwise $k_O = 1$:
        $k_O = K_1[4,5,6,7] \oplus K_3[4,5,6,7] \oplus K_5[4,5,6,7]$
6. If $I_E > 0$ then $k_E = 0$, otherwise $k_E = 1$:
        $k_E = K_2[4,5,6,7] \oplus K_4[4,5,6,7] \oplus K_6[4,5,6,7]$
```

The probability of recovering 2 key bits is $(0.977)^2 = 0.95$, and the complexity of this attack is $O(N) + O(2^8)$. Considering the complementation property of DES, described in section 2.2.3, we find the remaining 53 key bits by exhaustive search.

Note that for 6-round DES *for any S-box pair* we need at most $2^{31.4}$ plaintexts to recover two key bits. This means that if we let $N = 2^{31.4}$ then we can recover 16 key bits by using 8 counters T_i, one for each S-box pair, with little additional time/memory cost.

4.1.3 (n–2)-round Cryptanalysis

Davies and Murphy mentioned that the attack may be optimised by guessing the $2 \times 6 = 12$ relevent subkey bits that affect the S-box pair in the final round (and excluding it from the distribution calculation)[1] [6].

In the basic attack, we were not concerned with the value of the plaintext and ciphertext data blocks. We only needed to know the XOR of each plaintext and corresponding ciphertext. For the (n–2)-round attack, we also need to know the ciphertext (actually, only the 8 LSBs of the ciphertext block). The principle behind this improved attack is that since we are working with equations of the form

$$\mathcal{P}_L \oplus \mathcal{C}_H = \bigoplus_{i=1}^{\frac{n}{2}} F(R_{2i-1}, K_{2i}), \tag{4.8}$$

where the left-hand side of the equation is known (easily calculated from the plaintext and ciphertext), and the right-hand side is the basis for the attack, we can reformulate this to get

$$\mathcal{P}_L \oplus \mathcal{C}_H \oplus F(\mathcal{C}_L, K_n) = \bigoplus_{i=1}^{\frac{n}{2}-1} F(R_{2i-1}, K_{2i}), \tag{4.9}$$

where we have, effectively, "peeled off" the last round. We do not know the value of the cipher function on the left-hand side but, remembering that only 12 subkey bits affect the output of an S-box pair, we try all $2^{12} = 4096$ possible values of the subkey. We now try the basic attack on the result of each the 4096 equations (equation 4.9), and get 4096 corresponding distributions. Only one of these is the correct distribution — the distribution corresponding to the 12 actual key bits. Provided that we have used sufficient data for the analysis, the correct distribution will also be the closest to the expected distribution. So we evaluate the indicator on all 4096 distributions, and assume that the highest (absolute) indicator will be the one corresponding to the correct distribution. Hence, we know the value of the 12 key bits (the number of the distribution), and the value of a linear expression for a key bit (from the sign of the indicator).

The main advantage of this suggested improvement is that we require substantially less data than the basic algorithm, enabling 16-round DES to be broken with 2^{52} known plaintexts [2, 6]. By estimating 12 key bits in the first and final rounds, we can potentially recover upto 26 key bits (including 2 linear expressions).

To ensure that the correct calculated distribution is distinguishable from the other 4095 distributions, we require 4 times more plaintext/ciphertext data than the basic algorithm. The probability of success for recovering 13 key bits is 0.719. The probability of success for recovering 13 key bits when using 8 times more plaintext/ciphertext data than the basic algorithm is 0.990.

Biham and Biryukov describe an efficient algorithm to implement the (n–2)-round attack [2]. The algorithm is split into a data collection phase and a data analysis phase.

Algorithm 4.1.3 Improved S-Box Pairs Attack (13 bits)

1. **Data Collection Phase:**
   Initialise $T_{i,j}$, where $0 \le i < 2^{10}$ and $0 \le j < 2^8$, to zero.

2. For each plaintext $\mathcal{P}_i$ and corresponding ciphertext $\mathcal{C}_i$, where $0 \le i < N$, do the following:

[1]Gilbert is also acknowledged as one of the people who investigated this.

- ◇ Let $Q \leftarrow P^{-1}(\mathcal{P}_L \oplus \mathcal{C}_H)$.
- ◇ Increment $T_{j,k}$, where j is the 10-bit integer obtained from $(\mathcal{C}_L[8], \ldots, \mathcal{C}_L[1], \mathcal{C}_L[0], \mathcal{C}_L[31])$, and k is the 8-bit integer obtained from $(Q[7], \ldots, Q[0])$.
  The integer j represents the data bits entering the S-box pair in the final round, and k represents the XOR of S-boxes S_7 and S_8 in the even-numbered rounds.

3. **Data Analysis Phase:**
   Initialise $U_{i,j}$, where $0 \leq i < 2^{12}$ and $0 \leq j < 2^8$, to zero.
4. For each $i \in \mathbf{Z}_2^{12}$ and $j \in \mathbf{Z}_2^{10}$ (i.e., all combinations of 12 key bits and 10 data bits — inputs to an S-box pair), do the following:
   - ◇ Calculate the 6-bit inputs to the S-box pair:
   $$\begin{aligned} I &\leftarrow (i[11], \ldots, i[6]) \oplus (j[9], \ldots, j[4]), \\ J &\leftarrow (i[5], \ldots, i[0]) \oplus (j[5], \ldots, j[0]). \end{aligned}$$
     Note: I and J share the two data bits $j[4]$ and $j[5]$.
   - ◇ Calculate the 8-bit output of the S-box pair: $s = (S_7(I), S_8(J))$
   - ◇ For each $k \in \mathbf{Z}_2^8$, let $U_{i,k} \leftarrow U_{i,k} + T_{j,s \oplus k}$.
     In this step, we are calculating the S-box output distribution table from the data obtained in the Data Collection Phase. The purpose of the XOR is to peel off the final round.
5. For all $i \in \mathbf{Z}_2^{12}$, calculate $I_i = \sum_j U_{i,j} d_2(X,Y)$, where $d_2(X,Y) = e_2(X) f_2(Y)$, and X and Y are the most significant and least significant 4-bit halves of j respectively.
   Note: We use d_2 because we have peeled off an S-box.
6. Let I_{max} be the maximum of $|I_i|$, for all $0 \leq i < 4096$.
7. If $I_{max} > 0$ then $k = 0$, otherwise $k = 1$, where
   $$k = K_2[4,5,6,7] \oplus K_4[4,5,6,7],$$
   and $(K_6^7, K_6^8) = (K_6[11], \ldots, K_6[0]) = max$.

With around 98% probability of success, we can recover 26 key bits including 2 linear expressions by using additional counters (for the odd rounds) in algorithm 4.1.3. The additional counters add very little overhead.

The complexity of algorithm 4.1.3 is $O(N) + O(2^{30})$.

4.2 Comparison to Differential Cryptanalysis

Both differential cryptanalysis and the S-box pairs/triplets attack are essentially based on a non-uniform distribution of S-box outputs. This enables us to determine the S-box inputs if we have a sufficient number of plaintexts and corresponding ciphertexts. Since the inputs to the S-boxes are the data and key, and the data is known, we can (potentially) determine some of the key.

In the S-box pairs attack, the non-uniform distribution is due to the common data bits in a pair of adjacent S-boxes. In differential cryptanalysis, the non-uniform distribution is due to the XOR of a pair of inputs and the XOR of the corresponding outputs of an S-box.

The main difference between differential cryptanalysis and the S-box pairs attack is that the S-box pairs attack relies heavily on the structure of the cipher function (the fact that we can obtain information from the S-box output that depends solely on the key), whereas differential cryptanalysis is fairly general (it depends mostly on the non-linear function, the S-boxes, and so can be tested on any block cipher[2]). Differential cryptanalysis is a chosen-plaintext attack, since we are interested in the XOR of a pair of plaintexts and ciphertexts, not the actual data itself.

On the other hand, the S-box pairs cryptanalysis also very similar to linear cryptanalysis; such as the way we collect data, then evaluate them, to determine the most likely keys. In S-box pair cryptanalysis and linear cryptanalysis, we confine ourselves to one part of the cipher (an S-box pair, or a linear path); differential cryptanalysis attempts to solve a substantial part of a subkey in one swoop — it has nothing to do with linear equations.

It is worth mentioning that DES is also resistant to differential cryptanalysis. Biham and Shamir have shown that any naïve modification of the S-boxes (changing the order, using random permutations in the rows, etc.) can significantly weaken DES with respect to differential cryptanalysis [3].

[2]Almost all modern block ciphers are designed to defeat differential and linear cryptanalysis.

Bibliography

[1] H. Beker and F. Piper. *Cipher Systems: The Protection of Communications.* Northwood Books, 1982.

[2] E. Biham and A. Biryukov. An improvement of Davies' attack on DES. In *Advances in Cryptology—Proceedings of EUROCRYPT '94*. Springer-Verlag, 1994.

[3] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, 1993.

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

[5] D. Davies. Some regular properties of the DES. In *Advances in Cryptology—Proceedings of CRYPTO '82*. Plenum Press, 1983.

[6] D. Davies and S. Murphy. Pairs and triplets of DES S-boxes. *Journal of Cryptology*, 8:1–25, 1995.

[7] D. W. Davies. Investigation of a potential weakness in the DES algorithm. unpublished, 1987.

[8] M. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer. Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard. Technical Report SEL 76-042, Information Systems Laboratory, Department of Electrical Engineering, Stanford University, 1976.

[9] A. G. Konheim. *Cryptography: A Primer.* John Wiley & Sons, 1981.

[10] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology—Proceedings of EUROCRYPT '93*. Springer-Verlag, 1993.

[11] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In *Advances in Cryptology—Proceedings of CRYPTO '94*. Springer-Verlag, 1994.

[12] S. Murphy. The cryptanalysis of FEAL-4 with 20 chosen plaintexts. *Journal of Cryptology*, 2:145–154, 1990.

[13] National Bureau of Standards. *Data Encryption Standard.* Federal Information Processing Standards Publication 46. US Department of Commerce, 1977.

[14] National Bureau of Standards. *Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard.* NBS Special Publication 500–20. National Bureau of Standards, 1980.

[15] R. A. Rueppel. *Analysis and Design of Stream Ciphers.* Springer-Verlag, 1986.

[16] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, 1994.

[17] A. Shamir. On the security of DES. In *Advanced in Cryptology—Proceedings of CRYPTO '85.* Springer-Verlag, 1986.