Acknowledgements

't Is pleasant sure to see one's name in print;
A book's a book, although there's nothing in't.
Byron

It is a real pleasure for me to thank all the people who helped me to realize this Ph.D. thesis.

In the first place I thank my promoters, prof. René Govaerts and prof. Joos Vandewalle, who allowed me to spend four years at the COSIC lab. Their guidance and support are greatly appreciated.

For the nice working atmosphere at COSIC, for helping me in various stages of my research and for extensively proofreading this manuscript, I thank my colleagues: Johan Borst, Antoon Bosselaers, Joan Daemen, Danny 'Godot' De Cock, Erik De Win, Stef Hoeben, Lars Knudsen, Keith Martin, Filip Mertens, Cristian Radu, Mark Vandenwauver, Luc van Linden, Bart van Rompay and Jan Verschuren.

I want to send special thanks to Bart Preneel, who put me on the path of cryptanalysis and continuously provided me with assistance and signposts along the way. For doing the necessary background paperwork, thank you to Rita De Wolf.

I want to thank prof. J.-J. Quisquater for reviewing this manuscript and for serving on the jury. I also thank the jury members prof. M. Van Barel and prof. J.L. Massey, and the chairman, Prof. Y.D. Willems.

The support of the F.W.O. (Fund for Scientific Research - Flanders (Belgium)) is greatly appreciated.

Also a well-meant thank you to Scouts Boven-Lo, for ever providing me with ample opportunities to divert my mind. Nobody was more clear in pointing out to me that there *is* a life after cryptography. Together with the people from Utimaco-Belgium you have done a great job in keeping my feet firmly on the ground.

Abstract

The subject of this thesis is the study of iterated block ciphers. The first part deals with the cryptanalysis of block ciphers. Most attacks on block ciphers are variants of differential and linear cryptanalysis. Firstly the principles of differential and linear cryptanalysis are explained. Afterwards a number of modifications are presented that allow these attacks to be extended. Techniques from probability theory allow the data processing phase of attacks to be improved in such a way that cryptanalysis of block ciphers used in special modes becomes possible. Attacks are then introduced that use relations that have key dependent probabilities, to cryptanalyse ciphers that rely on a nonlinear key addition (e.g.: IDEA, MAA). A new attack is presented and applied to the block cipher CAST.

The second part of this thesis deals with the design of iterated block ciphers. Further building upon the *Wide Trail* design strategy, construction methods for the building blocks of a round transformation are developed. Two new designs are presented, together with a first analysis of these designs.

Contents

List of Notations

List of Abbreviations

CBC : Cipher Block Chaining
CFB : Cipher FeedBack
ECB : Electronic Code Book
MAC : Message Authentication Code
OFB : Output FeedBack
PRN : Pseudo-Random Noise

List of analysed Cryptographic Algorithms

Akelarre
Blowfish
CAST
DES : Data Encryption Standard
IDEA : Improved Data Encryption Algorithm
LOKI
MAA : Message Authenticator Algorithm
MacGuffin
SHARK
SQUARE

List of other Cryptographic Algorithms

SAFER : Secure And Fast Encryption Routine
Threeway

List of Mathematical Symbols

$a_{i,j}$: the element on row i and column j of the matrix A
$\mathcal{C}_\theta$: the code associated with the mapping θ
$E[k](x)$: the encryption of the plaintext x under the key k
F : the round function of a Feistel Network
G^n : the vector space of binary n-tuples
l : the block length of a block cipher
k : the cipher key
k^r : the round key of round r
R : the number of rounds
$\mathcal{W}$: the Walsh transform
x : a vector
x_i : component i of the vector x
x' : the difference between the vectors x and x^*
X : a stochastic variable, or a matrix
X^t : the transpose of the matrix X
ρ : the round transformation of an iterated block cipher
$\cdot_{\mathtt{x}}$: a hexadecimal value
$\|$: the concatenation of two bit strings (vectors)
$\#V$: the cardinality of the set V

Chapter 1

Introduction

In contemporary society the science of cryptology is continually attracting more and more attention. Whereas in the past cryptology was an art, almost solely practiced by diplomats and military commanders (think of the Caesar cipher), the publication of the DES [39] in 1977 has been a trigger for academics to start public research in this field. Today the main activity of many companies is selling cryptographic products and performing consultancy on data security. Cryptology is used in everyday life by banks and credit card companies.

Cryptology receives a lot of attention because it deals with the protection of *information*. Information is nowadays a valuable resource. The development of fast electronic equipment for processing, transportation and storage of digital information has led to a new industrial revolution. Digital technology allows people all over the world to communicate with each other in a reliable way and at low cost. With the flick of a few keys it is possible to consult libraries of information on topics that range from the ancient abacus to the latest developments in zoology. On the one hand, the rapid spread of information allows people to make decisions based on recent information. On the other hand, sifting the interesting bits from the enormous amount of available, rapidly outdating, information has become a Sisyphean task. Thus, while most raw information is available for free and does not need any protection, processed information has good value, becomes an item of commerce and consequently needs protection. Equally important as economically valuable information, personal communication needs also protection against nosy individuals and organisations.

Digital information is transported over publicly accessible channels and can be as easily tapped as it can be processed. One of the goals of cryptology is to secure the transportation of information. This is done by *encryption*: the information is encoded by an algorithm that depends on a small piece of secret information, *the key,* in a way that makes it impossible to decode it without

knowledge of the key. Even if the information is not transported but only stored locally, it may be encrypted in order to prevent hackers from 'stealing' (copying) the information.

Nowadays cryptology no longer deals exclusively with the encryption of information. New concepts like cryptographic hash functions and public key algorithms have led to new applications like digital signatures and electronic money.

1.1 Cryptography and Cryptanalysis

The security of a design can be defined in three different ways [102]. The first definition is based on information theory: a system is called *unconditionally secure* if it can not be broken, even if the adversary has unlimited computing power. The second definition is based on complexity theory. The approach starts by defining a certain computational unit of operation. An algorithm is then 'feasible' if it can be executed in a number of operations that is (asymptotically) polynomial in the size of the input. The algorithm is considered secure if it can be proven that breaking the algorithm is an NP-complete problem. There is a confident and widespread belief that solving NP-complete problems requires a number of operations that increases exponentially in terms of the size of the input of the cipher. The study of the possible attacks on cryptographic algorithms is called *'cryptanalysis'*, while *'cryptography'* concerns the research of new algorithms and applications.

The third approach is based on practical methods. The security of an algorithm is evaluated by estimating the computing power that would be needed to break it. The estimates are based on the results of known attacks and the scrutiny of experienced cryptanalysts. This last approach has the advantage that it demands the weakest requirements from the algorithms and thus makes it the easiest approach for the production of practical algorithms that can be used in real life applications. In this approach there is a strong interaction between cryptography and cryptanalysis. The validity of the security level that is through this process assigned to an algorithm depends on the quality of the cryptanalysis that has been previously performed.

1.2 This Thesis

This thesis deals with the cryptanalysis and design of an important subset of cryptographic algorithms: block ciphers. In their simplest mode of use, block ciphers can be considered as Electronic Code Books (ECB [40]). Under the control of a relatively short key (56 bits to a few hundred bits) message blocks of a fixed size are replaced by other blocks. The encryption is *symmetric* because

the same key has to be used to decrypt the message.

There exist more sophisticated modes of use that give better secrecy [40]. Block ciphers are also often used as the compression function of hash functions [82, 103] and MAC algorithms [49]. In this thesis a construction is given that uses block ciphers for *asymmetric* encryption, where the encryption key can be revealed without endangering the secrecy because the decryption key cannot be easily recovered from it.

The security model most often used in the design of block ciphers, and also in this thesis, is the one based on the practical approach. Much attention is given to the analysis of existing designs. In this way it is possible to learn from existing failures and successes. And of course, every design that can be broken is one competitor less for our own designs !

1.3 Outline and Main Contributions

The first part of the thesis deals with the analysis of existing designs. Chapters 2 and 3 introduce basic definitions and well-known cryptanalytic techniques. Our original work for these chapters consists of Proposition 3.2, and the analysis of MacGuffin and a reduced version of Blowfish. The analysis of Blowfish is a joint work with Bart Van Rompay and Jan Verelst. The analysis of MacGuffin is a joint work with Bart Preneel and has been published in [110].

In Chapter 4 our improvements to differential cryptanalysis are presented. They involve the use of Bayes' rule and maximum likelihood estimators during the data processing phase of a differential or a linear attack. Also, the cryptanalysis of hash functions based on block ciphers is improved. These improvements are illustrated on the DES: used in 8-bit CFB mode and as compression function of a hash algorithm. This is a joint work with Bart Preneel and the results have been published in [105, 109].

In Chapter 5 we demonstrate that the average resistance against attacks of a cipher is not a good security measure. We present attacks that exploit differential characteristics with a probability that varies significantly over the key space. We illustrate this with two new attacks on IDEA, which were developed in cooperation with Lars R. Knudsen and Johan Borst and has been published in [14], and an attack on MAA, which was developed in cooperation with Bart Preneel and have been published in [106].

Chapter 6 introduces a new attack on block ciphers with an unbalanced round function and a small number of rounds. This attack is shown to break several members of the CAST block cipher family. It was developed in cooperation with Bart Preneel and Erik De Win and has been published in [112, 113].

The second part of the thesis deals with the design of new block ciphers. Chapter 7 elaborates on the function of the different components of the round

transformation and their construction. The emphasis lies on the construction of mappings with good diffusion properties. By associating mappings with linear codes we are able to give elegant constructions for optimal diffusion mappings. The chapter ends with a construction method for trapdoor ciphers, which was developed in cooperation with Bart Preneel and has been published in [114].

Chapter 8 presents two new block ciphers that were designed as part of the research. SHARK was developed in a joint effort with Antoon Bosselaers, Joan Daemen, Erik De Win and Bart Preneel and has been published in [111]. SQUARE was designed in cooperation with Joan Daemen and Lars R. Knudsen and has been published in [27, 28].

Chapter 9 concludes and discusses some open problems.

Appendix A gives a survey of existing block ciphers and known attacks.

Part I

Cryptanalysis of Iterated Block Ciphers

Chapter 2

Basic Concepts

In this chapter iterated block ciphers are defined. The Feistel structure and the uniform round structure are introduced and illustrated with examples. Block ciphers can be used, and will be studied, in several operation modes that are introduced in Section 2.2. A survey of the applications of block ciphers is presented. There then follows a discussion of the attack models that are most commonly used to analyse the security offered by a cipher.

2.1 Iterated Block Ciphers

The origin of iterated block ciphers appears to have been lost in time. C.E. Shannon describes *product ciphers* [122], that are formed by concatenating different transformations. The first to use a product of *identical* round transformations seems to have been H. Feistel [37].

Definition 2.1 *An* iterated block cipher *is an algorithm that transforms a plaintext block of a fixed size l into a ciphertext block of a fixed size l' under the influence of a key k, by a repeated application of an invertible transformation ρ, called* the round transformation. *Denoting the plaintext with x^0, the ciphertext with x^R and the intermediate values with x^r, the encryption operation can be written as:*

$$x^{r+1} = \rho[k^r](x^r) \qquad r = 0, 1, \ldots, R-1\,. \tag{2.1}$$

The values k^r are the round keys, that are derived from k by means of a key scheduling.

Here only block ciphers that do not expand the blocks will be considered ($l = l'$). The round transformation ρ is often built from different components, each with

a different functionality. The choice of the different components is an important part of the design strategy and will be treated in detail in part 2. For now, the following general description suffices:

key addition: The key addition inserts the unknown round key and mixes it with the intermediate value.

substitution layer: The substitution layer provides a complex nonlinear mixing of bits that are 'close' to one another in the bitstring that is processed. Since this layer involves complex operations on a few bits at a time, it is usually implemented with table lookups in *substitution boxes* (or *S-boxes*).

diffusion layer: The diffusion layer rearranges the bits such that bits that are 'close' to each other in round r, are not close to each other in round $r+1$. The diffusion layer usually consists of a simple operation that can be efficiently implemented without table lookups.

Not every block cipher follows this structure, e.g., the substitution layer sometimes depends on the key.

The main motivation for iterated block ciphers is the observation that a repeated application of a round transformation that is weak by itself can lead to a strong cipher [37]. It is clear that the round transformation ρ determines for a large part the resistance of the cipher to cryptanalysis. The two most used round transformation structures are the Feistel network [37] and the Uniform Transformation structure [36].

2.1.1 Feistel Network

In a Feistel network, the intermediate value x^r is split into two halves: s^r and t^r. The round transformation ρ uses a *round function* F, also called the *F-function*. The function F depends on the round key and takes one of the halves as input. The output of F is added to the other half. Subsequently both halves are swapped.

$$(s^{r+1}, t^{r+1}) = \rho[k^r](s^r, t^r) \Leftrightarrow \begin{cases} s^{r+1} &= t^r \\ t^{r+1} &= s^r \oplus F[k^r](t^r) \end{cases} \tag{2.2}$$

Here '$\oplus$' stands for the bitwise exor operation, but can be replaced by any other reversible operation. In the last round the swap of both halves is omitted. In this way the decryption operation is the same as the encryption operation with the round keys used in reverse order, independent of the choice for the round function F. Note that t^{R-1}, the text input of the round function in the last round is visible in the ciphertext.

The best known example of an iterated block cipher, the Data Encryption Standard (DES) [39], is a Feistel network. The DES uses 16 rounds, two of which are shown in Figure 2.1. The DES operates on 64-bit text blocks and uses a 56-bit key. The round function starts with an expansion E, which duplicates some of the 32 input bits to produce 48 output bits. The expanded input is then exored with the round key k^r. The result is split into eight 6-bit values that are used as indices in eight different lookup tables with 4-bit entries (the so-called S-boxes). The eight 4-bit entries of the tables are mixed by the bit permutation P to produce the output of the F-function. The DES also features an initial permutation IP of the plaintext block and a final permutation IP^{-1}. The cryptographic significance of these permutations remains unclear (but see Section 4.2.4 for some consequences that these permutations have on the security of certain modes of use).

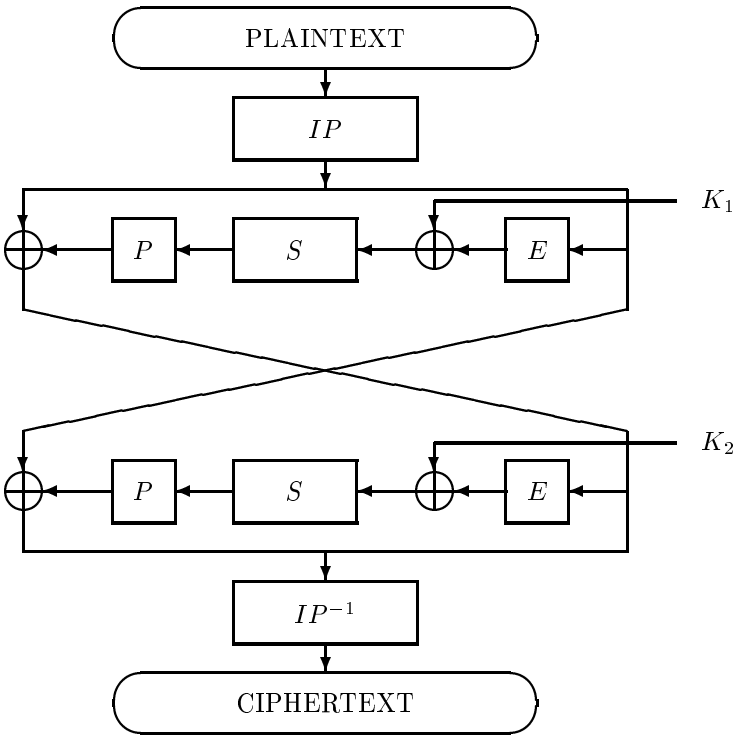


Figure 2.1: Two rounds of the DES, the most famous block cipher. It is a Feistel network. The full DES has 16 rounds.

The Feistel network can be generalized [12]: instead of splitting the intermediate value x^r into two equal halves, it is proposed that 'unbalanced' divisions be made. MacGuffin, the first example of a cipher that uses this generalized structure, is analysed in Section 3.5. Also the structure of IDEA [69] can be seen to be a generalisation of a Feistel network, in which the output of the F-function is added to both halves of x^r. A more detailed description of IDEA is provided in Chapter 5.

2.1.2 Uniform Transformation Structure

The uniform transformation structure is sometimes called 'the substitution-permutation structure (SP-structure)'. The round transformations of this block cipher structure are built by alternating nonlinear ('substitution') layers that operate on *all* the bits of the intermediate values and linear diffusion layers ('permutations'). (A 'linear functions' is defined in Definition 3.2.) Note that both the nonlinear layers and the linear layers have to be invertible, and are thus permutations. In the context of a linear layer, the term 'permutation' usually means 'bit permutation'. Because of this ambiguity and since the linear layer is not restricted to bit permutations, the classification 'SP-structure' is somewhat misleading. Figure 2.2 shows one round of SHARK, a block cipher with the uniform transformation structure.

The first block cipher with this round structure is an early version of Lucifer, described by Feistel in [36]. The best known example is probably SAFER [75]. Other examples are SHARK [111] and SQUARE [27], described in this thesis (cf. Chapter 8), and THREEWAY [24].

The uniform transformation structure is a very general round transformation. In fact, for any Feistel Network there exists an equivalent description in terms of a uniform transformation structure with a rather peculiar nonlinear layer.

2.2 Modes of Operation and Applications

A block cipher is a cryptographic primitive that substitutes l-bit strings under the influence of a key. It can be used in different modes and different cryptographic applications.

Encryption Modes

Four standard 'encryption' modes [40, 52] have been defined for block ciphers. These modes can be used in a symmetric encryption scheme, but also in the construction of a MAC, an asymmetric encryption scheme or a digital signature scheme.

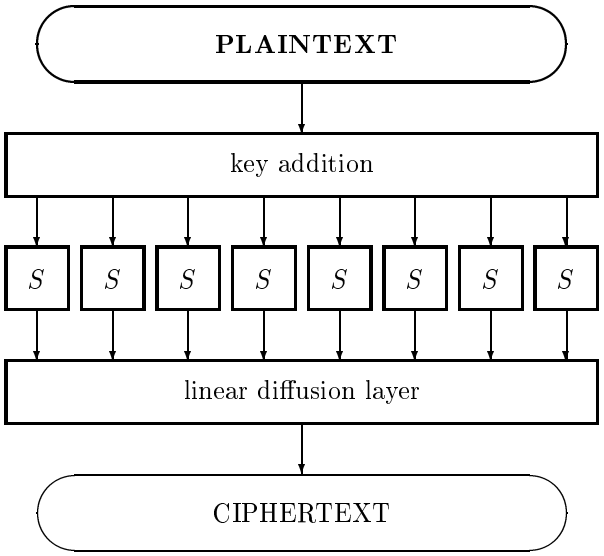


Figure 2.2: One round of SHARK, a block cipher with the uniform transformation structure. The nonlinear layer is implemented with eight parallel S-boxes.

The modes process a number of bits m at a time. For some modes $m = l$, for other $m < l$. If the string to be processed has a bit-length that is not a multiple of m, then it has to be *padded* using a *padding rule.* Let the text string consist of t l-bit blocks and denote one operation of the block cipher by $y = E[k](x)$, where k is the key. The blocks of the input string are denoted p^i ('plaintext'), the blocks of the output are denoted c^i ('ciphertext'), $0 \leq i < t$.

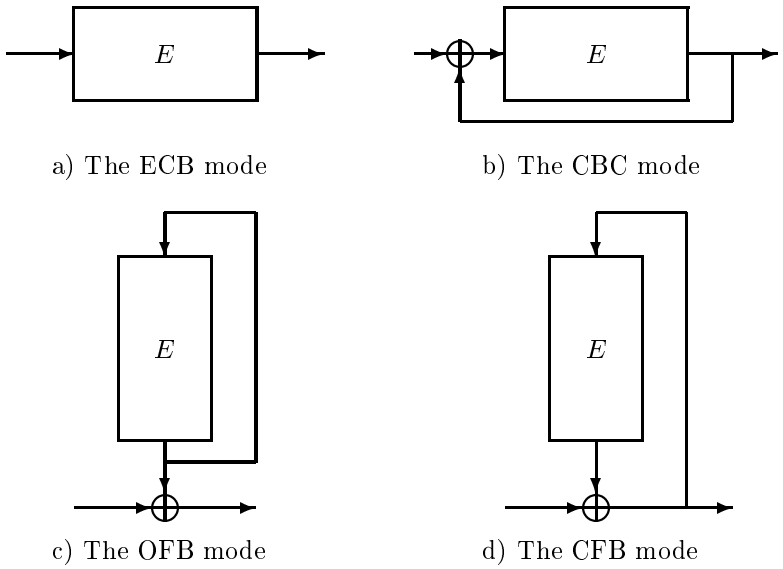
The four standard 'encryption' modes are the following.

1. The Electronic Code Book (ECB) mode is the basic encryption mode. Every block of the plaintext is encrypted independently. The blocks of the ciphertext are given by

$$c^i = E[k](p^i)\,.$$

   Since the blocks are encrypted independently, a repetition of blocks in the plaintext will lead to a repetition of blocks in the ciphertext.

2. In the Cipher Block Chaining (CBC) mode every ciphertext block depends on the corresponding plaintext block and on the previous ciphertext block.



a) The ECB mode b) The CBC mode

c) The OFB mode d) The CFB mode

Figure 2.3: The four standardized encryption modes of operation for block ciphers.

The first block depends on an initial value IV that can be public or secret.

$$\begin{aligned} c^0 &= E[k](p^0 \oplus IV) \\ c^i &= E[k](p^i \oplus c^{i-1}) \quad 1 \leq i < t \end{aligned}$$

Repeated blocks in the plaintext will no longer lead to repetitions in the ciphertext. By varying the initial value it is even possible to have different ciphertexts corresponding to the same plaintext.

3. The Output Feed Back (m-bit OFB) mode uses the block cipher to emulate a stream cipher. The ciphertext is produced by exoring a pseudo-random key-stream to the plaintext. The key-stream is generated by repeated encryption of the initial value. After every encryption the m leftmost bits of the output are concatenated to the key stream ($1 \leq m \leq l$). Denoting by p_m^i, c_m^i the i-th m-bit block of plaintext and ciphertext ($0 \leq i < tl/m$), by $\text{left}_m(x)$ the block that consists of the m leftmost bits of x and by $\text{right}_m(x)$ the m rightmost bits, the encryption process can be described

as follows.

$$\begin{array}{lcll} x^0 & = & IV & \\ x^i & = & E[k](x^{i-1}) & 1 \leq i < tl/m \\ c_m^i & = & p_m^i \oplus \text{left}_m(x^i) & 0 \leq i < tl/m \end{array}$$

The initial value has to be changed for every message in order to produce a different key stream. This is required to avoid a known plaintext attack. The blocks are encrypted independently, but the ciphertext blocks depend on the position of the blocks in the plaintext, so that repetitions in the plaintext will not lead to repetitions in the ciphertext.

4. The second stream mode is the Ciphertext Feed Back (m-bit CFB) mode. The difference between the OFB mode and the CFB mode lies in the updating of x. In the CFB mode x^i depends on x^{i-1} and c_m^{i-1}.

$$\begin{array}{lcll} x^0 & = & IV & \\ x^i & = & \text{right}_{l-m}(x^{i-1}) \| c_m^{i-1} & 1 \leq i < tl/m \\ c_m^i & = & p_m^i \oplus \text{left}_m(E[k](x^i)) & 0 \leq i < tl/m \end{array}$$

Here '$\|$' denotes concatenation of two bit strings. Each ciphertext block depends on the previous ciphertext block.

A pictorial representation of the modes is given in Figure 2.3. In this thesis only the ECB mode of the studied block ciphers is considered, except for the DES, where the ECB mode is already studied in depth in the literature [10, 29, 77] and thus the CFB mode is analysed here.

Hashing Modes

Block ciphers can also be used as the compression function of an iterated hash function. Iterated hash functions have a state variable h that is initialized to an initial value IV. In every iteration the round function (of the hash function) takes as input the state variable and a message block to update the value of the state variable. The main motivation behind using block ciphers to construct hash functions is the minimisation of design and implementation effort. In the simplest configurations the length of the hash result is equal to the block length l. A well-established example uses the following round function:

$$\begin{array}{lcl} h^0 & = & IV \\ h^i & = & f(p^i, h^{i-1}) = E[h^{i-1}](p^i) \oplus p^i \, . \end{array}$$

This mode is illustrated in Figure 2.4.

This scheme was proposed by S.M. Matyas, C.H. Meyer and J. Oseas [82], and is described in [103] together with eleven variants with an equivalent security

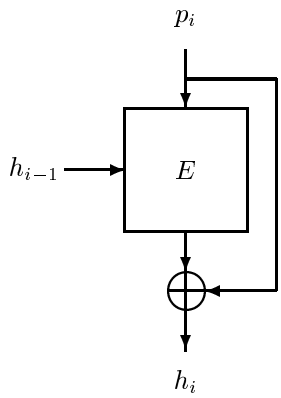


Figure 2.4: A hash mode for a block cipher.

level. Other configurations like MDC-2 and MDC-4 [16] produce results with a length of $2l$. B. Preneel and L.R. Knudsen give schemes with even larger lengths [63].

2.3 Applications

Block ciphers are used in many different cryptographic publications. Two important applications have already been mentioned in the previous section: symmetric encryption schemes are built from any of the four encryption modes; hash functions are built from the hashing modes.

Authentication Schemes

Another widely used scheme that uses a block cipher is the CBC-MAC [49, 51]. Section 2.4 explains the functionality of a MAC. The CBC-MAC uses a block cipher in CBC mode to process the message. The IV is initialized to zero. The last 'ciphertext' block gives the output of the MAC algorithm.

$$\begin{aligned} c^0 &= E[k](p^0) \\ c^i &= E[k](p^i \oplus c^{i-1}) \quad 1 \leq i < t \\ MAC[k](p) &= c^{t-1} \end{aligned}$$

A digital signature is an authentication scheme that uses asymmetric encryption. In [86] R.C. Merkle describes a digital signature scheme that uses a

block cipher as a cryptographic primitive. The scheme is based on the fact that for a secure block cipher the equation

$$y = E[k](x)$$

is difficult to solve for k if x and y are given. To sign a one-bit message with the basic signature scheme, the user selects two secret keys k^0 and k^1 and an arbitrary string x. The values $x, y^0 = E[k^0](x)$ and $y^1 = E[k^1](x)$ are placed in a public directory. If the user wants to sign a bit with value b, he reveals k^b. Everyone can verify that $y^b = E[k^b](x)$. The values y^0, y^1 can be used only once; therefore they are called *one-time signatures.* Merkle has shown a way to reduce the need for y-values.

Asymmetric Encryption

By building trapdoors into block ciphers it is possible to construct an asymmetric encryption scheme. In such a scheme the block cipher is the public key, and the trapdoor is the secret key. If Bob wants to send a message to Alice, he chooses a random key, encrypts his message with Alice's block cipher and sends it. Only Alice knows the trapdoor in her cipher and is able to recover the message without knowing the key. A practical realisation of such a scheme is given in Chapter 7.

Pseudo-Random Noise Generation

Block ciphers can be used as pseudo-random noise generators. The OFB mode in fact uses the block cipher to produce a pseudo-random bit sequence. Another possibility is to use the block cipher in *counter mode:* the pseudo-random stream is built by successive encryptions of a counter's output.

2.4 MAC Algorithms

A MAC scheme is a symmetric technique that is used to provide data origin authentication and data integrity. It associates with an input p and a secret key k a short bitstring $c = \mathrm{MAC}[k](p)$. The sender sends c along with p. The receiver, who shares the key k with the sender, will recompute the MAC on the received data and verify whether it matches the transmitted value.

MAC schemes can be built by using block ciphers (cf. supra) or by using algorithms that are developed only for this purpose. In Chapter 5 an attack is presented on MAA, a MAC algorithm that is very similar to a block cipher.

2.4.1 Attacks on MACs

Finding an arbitrary message and a corresponding MAC value without knowledge of the key is called an *existential forgery*. If a cryptanalyst has control over the message, this attack is called a *selective forgery*. A forgery is called verifiable when an adversary knows that the MAC is correct with probability close to 1. However, for some applications it might be sufficient that a MAC is correct with probability significantly higher than 1/(the number of all possible MAC values). A *key recovery* attack is more serious than a forgery attack: an adversary who can recover the key can perform arbitrary selective forgeries.

2.5 Security and Security

When the security of an algorithm is evaluated, different approaches are possible, which may result in different answers to the question: "Is this cipher secure ?"

In a real-world top secret application, every detail of an algorithm is likely to be a well-kept secret. Under these conditions a cryptanalyst faces a very tough job in trying to recover the key or the plaintext. Nevertheless most people will consider a block cipher secure only if it resists attacks under *Kerckhoffs' assumption*. This assumption states that the enemy cryptanalyst knows all the details of the algorithm, except for the value of the secret key.

A first classification of attacks can be made according to the result that the attack has [59]:

- A *key recovery* attack enables the cryptanalyst to recover the actual key that was used.
- A *global deduction* provides the cryptanalyst with enough information about the key to encrypt and decrypt messages at will, but without revealing the actual key that was used.
- A *local deduction* occurs when the cryptanalyst can encrypt or decrypt one message (without asking the legitimate user to do it for him).
- An *information deduction* occurs when the cryptanalyst learns something about the key, plaintexts or ciphertexts. This might be, for instance, the distribution of the plaintext or a relation between some bits of the key.

A second classification can be made according to the access that is given to the cryptanalyst. If a cryptanalyst mounts an attack using only the information he gets under Kerckhoffs' assumption and the values of some ciphertexts, this attack is called a *ciphertext-only attack*. This requires that the cryptanalyst knows some general statistics about the corresponding plaintexts, e.g., that it is

ASCII-coded text. An attack is considered successful not only if it recovers the key but also if it allows the plaintext to be obtained from a given ciphertext in any other way. A ciphertext-only attack is the most dangerous attack because it is arguably the most realistic. If a cipher can be broken using a ciphertext-only attack, then its use in practice is very limited. For most ciphers designed today there are no known ciphertext-only attacks. An example of a ciphertext only attack is the linear attack on the DES, reduced to eight rounds [77].

An easier attack is a *known plaintext attack*. The assumption here is that the cryptanalyst has access to some ciphertexts and the corresponding plaintexts. An attack needing only a few known plaintexts might be practical if the system is used to send documents in standard format or with (partly) predictable contents, or if plaintext is later on released, before the key has been changed. Notice that known plaintext attacks always exhibit a certain symmetry, since they make no functional distinction between plaintext and ciphertext. Most linear attacks are known plaintext attacks [77]. Another variant is the attack on CAST described in Chapter 6, that is based on the fact that the round function of CAST is not surjective. A known plaintext attack is successful if it recovers the key or if it allows a previously unknown plaintext to be calculated from a ciphertext.

A *chosen plaintext attack* gives even more access and capacity to the cryptanalyst, since he is allowed to choose some plaintexts and then to get the corresponding ciphertexts encrypted under the unknown key. Variants of this attack are the chosen ciphertext attack and the adaptive chosen plaintext attack, where the cryptanalyst can choose the next plaintext after processing the ciphertext corresponding to the previous choice. Practical situations where this attack can be mounted are very rare indeed. Differential attacks are typically chosen plaintext attacks [10, 14].

In a *related-key attack* the cryptanalyst has ciphertexts at his disposal that are the results of encrypting a set of plaintexts under different keys. Related-key attacks are studied in [11, 58, 55].

It is clear that the attacks in this list are ordered such that they become less and less applicable in practice. Cryptographers sometimes object to the practice of evaluating the strength of their designs against chosen plaintext related-key attacks, since they are not very realistic. Cryptanalysts however argue that the different attacks make it possible to rank different ciphers: a cipher that does not succumb to a related-key attack is considered to be more secure than one that does, irrespective of the actual usage in practice. Since the security of a practical cipher can almost never be proven, considering these "impractical" attacks offers a kind of safety margin. Also, most chosen plaintext attacks involve a certain freedom in the choice of plaintexts and thus they can often be converted into a known plaintext attack by simply collecting known plaintexts until some desired texts have been obtained.

2.6 Conclusions

This introductory chapter presented the definition of an iterated block cipher. The Feistel Network and the Uniform Transformation Structure, the two most used round transformations in an iterated block cipher were explained. The standard modes of operation for a block cipher were shown and a taxonomy was given for attack models used in block cipher cryptanalysis.

Chapter 3

Basic Cryptanalytic Tools

If the only tool you have is a hammer,
you tend to see every problem as a nail.

This chapter introduces the necessary mathematical tools for discussing linear and differential cryptanalysis.

The standard versions of the differential and linear attacks are explained. Also, the principles of a differential-linear attack and truncated differentials are presented. A new characterisation of a differential is given. All these attacks share a number of underlying assumptions that are made in order to simplify the problem of estimating the performance of the attacks, or equivalently the resistance of the analysed ciphers against the attacks. Limitations of the validity of the assumptions are shown.

The last sections are illustrations of the basic attacks: a linear and a differential attack on MacGuffin [12], which have been published in [110], and a second order differential attack on a reduced version of Blowfish [121]. In later chapters ciphers will be analysed where these basic assumptions are no longer valid.

3.1 Mathematical Tools

This section introduces definitions for Boolean functions and substitution boxes. The link between both is made by a special Boolean function: the characteristic function of the substitution box. The Walsh transform allows the tables that will be used in differential and linear cryptanalysis to be efficiently calculated.

3.1.1 Boolean Functions

The definition of a Boolean function that is used in cryptography deviates a bit from the formal mathematical approach as presented in [117]. In cryptographic literature, usually only two-element Boolean algebra is used. Furthermore the zero element and the unit element of this algebra are often identified with the integers 0 and 1, so that addition and other functions over $\mathbb{Z}$ of the 'Boolean' variables are well-defined. Let G^n denote the vector space of binary n-tuples. Elements of G^n are represented as row vectors $x = (x_1, x_2, \ldots, x_n)$.

Definition 3.1 *A* Boolean function f *is defined as a mapping from* G^n *to the set* $\{0, 1\}$.

$$f : G^n \to \{0, 1\} : (x_1, x_2, \ldots x_n) = x \mapsto f(x)$$

Addition modulo two, also known as exor, will be denoted by $\oplus$. The associated function $\hat{f}$ of the Boolean function f, is defined as $\hat{f}(x) = 1 - 2f(x)$. Because of the limited range of a Boolean function, this relation can also be expressed as $\hat{f}(x) = (-1)^{f(x)}$. It is easy to verify that $h = f \oplus g \Leftrightarrow \hat{h} = \hat{f} \cdot \hat{g}$.

Definition 3.2 *A Boolean function* f *is* linear *if and only if* $\forall x, y : f(x \oplus y) = f(x) \oplus f(y)$.

Denote by e^i the vector with $e^i_j = 0, \forall j \neq i$ and $e^i_i = 1$. Since every vector x can be written as a linear combination of $e^i, i = 1, \ldots n$:

$$x = \bigoplus_{i=1}^{n} x_i e^i \,,$$

a linear function is completely characterized by its images of the *basis* $\{e^1, \ldots, e^n\}$:

$$f(x) = f(\bigoplus_{i=1}^{n} x_i e^i) = \bigoplus_{i=1}^{n} x_i f(e^i) \,. \tag{3.1}$$

Equation (3.1) implies that every linear function l_ω can be written as

$$l_\omega(x) = \omega \bullet x \,,$$

where the vector ω is defined as $\omega_i = f(e^i)$ and the *dot product* of two vectors is defined as

$$x \bullet y = x \cdot y^t = \bigoplus_{i=1}^{n} x_i \cdot y_i \,. \tag{3.2}$$

The associated function $\hat{l}_\omega$ can be written as

$$\hat{l}_\omega(x) = 1 - 2(\omega \bullet x) = (-1)^{\omega \bullet x}.$$

The *Hamming distance* between two functions f, g is equal to the number of function values in which they differ.

$$\begin{aligned} d(f,g) &\triangleq \#\{x \in G^n \mid f(x) \neq g(x)\} && (3.3)\\ &= \sum_{x \in G^n} f(x) \oplus g(x) && (3.4)\\ &= 2^{n-1} - \frac{1}{2} \sum_{x \in G^n} \hat{f}(x)\hat{g}(x) && (3.5) \end{aligned}$$

The *Hamming weight* of a function f is equal to its distance from the constant zero function f_0. The Hamming weight of a vector x is equal to the number of non-zero components.

$$w_h(x) \triangleq \#\{i \mid x_i \neq 0\} \tag{3.6}$$

The *correlation* between two functions f, g is related to the probability that their values are equal.

$$\begin{aligned} c(f,g) &= 2^{-n} \cdot (\#\{x \mid f(x) = g(x)\} - \#\{x \mid f(x) \neq g(x)\}) && (3.7)\\ &= 1 - 2^{1-n} \cdot d(f,g) && (3.8)\\ &= 2^{-n} \sum_x \hat{f}(x)\hat{g}(x) && (3.9) \end{aligned}$$

This can be rewritten as:

$$\Pr(f(x) = g(x)) = \frac{1 + c(f,g)}{2}. \tag{3.10}$$

Lemma 3.1 *The correlation between two different linear functions is zero.*

Proof: The correlation between two linear functions l_α, l_β is given by:

$$\begin{aligned} c(l_\alpha, l_\beta) &= 2^{-n} \sum_x \hat{l}_\alpha \hat{l}_\beta \\ &= 2^{-n} \sum_x (-1)^{\alpha \bullet x} (-1)^{\beta \bullet x} \\ &= 2^{-n} \sum_x (-1)^{(\alpha \oplus \beta) \bullet x}. \end{aligned}$$

If $\alpha \neq \beta$ the sum is zero. If $\alpha = \beta$, the functions are equal and the correlation is one. ■

The *autocorrelation function* $\hat{r}_f : G^n \to \mathbb{Z}$ of the function $\hat{f}$ is defined as the *convolution* of the function with itself.

$$\hat{r}_{\hat{f}}(x) \triangleq \sum_v \hat{f}(v)\hat{f}(x \oplus v) \tag{3.11}$$

The *cross-correlation function* $\hat{c}_{\hat{f},\hat{g}}(x)$ of two functions $\hat{f}$ and $\hat{g}$ is equal to the convolution of the functions.

$$\hat{c}_{\hat{f},\hat{g}}(x) \equiv (\hat{f} \otimes \hat{g})(x) = \sum_v \hat{f}(v)\hat{g}(x \oplus v) \tag{3.12}$$

The correlation between f and g can be calculated from the cross-correlation function:

$$c(f,g) = 2^{-n}\hat{c}_{\hat{f},\hat{g}}(0).$$

3.1.2 Walsh Transform

The Walsh transform can be defined for any real-valued function with domain G^n [8].

Definition 3.3 *The* Walsh transform $F : G^n \to \mathcal{R}$ *of a real-valued function* $f : G^n \to \mathcal{R}$ *is defined by:*

$$F(\omega) \triangleq \mathcal{W}(f)(\omega) \triangleq \sum_x f(x) \cdot (-1)^{\omega \bullet x} = \sum_x f(x) \cdot \hat{l}_\omega(x)$$

The Walsh transform of the associated function is denoted with $\hat{F}(\omega)$.

$$\begin{aligned}
\hat{F}(\omega) &\triangleq \mathcal{W}(\hat{f})(\omega) && (3.13)\\
&= \sum_x \hat{f}(x) \cdot (-1)^{\omega \bullet x} && (3.14)\\
&= \sum_x \hat{f}(x) \cdot \hat{l}_\omega(x) && (3.15)\\
&= 2^n \hat{c}(f, l_\omega) && (3.16)
\end{aligned}$$

The relation between $\hat{F}(\omega)$ and $F(\omega)$ is given by:

$$\hat{F}(\omega) = 2^n \delta(\omega) - 2F(\omega)\,,$$

where the function $\delta(\omega)$ equals zero, except when $\omega = 0$, where it is one. The evaluation of the Walsh transform of a Boolean function at a point ω gives the correlation of the Boolean function and the linear function $\hat{l}_\omega$.

Analogous to the fast Fourier transform routines, there also exists a fast algorithm for the Walsh transform, requiring $\mathcal{O}(n\,2^n)$ operations. Another parallel with the Fourier transform is that the inverse Walsh transform only differs in a constant factor from the forward Walsh transform:

$$f(x) = \mathcal{W}^{-1}(F)(x) = 2^{-n}\mathcal{W}(F)(x)\,. \tag{3.17}$$

Equations (3.16) and (3.17) show that like the Fourier transform, which can be seen as the decomposition of a function into sinusoidal components, the Walsh transform of a Boolean function can be seen as the decomposition of the function into its linear components $\hat{l}_\omega$.

There is a Walsh version of Parseval's theorem:

$$\sum_\omega (\hat{F}(\omega))^2 = 2^n \sum_x (\hat{f}(x))^2\,. \tag{3.18}$$

When f is a Boolean function this becomes

$$\sum_\omega (\hat{F}(\omega))^2 = 2^{2n}\,,$$

and using (3.16) this results in a bound on the correlations of an arbitrary Boolean function with all the linear functions:

$$\sum_\omega \hat{c}^2(\hat{f}, \hat{l}_\omega) = 1\,. \tag{3.19}$$

It is easy to see that (3.19) implies that for any Boolean function f

$$\max_\omega \hat{c}^2(\hat{f}, \hat{l}_\omega) \geq 2^{-n}\,.$$

The functions that reach this lower bound, are called *bent functions* [116].

Definition 3.4 *f is a* bent function *if and only if*

$$\forall\omega:\ |\hat{F}(\omega)| = 2^{n/2}\,.$$

Bent functions only exist if n is even.

The Walsh transform of the convolution of two functions equals the product of the Walsh transforms of the functions.

$$\mathcal{W}(\hat{f} \otimes \hat{g})(\omega) = \hat{F}(\omega) \cdot \hat{G}(\omega) \tag{3.20}$$

The Wiener-Khintchine theorem is a corollary of (3.20):

$$\hat{R}_{\hat{f}}(\omega) = (\hat{F}(\omega))^2\,. \tag{3.21}$$

3.1.3 Substitution Boxes

Definition 3.5 *An* $n \times m$ substitution box (S-box) *is a mapping*

$$s : G^n \to G^m : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_m) = s((x_1, \ldots, x_n))$$

S-boxes are usually studied in terms of their *component functions* and the linear combinations of their component functions.

$$\begin{aligned} s_i(x) &= e^i \bullet s(x), \qquad i = 1, \ldots, m \\ s(x) &= \bigoplus_{i=1}^{m} s_i(x) \cdot e^i \end{aligned}$$

To study the properties of the component functions, the characteristic function of an S-box is defined [18].

Definition 3.6 *The* characteristic function θ_s *of an S-box is the Boolean function*

$$\theta_s : G^{n+m} \to \{0,1\} : (x\|y) = (x_1, \ldots, x_n, y_1, \ldots, y_m) \mapsto \begin{cases} 1 & \text{if } s(x) = y \\ 0 & \text{else} \end{cases}$$

This leads to the following relation between the characteristic function and the component functions in the Walsh domain:

$$\begin{aligned} \Theta_s(\alpha\|\beta) &= \mathcal{W}(\theta_s)(\alpha\|\beta) \\ &= \sum_x \sum_y \theta(x\|y) \cdot \hat{l}_{(\alpha\|\beta)\bullet(x\|y)} \\ &= \sum_x \hat{l}_{(\alpha\|\beta)\bullet(x\|s(x))} \\ &= \sum_x \widehat{(\beta \bullet s(x))} \cdot \hat{l}_\alpha(x) \\ &= \mathcal{W}(\widehat{\beta \bullet s(x)})(\alpha) \end{aligned}$$

The table that lists $\Theta_s(\alpha\|\beta)$ for all values of $\alpha\|\beta$ is called the *Linear Approximation Table (LAT)*, and is used in linear cryptanalysis. From (3.16) it follows that the LAT entries are proportional to the correlations between linear combinations of the outputs of the S-box and linear functions of the input, sometimes called input-output correlations.

$$\mathrm{LAT}_s(\alpha\|\beta) = 2^n c(\beta \bullet s(x), \alpha \bullet x) \tag{3.22}$$

The λ-*parameter* of an S-box is defined as the maximal input-output correlation of the S-box.

$$\lambda_s = \max_{\alpha,\beta \neq 0} c(\beta \bullet s(x), \alpha \bullet x) \tag{3.23}$$

The use of the characteristic function in differential cryptanalysis will be discussed in Section 3.2.

An S-box is called *a linear S-box* if all its component functions are linear functions. A linear $n \times m$ S-box can be described using an $n \times m$ Boolean matrix. Let $\omega_{ij} = s_j(e^i)$, then

$$\begin{array}{rcl} y & = & s(x) \\ & \Updownarrow & \\ (y_1, y_2, \ldots, y_m) & = & (x_1, x_2, \ldots, x_n) \cdot \begin{bmatrix} \omega_{11} & \omega_{12} & \ldots & \omega_{1m} \\ \omega_{21} & \omega_{22} & \ldots & \omega_{2m} \\ \ldots & \ldots & \ldots & \ldots \\ \omega_{n1} & \omega_{n2} & \ldots & \omega_{nm} \end{bmatrix} \\ & \Updownarrow & \\ y & = & x \cdot \Omega\,. \end{array}$$

For a linear S-box it holds that all the linear combinations of the component functions of the S-box are also linear functions.

$$\forall b \in G^m, \exists a \in G^n : \; b \bullet s(x) \equiv a \bullet x \tag{3.24}$$

This follows from straightforward calculation:

$$\begin{array}{rcl} b \bullet s(x) & = & b \cdot y^t \\ & = & b \cdot \Omega^t \cdot x^t \\ & = & a \cdot x^t, \end{array}$$

where $a = b \cdot \Omega^t$, or $a = s^t(b)$.

3.2 Differential Cryptanalysis

E. Biham and A. Shamir were the first to give a general description of the cryptanalytic technique that is known as differential cryptanalysis [10]. Similar techniques had already been used in the public world to cryptanalyse a specific cipher proposal on an ad-hoc basis [43, 90]. Also, the designers of the DES claim to have known about differential cryptanalysis back in 1974, when they designed the algorithm [21].

3.2.1 Differences

Differential cryptanalysis is a chosen plaintext attack. In fact only the plaintext *differences* are chosen. Plaintext differences can be defined in several ways: every group operation $*$, that is well-defined over the set of plaintexts, has a corresponding difference operation Δ: if $\mathrm{inv}(a)$ denotes the inverse element of a with respect to $*$, then

$$a\Delta b = a * \mathrm{inv}(b)\,. \tag{3.25}$$

The difference between two instances of the same variable is written as $a' = a\Delta a^*$. The difference table of a mapping is a compact way to present the information about its differential properties.

Definition 3.7 *Let $s : V \to W$ be an arbitrary mapping. The* difference table *e_s of s is defined by*

$$e_s((a,b)) = \#\{c \in V \mid s(c * a)\Delta s(c) = b\}$$

Later, $e_s((a,b))$ will be abbreviated to $e_s(a\|b)$. The value $e_s(0\|0)$ is always equal to the cardinality of V. If s is linear with respect to the $*$ operation, $s(c * a) = s(c) * s(a)$. In this case $e_s(a\|b)$ is always zero, except if $s(a) = b$, in which case $e_s(a\|b)$ is equal to $e_s(0\|0)$.

The *δ-parameter* of a mapping s is defined by

$$\delta_s = \frac{1}{e_s(0\|0)} \cdot \max_{a \neq 0, b} e_s(a\|b)$$

The product $\delta_s \cdot e_s(0\|0)$ is called *the differential uniformity* of the mapping s.

Sometimes the input difference and the output difference are only partially specified, e.g. in the case of truncated differentials (cf. Section 3.2.4). In this situation the reduced difference table can be used.

Definition 3.8 *Let ϕ, ψ be two surjective mappings, $\phi : V \to V_1$ and $\psi : W \to W_1$, with $V_1 \subseteq V$ and $W_1 \subseteq W$. Then the* reduced difference table *of s is defined as the difference table that merges the rows where $\phi(a_1) = \phi(a_2)$ and the columns where $\psi(b_1) = \psi(b_2)$.*

$$\begin{aligned} r_s(ab) &= \#\{c, d \in V \mid \phi(c)\Delta\phi(d) = a \text{ and } \psi(s(c))\Delta\psi(s(d)) = b\} \\ &= \sum_{\phi(c)=a} \sum_{\psi(d)=b} e_s(cd) \end{aligned}$$

Usually the sets V and W are identified with G^n and G^m and the difference used is exor. The corresponding difference table is called an exor-table.

The exor-table of a mapping can be calculated as the convolution of the characteristic function with itself:

$$\begin{aligned}(\theta_s \otimes \theta_s)(v\|w) &= \sum_{x\|y} \theta(x\|y) \cdot \theta((x\|y) \oplus (v\|w)) \\ &= \sum_{x} \theta((x\|s(x)) \oplus (v\|w)) \\ &= \#\{x \in G^n \mid s(x \oplus v) = s(x) \oplus w\} \\ &= e_s(v\|w)\end{aligned}$$

3.2.2 Differential Characteristics

A differential attack is based on the fact that, for an adequate choice of Δ, given a pair of plaintexts with a certain input difference $x' = x\Delta x^*$, it is possible to predict the differences of the intermediate values in each round of the encryption algorithm with a certain overall probability p, even if the used key is unknown. The tuple $(x^{0\prime}, x^{1\prime}, \ldots x^{R-1\prime})$ consisting of the known input difference of the first round and the predicted differences of the intermediate values is called a *differential characteristic*. Pairs of plaintexts that exhibit the predicted intermediate differences are called *good pairs*, they *follow* the characteristic. The other pairs are *wrong pairs*. Typically Δ and the differences $x^{r\prime}$ are chosen to maximize p.

The cryptanalyst finds the key in the following way. He chooses plaintext pairs with the correct difference and gets the corresponding ciphertext pairs. For each pair of ciphertexts he verifies whether the output difference and the actual output values are compatible with the predicted differences $x^{R-1\prime}$. If not, he knows that this pair is a wrong pair, and he discards it, the pair is *filtered*. If the pair is compatible, the cryptanalyst starts making assumptions about (a part of) the key. With the assumed value of the key the cryptanalyst can partially calculate some of the intermediate values of the encryption algorithm. For most assumed values of the key, there will again arise incompatibilities between the calculated intermediate differences and the intermediate differences predicted by the differential characteristic.

If the cryptanalyst only calculates intermediate values of the last round, the differential attack is called a *1R-attack*. In an *nR-attack* intermediate values of the last n rounds are calculated. How far back a cryptanalyst can calculate depends on the specific structure of the cipher.

Key values that cause no incompatibilities are called *suggested key values*. For good pairs, the correct key value will be among the suggested key values. Wrong and good pairs will suggest wrong key values. Only if the correct key value is suggested significantly more often than the wrong key values will the

differential attack succeed. It is often assumed that the wrong suggested key values are randomly distributed. The *signal-to-noise ratio (S/N)* of the attack is defined in the following way. Let f be the probability that a wrong pair survives the filtering. Define s as the probability that a wrong key value is suggested (s equals the average number of suggested key values divided by the number of possible key values). Then

$$S/N = p/(f \times s)\,. \tag{3.26}$$

The number of required pairs is proportional to p^{-1} if S/N is well above one. For S/N values close to one the number of required pairs increases very fast.

Active and Passive Components

An encryption algorithm often contains transformations that are composed of a number of parallel mappings. For instance, the round function of the DES contains eight parallel S-boxes. A nonzero input difference to the transformation does not necessarily lead to a nonzero input difference for all the parallel component mappings. If the two inputs to a mapping are equal, its outputs are also equal. Thus a zero input difference leads to a zero output difference with probability one. A mapping with zero (predicted) input difference is called *passive*, an *active* component is one that has a nonzero input difference.

3.2.3 Differentials

In [69] X. Lai and J.L. Massey observe that in a differential attack often only a few of the predicted differences $x^{r\prime}$ are actually used. The R-tuple $(x^{0\prime}, x^{1\prime}, \ldots x^{R-1\prime})$ can be reduced to a tuple $(x^{0\prime}, x^{r_1\prime}, \ldots, x^{r_{a-1}\prime})$ that only contains the predicted values that are actually used in the attack. A tuple that does not specify all of the intermediate differences, is called a *differential*. The probability of a differential is calculated by adding the probabilities of all the characteristics that reduce to this differential. The probability of the differential is a more accurate measure for the success rate of a differential attack.

Define the *maximal differential* as the tuple that specifies only the intermediate differences that are actually used in the differential attack. The following proposition is a new result; it provides new insight into the relation between the S/N-ratio and the success probability of a differential attack.

Proposition 3.2 *In a differential attack (as described previously in Section 3.2.2) the correct key value is* only *suggested by a pair that follows the maximal differential of the attack.*

Proof: Suppose the pair survives the filtering. This means that it exhibits ciphertext differences that are compatible with the differential. To verify whether

a particular key value k is suggested, the intermediate differences are calculated, using k as the key. Only if the calculated differences agree with the differences predicted by the differential is the key value suggested.

If k is the correct key value, the calculated differences are equal to the actual differences. Therefore, if the calculated differences of the pair agree with the predicted differences of the differential, the actual differences agree with the predicted differences and the pair follows the maximal differential.

Thus if the correct key value is suggested, this implies that the pair is a good pair. ■

Wrong key values are suggested both by the good and the wrong pairs. If $f \gg p$, the number of wrong pairs that survives filtering vastly exceeds the number of good pairs. In this case, the contribution of the good pairs to the number of suggestions for a wrong key value can be neglected.

From this observation it follows that a differential attack can work in two different ways: if $p \gg f \cdot s$, or $S/N \gg 1$ the correct key value can be found by looking for the most suggested key value. This is the standard differential attack. If $p \ll f \cdot s$, or $S/N \ll 1$, the correct key value can be found by searching for the *least* suggested key value. This is because the 'noise' of the wrong pairs will not be added to the counts of the correct key. In Section 5.2.4 an attack is described that uses this principle. Only if $S/N \approx 1$ does a differential attack become very difficult. Note that it is not a trivial task to find an attack with $S/N \ll 1$.

3.2.4 Truncated Differentials

The first description of the concept of truncated differentials was given by L.R. Knudsen in [60]. The most successful attack using truncated differentials is the attack on five rounds of SAFER [62, 75].

The naming 'truncated differential' is a bit confusing, since the concept applies to differential characteristics as well as to differentials. The basic idea however is very simple. Recall that a differential characteristic predicts the difference of all intermediate values in the encryption algorithm, while a differential predicts only the difference of a few intermediate values. The idea of a 'truncated characteristic' is to predict only a part of each intermediate value. For instance in the byte-oriented algorithm SAFER, a truncated characteristic typically only predicts which intermediate *bytes* are equal to zero, as opposed to a prediction of the individual *bits* in an ordinary differential characteristic. A truncated differential would logically be defined as a tuple that predicts only a few intermediate values, and then only partially. Since a truncated characteristic itself can be seen as a collection of ordinary characteristics, L.R. Knudsen prefers to make no distinction between a truncated characteristic and a truncated differential,

combining both concepts under the name 'truncated differential'.

The concept of truncated differentials seems particularly useful for attacking ciphers that do not operate on individual bits, but rather on higher level entities, e.g. the byte-oriented SAFER and the 16-bit word oriented IDEA [69]. Section 5.2.4 discusses a truncated differential attack on a reduced version of IDEA.

3.2.5 Higher Order Differentials

The derivative of a mapping s at the point a is defined as [59]:

$$\Delta_a s(x) = s(x * a)\Delta s(x)\,.$$

The connection with a differential of an encryption algorithm is clear. The definition can be extended to higher order derivatives [59]. The nth order derivative of the mapping s at the points $a_1, \ldots a_n$ is given by:

$$\Delta^{(n)}_{a_1,\ldots,a_n} s(x) = \Delta_{a_n}(\Delta^{(n-1)}_{a_1,\ldots,a_{n-1}} s(x))\,.$$

Higher order derivatives, or differences, can be used in a differential attack in the same way as first order derivatives. It seems that in practice higher order differences are most useful against ciphers with a small number of rounds [54]. Section 3.6 describes a second order differential attack on a reduced version of Blowfish.

3.3 Linear Cryptanalysis

Linear cryptanalysis was first described by M. Matsui in [77]. In [21] D. Coppersmith admits that to the best of his knowledge even the design team of the DES did not know about linear cryptanalysis.

Linear cryptanalysis is a known plaintext attack. Typically only a fraction of the bits of each plaintext have to be known. Also, if the cryptanalyst has only statistical information about the plaintext, he can apply a slightly modified version of the linear attack (this is in fact a ciphertext-only attack). As in differential cryptanalysis, the cryptanalyst can freely choose which operation to use as the 'linear' operation.

A linear attack uses a linear relation between the inputs and outputs of the encryption algorithm that holds with a certain probability. Linear relations for an algorithm are constructed by summing linear relations for the different components of the algorithm. Below, a linear attack is described where the bitwise exor is chosen as the linear operation and where the linear expression approximates the first $R-1$ rounds. In this case the linear relation contains

some bits of the input, some bits of the key and some bits of the input of the last round. Let

$$\alpha \bullet x^0 = \bigoplus_{i=1}^{l} \alpha_i x_i^0$$

denote the sum of plaintext bits that are selected by the l-bit vector α, and define with β and κ similar sums for the input of the last round x^{R-1} and the key k. The linear relation

$$(\alpha \bullet x^0) \oplus (\beta \bullet x^{R-1}) \oplus (\kappa \bullet k) = 0 \tag{3.27}$$

is called *effective* when it holds with a probability P that is not equal to 0.5. The *deviation* $d = |P - 0.5|$ is a measure for the effectiveness of the relation. Since the key is fixed, the relation

$$(\alpha \bullet x^0) \oplus (\beta \bullet x^{R-1}) = 0 \tag{3.28}$$

will hold with probability P or $1-P$, depending on the value of $\kappa \bullet k$. Therefore (3.28) will have the same deviation as (3.27).

The input of the last round can be expressed as a nonlinear function ρ^{-1} of the output x^R and the last round key k^R. In general the cryptanalyst chooses β in such a way that $\beta \bullet x^{R-1} = \beta \bullet \rho^{-1}[k^R](x^R)$ depends only on a small part of k^R. Substituting $\beta \bullet x^{R-1}$ in (3.28) gives

$$(\alpha \bullet x^0) \oplus (\beta \bullet \rho^{-1}[k^R](x^R)) = 0\,. \tag{3.29}$$

The attack now proceeds in the following way. For each possible value of k^R, the cryptanalyst calculates the deviation of (3.29) over a large set of encrypted texts. Because of the nonlinearity of $\rho^{-1}[k^R](x^R)$ the deviation of (3.29) will be largest when the guess for k^R is correct. This effect is called *'wrong key randomisation'* [45]. The number of required texts is proportional to d^{-2}.

Since the basic linear attack is a known-plaintext attack, there is an inherent equivalence between ciphertext and plaintext. A linear attack can be optimized by eliminating the first round linear relation from Relation (3.28). In this way only $R-2$ rounds are approximated, and parts of the subkeys of the first and the last round are searched for at the same time. This optimisation becomes impractical when too large a part of the key is involved, since a counter is required for every possible key value.

The *Linear Approximation Table* of a mapping forms the linear cryptanalysis equivalent of the difference table of differential cryptanalysis. If the exor operation is chosen as the linear operation, the LAT of a mapping can be calculated efficiently by using the fast Walsh transform. A comparison of (3.27)

with (3.10) shows that the deviation of a linear relation for a mapping is in fact equivalent to the correlation between a linear combination of the input bits and a linear combination of the output bits of the mapping, also called input-output correlation.

3.3.1 Differential-Linear Cryptanalysis

Differential-linear cryptanalysis was invented by S.K. Langford and M.E. Hellman [70]. A differential-linear attack is essentially a linear attack that is optimized by using chosen plaintexts. The attack uses an effective linear relation between the input of the last round and an intermediate value x^e, resulting in a relation similar to (3.29):

$$(\epsilon \bullet x^e) \oplus (\beta \bullet \rho^{-1}[k^R](x^R)) = 0\,. \tag{3.30}$$

The intermediate value x^e is determined with a certain probability by a differential characteristic.

For most ciphers it is easy to find differential characteristics with a high probability over a small number of rounds, whereas this probability decreases rapidly for an increasing number of rounds. Typically a differential-linear attack is very effective for ciphers with a small number of rounds, e.g. FEAL-8 succumbs to a differential-linear attack using as few as 12 chosen plaintexts [6] (but the attack has a high workload, cf. Appendix A).

3.4 Common Assumptions

In a differential attack the cryptanalyst needs to estimate the probability of a characteristic, in linear cryptanalysis an estimation for the deviation of a linear relation is required. These are necessary in order to estimate the probability of success, but also to select the best possible characteristic or relation, i.e. the one that gives the largest expected success probability for the attack.

To facilitate the estimation of the probability of a differential characteristic or a linear relation the following assumptions are often implicitly made.

1. One characteristic dominates the probability.

   In Section 3.2 it was explained that the probability of a differential is a better measure than the probability of a characteristic. However, often cryptanalysts assume that one characteristic has a much larger probability than the other characteristics of the differential. The probability of the characteristic is taken as an estimate of the probability of the differential. The analogous concept in linear cryptanalysis is less known: it is the

linear hull [59, 94]. The deviation of a linear hull is also estimated by the deviation of a linear relation.

2. The rounds are independent.

   The cryptanalyst estimates the probability of an r-round characteristic (or linear relation) by the product of the probabilities of the r component 1-round characteristics (or linear relations). This approach assumes that the probabilities of the different rounds are independent.

3. The hypothesis of stochastic equivalence [69].

   Since the key is unknown, the cryptanalyst often averages the probability over all keys and uses this as an estimate for the probability of the characteristic/relation. This is actually a first order approximation: the stochastic variable is estimated by its mean value.

These assumptions are necessary because it is clearly infeasible to calculate the correct probability of a differential or a linear relation for each possible key. Due to the inherent complexity of an encryption algorithm, this calculation would involve encrypting all plaintexts with all keys and storing all the corresponding ciphertexts.

A clear example that shows the limitations of two of these assumptions is given by a double encryption scheme. Let $D[k](x)$ denote the decryption operation that is the inverse of the encryption $E[k](x)$. The specific choice for the encryption algorithm is not important, provided that it exhibits no characteristics with probability one. A double encryption scheme can be constructed as $E2[k_1\|k_2](x) = D[k_2](E[k_1](x))$. The probability of the differentials of this scheme is clearly key dependent. If k_1 equals k_2, there exist differentials with probability one, and linear hulls with deviation 0.5. The probability of the differentials cannot be approximated by the probability of a characteristic. The same applies to the linear equivalents.

There are also more practical examples where the assumptions are violated. A common example is the existence of 'weak keys,' where some characteristics have higher probability than for other keys. In Chapter 5 it is shown that for certain keys the Message Authenticator Algorithm (MAA) exhibits many more collisions than the average case. In the same chapter it is shown that there are many key dependent differentials in IDEA. Also an attack is presented that makes use of a set of linear relations. While the average deviation of the relations is low, for each key there exists at least one relation with a high deviation.

Nevertheless, there are situations where the assumptions hold and a basic analysis breaks the cipher. Two examples are given in the next sections.

3.5 Cryptanalysis of MacGuffin

Under the assumptions of the previous section, the problem of finding the differential characteristic with the highest probability or the linear relation with the highest deviation can be translated into the search for the shortest path through a graph. It is then possible to apply the A^*-algorithm [91] to solve the problem. In [79] M. Matsui proposes a simple implementation of this algorithm. He uses it to find the best differential characteristic and the best linear relation of the DES. Algorithm 3.1 searches for the best differential characteristic over n rounds, using over-estimations for the probabilities of the best characteristics over $1, \ldots, n-1$ rounds, and an under-estimation for the probability of the best n-round characteristic. The more accurate the estimations are, the faster the search algorithm can prune wrong paths (corresponding to characteristics with a low probability).

It is possible to optimize the program by taking restrictions on the form of a characteristic into account [7]. The A^*-algorithm was implemented and used to cryptanalyse the block cipher MacGuffin [12]. These results on MacGuffin have been published in [110].

3.5.1 MacGuffin

B. Schneier and M. Blaze [12] introduce a new kind of round transformation: the *Generalized Unbalanced Feistel Network*. Together with the general architecture they give a complete specification of an example: the cipher MacGuffin. The basic idea is to split the input of each round into unequal parts. In MacGuffin, the 64-bit input is split into a 48-bit input of the round function, and a 16-bit part that is exored with the output of the round function. After four rounds all 64 bits have been exored once with the output of the round function. Figure 3.1 sketches four rounds of MacGuffin. The round function consists of the eight S-boxes of the DES, but the two middle output bits of each S-box are neglected in order to obtain a 16-bit output. Since the round function only modifies half as many bits of the intermediate value as in the case of the DES, the designers have chosen to use twice as many rounds: 32.

3.5.2 Differential Cryptanalysis

The analysis of MacGuffin [110] was done in an 'automatic' way by running Algorithm 3.1. Figure 3.1 shows the four-round iterative building block of the best differential characteristic for MacGuffin. It has a probability of $\frac{1}{149}$, which should be compared with $\frac{1}{234}$ for the best two-round iterative building block for a DES-characteristic. Table 3.1 gives the probabilities of the best differential characteristics of MacGuffin. The difference values are given in hexadecimal

Algorithm 3.1 A simple algorithm to search for the best differential characteristic. The values b_r are estimations for the probability of the best r-round characteristic: $b_1, \ldots b_{n-1}$ are over-estimations, b_n is an under-estimation. x_i, y_i denote the input and output difference of the round function of round i, $\Pr(x_i \mapsto y_i)$ denotes the transition probability.

```
round-1()
   for every x_1
      p_1 = max_{y_1} Pr(x_1 ↦ y_1);
      if (p_1 × b_{n-1} ≥ b_n) then round-2();
   return; /* exit program */

round-2()
   for every x_2, y_2
      p_2 = Pr(x_2 ↦ y_2);
      if (p_1 × p_2 × b_{n-2} ≥ b_n) then round(3);
   return;

round(r)  /* r = 3, ... , n-1 */
   x_r = x_{r-2} Δ y_{r-1};
   for every y_r
      p_r = Pr(x_r ↦ y_r);
      if (p_1 × p_2 × ... × p_r × b_{n-r} ≥ b_n) then
         if (r+1 < n) then round(r+1);
         else round-n();
   return;

round-n()
   x_n = x_{n-2} Δ y_{n-1};
   p_n = max_{y_n} Pr(x_n ↦ y_n);
   if (p_1 × p_2 × ... p_n ≥ b_n) then b_n = p_1 × p_2 × ... p_n;
   return;
```

notation, which is denoted by the subscript `x`. It turns out that the probability of the best $2n$-round characteristic of MacGuffin is significantly larger than the

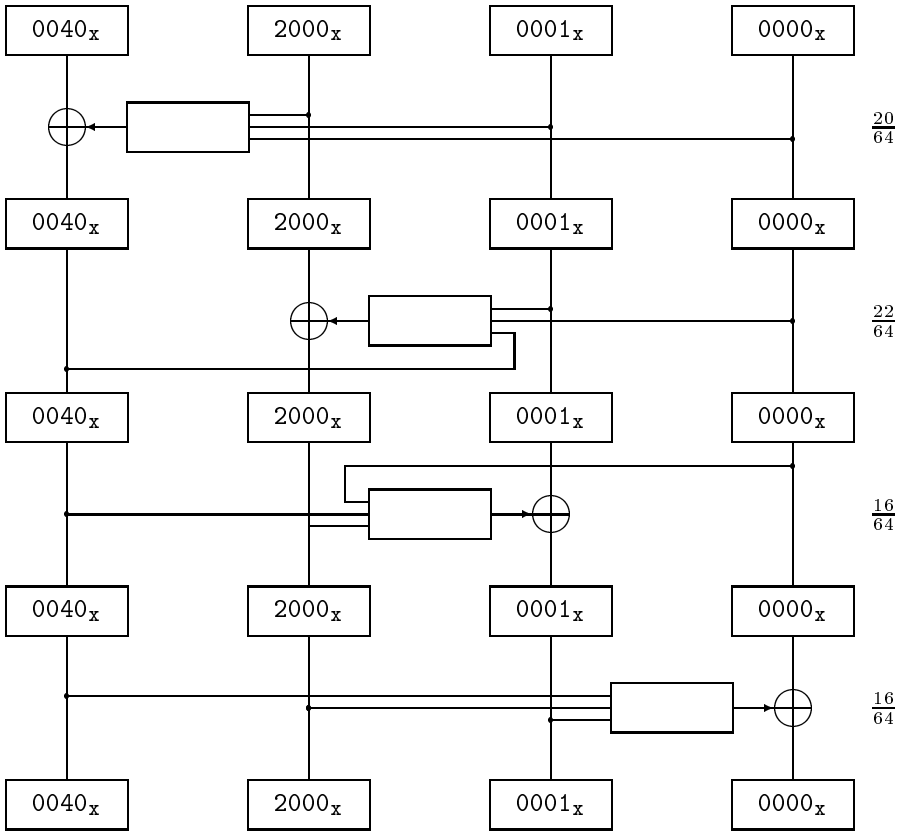


Figure 3.1: Four rounds of MacGuffin and the iterative building block of the best differential characteristic.

probability of the best n-round characteristic of the DES. From this viewpoint 32 rounds of MacGuffin is weaker than 16 rounds of the DES.

E. Biham and A. Shamir [10] used a 13-round characteristic for their attack on the full DES. The first round is passed with probability one by enciphering large structures of plaintexts. The last two rounds are treated by the 2R-attack.

An attack on MacGuffin can be mounted using a 27-round characteristic from the second to the 28th round. Extended to the first round, this characteristic has an input exor that is different from zero for S-box eight only. Since each S-box has only two output bits, only four different output exors are possible. Therefore the first round is passed with probability one by enciphering structures of only

MacGuffin		DES	
n	$\log_2(p)$	n	$\log_2(p)$
8	-11.6	4	-9.6
12	-19.4	6	-20.0
16	-27.7	8	-30.5
20	-34.8	10	-38.4
24	-42.7	12	-46.2
26	-46.4	13	-47.2
27	-47.9		
28	-49.9	14	-54.1
29	-51.6		
32	-57.2	16	-62.0

Table 3.1: The probabilities of the best characteristics for MacGuffin and the DES.

eight plaintexts (in comparison to 8192 for the DES). Such a structure consists of the messages $x \oplus (v, 0000_{\mathtt{x}}, 0000_{\mathtt{x}}, 0000_{\mathtt{x}})$, $x \oplus \alpha \oplus (v, 0000_{\mathtt{x}}, 0000_{\mathtt{x}}, 0000_{\mathtt{x}})$, where $\alpha = (4040_{\mathtt{x}}, 2000_{\mathtt{x}}, 0001_{\mathtt{x}}, 0000_{\mathtt{x}})$ and v takes the values from the set $\{0000_{\mathtt{x}}, 0001_{\mathtt{x}}, 0002_{\mathtt{x}}, 0003_{\mathtt{x}}\}$.

Because the diffusion per round of MacGuffin is weaker than that of the DES, it becomes possible to mount a 4R-attack. This attack gives the round key of the last round. Since the relation between master key and round keys is difficult to invert, the attack proceeds by peeling off the last round and by repeating the attack on the reduced version of MacGuffin. This step does not significantly add to the complexity of the attack.

Taking the DES S-boxes and reducing the output by chopping off the middle output bits, is a rather arbitrary design decision. It turns out that the resistance against differential cryptanalysis can be improved by selecting the first two output bits from the DES S-boxes and swapping them. The probability for the best 27-round characteristic becomes $2^{-50.8}$. Better still would be to design S-boxes specifically for MacGuffin.

3.5.3 Linear Cryptanalysis

A linear relation can be viewed as the tracing of bits through the different rounds of the algorithm. Every 'forked branch' is then a 'crossroads' where the cryptanalyst can choose which way to follow the bits. As a consequence of the imbalance of MacGuffin, there are 50 % more forked branches in each round than for the DES (48 'forked' bits instead of 32). On the other hand,

the reduction of the output of the S-boxes reduces the number of possible linear relations for each S-box by 80 % (only 3 possible output masks instead of 15). The effect of reducing the number of output bits on the expected value of the probability of the best linear relation is discussed by K. Nyberg in [95].

MacGuffin		DES	
n	$\log_2(\|p-0.5\|)$	n	$\log_2(\|p-0.5\|)$
4	-2.0	2	-1.7
8	-5.0	4	-4.0
12	-9.7	6	-8.0
16	-13.7	8	-10.7
20	-18.4	10	-14.4
24	-21.9	12	-16.8
28	-26.6	14	-20.8
30	-28.6	15	-21.8
32	-30.1	16	-23.4

Table 3.2: The probabilities of the best linear relations for MacGuffin and the DES.

The probabilities of $2n$-round relations for MacGuffin are lower than the probabilities of n-round DES-relations (cf. Table 3.2). A straightforward application of the linear attack, using an approximation for 30 rounds, would require $2^{2\times(28.6-20.8)} \times 2^{43} = 2^{58.6}$ plaintexts in order to find 12 bits of the round keys of the first and the last round. This is still faster than exhaustive key search. In order to determine the remaining part of these two round keys, other linear relations should be used. The structure of the best 30-round linear relation is shown in Figure 3.2.

structure: - - E - - - A - B C D - - - A - B C D - - - A - B C D - - -

	box (1-8)	α	β	$2\|p-0.5\|$
E	4	3_x	2_x	0.1250
A	4	38_x	1_x	0.3125
B	2	26_x	1_x	0.1875
C	6	15_x	1_x	0.1875
D	4	$2F_x$	1_x	0.3125

Figure 3.2: Structure of the optimal 30-round linear relation.

In Section 3.5.2 it was shown that MacGuffin could be strengthened against

differential cryptanalysis by selecting other output bits from the DES S-boxes. Selecting the first two output bits produced the cipher with the highest resistance to differential cryptanalysis. It has approximately the same resistance to linear cryptanalysis as the original version.

3.6 A Differential Attack on Blowfish

Blowfish is a block cipher designed by B. Schneier [121]. Vaudenay published a differential attack on Blowfish [131]. The attack assumes that the key dependent S-boxes are known to the cryptanalyst. It works on eight rounds for all keys and on sixteen rounds for some weak keys (a fraction of 2^{-17} of the keys are weak). The differential attack presented here [129] works on Blowfish reduced to four rounds, for all keys and with unknown S-boxes.

3.6.1 Blowfish

Blowfish operates on 64-bit plaintexts and has a variable key size (with a maximum of 448 key bits). It has a 16-round Feistel structure with some slight modifications: the round keys are added in a different place and there are two extra round keys at the end of the algorithm. The round transformation for the first 15 rounds is given by

$$\begin{aligned} t^r &= s^{r-1} \oplus k^{r-1} \\ s^r &= t^{r-1} \oplus F(t^r), \qquad r = 1, 2, \ldots, 15. \end{aligned}$$

In the last round this becomes:

$$\begin{aligned} s^{16} &= s^{15} \oplus k^{15} \oplus k^{17} \\ t^{16} &= t^{15} \oplus F(s^{15} \oplus k^{15}) \oplus k^{16}\,. \end{aligned}$$

The F-function uses four S-boxes with eight input bits and 32 output bits. The 32-bit input of the F-function is split into four bytes a_i, $i = 0, 1, 2, 3$.

$$F(a_0 a_1 a_2 a_3) = ((S_0[a_0] + S_1[a_1]) \oplus S_2[a_2]) + S_3[a_3]$$

Here '$\oplus$' stands for the exor operation and the '$+$' operation transforms the 32-bit values to integers and adds them modulo 2^{32}. The four S-boxes and the 18 k^r-values are obtained from the user key by a complex setup scheme. Figure 3.3 shows four rounds of Blowfish.

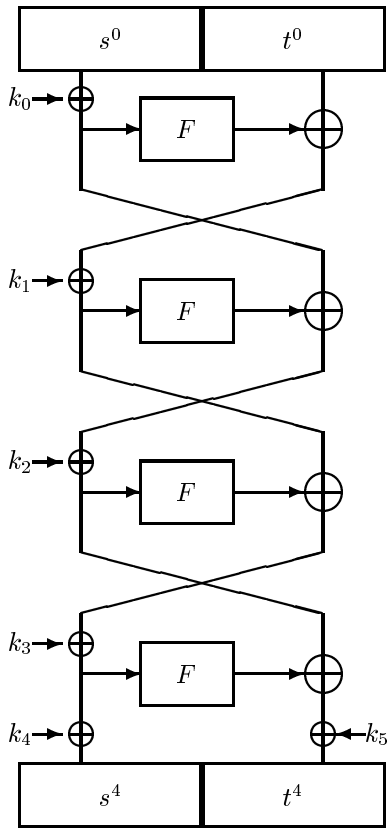


Figure 3.3: Four rounds of Blowfish.

Incompatible operations + and $\oplus$

The F-function of Blowfish uses two different operations to combine the outputs of the S-boxes: modular addition and bitwise exor. It is hoped that the incompatibility of these operations increases the security of the cipher.

However, modular addition and the exor operation are strongly related to each other. The same holds for their associated difference operations: modular subtraction and exor. Knowing $x \oplus y$, one can determine $x - y$ and $x + y$ with a certain probability, and vice versa. Values of $x - y$ correspond to a small set of possible values for $x \oplus y$. This is used in the attack: given the exor values $x^i \oplus y^i$ of a set of pairs (x^i, y^i) with a constant but unknown difference $x^i - y^i$, it is possible to determine this difference. Only a small number of exors is

required to determine the difference. Note that the sign of the difference cannot be determined, since exor commutes while subtraction does not.

For some values of x, special relationships hold. If $y \in \{0, 8000_{\mathtt{x}}\}$, $x + y = x - y = x \oplus y$. For other values of x, the exor and the subtraction have identical output for a large subset of the input space (e.g. $x = 4000_{\mathtt{x}}$).

Key equivalence classes

The Blowfish scheme can be simplified. For every set of S-boxes and k-values it is possible to determine an equivalent set of S-boxes and k-values, where k^0 equals 0, by 'pushing' the exor with k^0 through the algorithm. Table 3.3 shows how the new S-boxes and k-values relate to the old ones. This simplification permits a small speed-up of implementations since one exor operation can be omitted. The presented analysis assumes that this simplification has been carried out.

original	simplified
k^0	0
k^1	$k^1 \oplus k^0$
k^2	k^2
k^3	k^3
...	...
k^n	k^n
k^{n+1}	$k^{n+1} \oplus k^0$
k^{n+2}	$k^{n+2} \oplus k^0$
$F(x)$	$F(x \oplus k^0)$

Table 3.3: Simplifying the Blowfish scheme.

There are other key equivalences: the most significant bit of all entries in any two S-boxes can be flipped without changing the input-output mapping of the cipher. Any value can be added to all entries of S_0 and subtracted from all entries of S_1. Thus it is always possible to find an equivalent key with $S_0[0] = 0$.

These equivalences are between expanded keys. Due to the substantial expansion, only a small fraction of the keys from the expanded key space corresponds to actual cipher keys. It is very unlikely that there also exist equivalent cipher keys.

3.6.2 The Attack

Part One: exors of F-values

From the description of Blowfish [121] it follows that

$$s^4 = t^0 \oplus F(s^0) \oplus k^1 \oplus F(s^0 \oplus F(s^1 \oplus F(s^0) \oplus k^1) \oplus k^2) \oplus k^3 \oplus k^5 \,. \quad (3.31)$$

The attack uses a second order differential. It uses quartets of plaintexts, denoted $(a, b), (a, b^*), (a^*, b \oplus \delta), (a^*, b^* \oplus \delta)$. The second order difference of (3.31) is given by

$$\begin{aligned} s^4 \oplus s^{4^*} \oplus s^{4^{**}} \oplus s^{4^{***}} \quad = \quad & F(a \oplus F(b \oplus F(a) \oplus k^1) \oplus k^2) \\ \oplus \quad & F(a \oplus F(b^* \oplus F(a) \oplus k^1) \oplus k^2) \\ \oplus \quad & F(a^* \oplus F(b \oplus \delta \oplus F(a^*) \oplus k^1) \oplus k^2) \\ \oplus \quad & F(a^* \oplus F(b^* \oplus \delta \oplus F(a^*) \oplus k^1) \oplus k^2) \,. \end{aligned} \quad (3.32)$$

If $\delta = F(a) \oplus F(a^*)$, then (3.32) reduces to

$$\begin{aligned} s^4 \oplus s^{4^*} \oplus s^{4^{**}} \oplus s^{4^{***}} \quad = \quad & F(a \oplus x) \\ \oplus \quad & F(a \oplus y) \\ \oplus \quad & F(a^* \oplus x) \\ \oplus \quad & F(a^* \oplus y) \,, \end{aligned} \quad (3.33)$$

with $x = F(b \oplus F(a) \oplus k^1) \oplus k^2$ and $y = F(b^* \oplus F(a) \oplus k^1) \oplus k^2$. For random values of δ, (3.32) will be zero with a certain probability p; (3.33) will be zero with a larger probability because when $x = y$ it is always zero. This observation can be used to determine $F(a) \oplus F(a^*)$.

The attack works as follows. Firstly a set of plaintexts is encrypted with s^0 fixed to a. A second set is then encrypted with s^0 fixed to a^*. Quartets are built from these sets by taking two texts from each set. A quartet is valid if the sum of the t^0 values equals zero. The sum of the four s^4 values then equals the second order differential of (3.31). If this is zero, it gives two candidate values for $F(a) \oplus F(a^*)$: the first text from the first set can be combined with the first or the second text from the second set.

The candidate values are verified by experimentally verifying the probability that (3.32) is zero. For wrong values of $F(a) \oplus F(a^*)$ the probability will be much lower. This test was implemented for a reduced version of Blowfish, with only 32-bit block length. For the correct value the probability was 0.14%, for wrong values it was .01%.

Part Two: F-values

The technique of Part one allows values for the output exor of arbitrary input values of the round function F to be obtained. This technique is used with two inputs that differ only in the input of S_3 in the first round.

$$\begin{aligned} a &= a_0 a_1 a_2 a_3 \\ a^* &= a_0 a_1 a_2 a_3^* \end{aligned}$$

The difference of the two outputs is given by:

$$F(a) - F(a^*) = S_3[a_3] - S_3[a_3^*] \,.$$

Because of the similarity between exor and subtraction, this difference can be determined with a certain probability. By repeating the operation with different values for a_0, a_1, a_2 the difference can be determined uniquely, except for the sign bit.

In this way it is possible to collect enough information about exors and differences of the F-values to determine the absolute values, except for the sign bit. In the next step the absolute values of S_3 are determined. Subsequently S_2, S_1 and S_0 can be recovered, making use of the equivalence classes.

Once the F-function is known, the k^r-values can be easily recovered using standard techniques. Finally the most significant bits can be recovered by exhaustive search.

Plaintext requirements

On the reduced version of Blowfish with block length 32, about 8000 chosen plaintexts are required to determine one output exor value of the round function. The complete attack requires about 210 output exor values, or 2^{21} chosen plaintexts. The number of required plaintexts for Blowfish with block length 64 can be estimated as follows. The number of plaintexts to determine an output exor of the function will increase to approximately 2^{25}. The number of entries in the S-boxes increases, but each individual entry becomes easier to recover. The number of required exor values increases to 2^{13}. This gives a total of 2^{38} required chosen plaintexts.

3.7 Conclusions

This chapter started with the definition and basic properties of Boolean functions, linear functions, the Walsh transform and the other mathematical tools

that will be used in the following chapters. The principles of differential cryptanalysis were explained, and some extensions were discussed: the use of differentials instead of characteristics, truncated differentials and higher order differentials. The new concept of maximal differential was defined and it was shown that this allows the number of cases where a differential attack is applicable to be extended in a simple way. The principles of linear cryptanalysis and differential-linear cryptanalysis were explained. The underlying assumptions of both techniques were clearly stated and their limitations were shown.

Although these attacks have been known for some time, new designs have continued to be proposed that can be broken using these techniques. The block cipher MacGuffin has been broken by both linear and differential cryptanalysis. The differential characteristic and the linear relation were found by a search program that is guaranteed to find the best relations. This result has been published in [110]. The second order differential attack was illustrated with a new analysis of Blowfish, reduced to four rounds.

Chapter 4

Improved Differential Cryptanalysis

The ideas of differential and linear cryptanalysis can be extended in various ways. Several extensions have been described in the cryptographic literature. In [45] C. Harpes discusses a binary generalisation and a group generalisation of linear cryptanalysis. In [53] T. Jakobsen presents the generalisation of linear attacks to correlation attacks and discusses the links with differential attacks. While these extensions or generalisations are based on sound theoretical principles, few of them have been shown to improve an existing attack on any practical block cipher. The extensions to differential and linear cryptanalysis presented in this chapter were developed with applicability in mind and result in an improved attack on a practical block cipher (e.g., the DES).

This chapter is organized as follows: in Section 4.1 concepts of probability calculus are introduced into the framework of differential cryptanalysis. The new framework makes an improvement of differential cryptanalysis possible that extends the attack to cases where only a limited part of the cipher's output is visible. In Section 4.2 this extended attack is successfully applied to the DES in CFB-mode with a reduced number of rounds. The results on the DES have been published in [105].

In Section 4.3 a second extension introduces the idea of maximum likelihood. This technique is used to increase the number of recoverable key bits in an attack on the DES in CFB mode. It is also explained how to apply this technique to improve a linear attack.

The third extension is to block ciphers used as a hash function. Hash functions that are based on block ciphers can be analysed using a differential attack. For these attacks a special kind of differential characteristic is required.

Section 4.4 introduces a new approach to differential cryptanalysis of hash functions. These results have been published in [109].

4.1 Probability Calculus and Differential Analysis

This section introduces some elements of probability calculus. It starts with the definition of probability space that will be used, because this definition deviates from that which might be expected. Afterwards the operation of the standard differential attack is described and the extensions are shown.

4.1.1 Definition of Probability

Strictly speaking, the knowledge of one or two plaintexts and their corresponding ciphertexts is in most cases sufficient to determine the used encryption key uniquely. The required number of plaintexts is known as the *unicity distance* [126] of the cipher. For a block cipher that operates on blocks of l bits and uses p-bit keys, on average 2^{p-nl} keys are expected that map n given plaintexts to n given ciphertexts. For most practical ciphers $p \approx l$ and the unicity distance is approximately one.

Under these conditions it seems a bit strange to talk about the probability distribution of a key. Even if only a few bits of the output are visible, and l is reduced accordingly, a few known plaintexts suffice to bring the expected number of possible keys down to one. However, the equations that describe the key as a function of the known plaintext and ciphertext bits are highly nonlinear and, currently, the only known way to solve them is by exhaustive search of the key space, resulting in a very high work factor for the cryptanalyst.

In a differential attack, as in many other attacks, there is a trade-off between work and plaintext requirements. By using only statistical information from the plaintexts and ciphertexts, the cryptanalyst is able to reduce the work factor significantly. It is in this statistical context that the probability distribution of the key is defined.

Let an *experiment* be defined as the encryption of one pair of plaintexts (p, p^*) under the unknown key. The *result* of the experiment consists of the partial information about the pair of plaintexts and the corresponding pair of ciphertexts that will be used in the attack. If the ciphertext is only partially visible, then this is a first restriction on the information that is included in the result of the experiment. Further restrictions follow from the specific data processing stages of the attack. Let X be the key dependent function that takes as input the number of the experiment and produces as output the result of the experiment. x denotes a particular outcome of the experiment. If the key

is not known, X can be considered as a stochastic variable. Often an attack recovers only part of the unknown key, which we denote by K. The probability distribution for K conditional on the result of an experiment is defined as:

$$\Pr(K \mid X) = \frac{\Pr(K, X)}{\Pr(X)}. \tag{4.1}$$

As explained in Section 3.2, a differential attack is based on a differential characteristic. Let $G(X)$ denote the event that the encrypted pair follows the characteristic: $G(X) = 1$ for a good pair and $G(X) = 0$ for a wrong pair. By Bayes' rule

$$\Pr(K \mid X, G(X) = 1) = \frac{\Pr(X \mid K, G(X) = 1) \cdot \Pr(K \mid G(X) = 1)}{\Pr(X \mid G(X) = 1)} \tag{4.2}$$

In what follows, this probabilistic model is used to explain the underlying principles of a differential attack and to illustrate the simplifications that are made in a standard differential attack. It is then shown how the removal of some of the simplifications leads to an improvement of the attack.

4.1.2 The Basic Differential Attack Revisited

A differential attack consists of performing some experiments and deciding which key value is the most probable, based on the results of the performed experiments. Standard differential cryptanalysis uses the concept of suggested key values. A key value is suggested if, under the assumption that the encrypted pair follows the characteristic, the probability of the key producing the experimental result is non-zero.

$$\Pr(K = k \mid X = x, G(x) = 1) > 0 \Leftrightarrow \Pr(X = x \mid K = k, G(x) = 1) > 0$$

The conditional probability $\Pr(X = x \mid K = k, G(x) = 1)$ can be calculated from the extended difference table.

Definition 4.1 *Let s be a mapping from G^n to G^m. The* extended difference table *is then given by:*

$$ee_s : G^n \times G^n \times G^m \to G :$$

$$ee_s(i, i^*, u) = \begin{cases} 1 & \textit{if } s(i)\Delta s(i^*) = u \\ 0 & \textit{else} \end{cases} \tag{4.3}$$

The extended difference table of a mapping $s(\cdot)$ is equivalent to the characteristic function of $s(i)\Delta s(i^*)$, and can be visualised as a table that results from the splitting up of the rows of the difference table:

$$\sum_{j \in G^n} ee_s(j, j\Delta i, u) = e_s(i, u) \tag{4.4}$$

The set of table entries with the same value of i and i^* is called a row, the set of entries with a common u-value is called a column.

The relation between the extended difference table and the conditional probability distribution of the key is now illustrated for the case of a 1R-attack on a Feistel network.

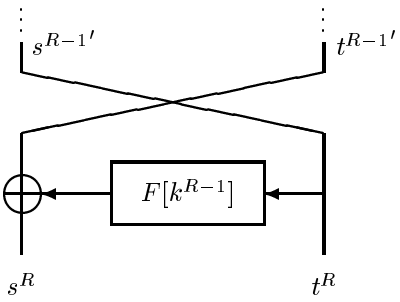


Figure 4.1: The last round of a Feistel Network. The values $s^R, t^R, s^{R^*}, t^{R^*}, t^{R-1}$ and $t^{(R-1)^*}$ are visible in the ciphertext; $s^{R-1\prime}$ is predicted by the characteristic of the 1R-attack.

Figure 4.1 shows the situation. The used differential predicts the difference at the input of the last round. The output difference of the round function F in the last round can be calculated from the ciphertext difference and the predicted input difference of the last round. The inputs of the round function F of the last round are visible in the ciphertext. The input values are denoted by i and i^*, the difference is denoted by $i' = i\Delta i^*$, the output value difference is u'. The key addition operation is denoted by '+'. The following then holds:

$$\Pr(x \mid k, G(x) = 1) \sim ee_F(i + k, i^* + k, u') \tag{4.5}$$

If $ee_F(i+k, i^*+k, u') > 0$, then by (4.2) it follows that $\Pr(k \mid x, G(x) = 1) > 0$, and the key value k is suggested. If $ee_F(i + k, i^* + k, u') = 0$, then k is not suggested.

If the pair does not suggest any keys, it is filtered. After encrypting a certain number of plaintext pairs, the most suggested key is likely to be the encryption key actually used. Since the attack only distinguishes between suggested key values and not suggested key values, it works only if not too many key values are suggested by each experiment. For the extended difference table, this means that in general the attack works if there are many zero entries. If $\Pr(k \mid x, G(x) = 1) > 0$, then for all values of k the experimental result x suggests all key values. If this is true for all possible results, then the basic differential attack fails, because all key values will be suggested by all pairs. It follows from Definition 4.1 and (4.5) that this situation cannot occur if the complete input value and output difference of the mapping are visible.

This situation changes when only a part of the input value or the output difference is visible. Entries that differ only in invisible parts of the input or the output have to be joined. Thus a partly visible output results in the joining of columns, and a partly visible input results in the joining of rows. Input or output values that differ only in invisible parts will be called indistinguishable values. Example 4.1 illustrates that this reduction can make the basic differential attack impossible. In this case there is a definite advantage in using the more sophisticated attack that is presented below.

Example 4.1 *Consider the mapping* $x \mapsto x^3 + 1 \bmod 4$ *in* $\mathbb{Z}$. *Table 4.1 shows the extended difference table and its reduced form.*

4.1.3 The Extended Attack

It is possible to get more information about the conditional probability $\Pr(K \mid X, G(X) = 1)$ than the simple zero versus non zero distinction that is made in basic differential cryptanalysis. The extended differential attack does this.

Assuming that the absolute values of the ciphertext output are almost uniformly distributed, the probability distribution of the key conditional on the result of one experiment is easily expressed as a function of the entries of the extended difference table:

$$\begin{aligned} &\Pr(K = k \mid i, i^*, u', G(x) = 1) \\ &\quad = \frac{\Pr(K = k \mid G(x) = 1) \cdot \Pr(i, i^*, u' \mid K = k, G(x) = 1)}{\Pr(i, i^*, u' \mid G(x) = 1)}. \end{aligned} \tag{4.6}$$

The factors on the right hand side can be calculated. Assuming that the characteristic is key independent, $\Pr(K = k \mid G(x) = 1)$ equals $\Pr(K = k)$. The

inputs		output difference			
i	i^*	00	01	10	11
00	00	1	0	0	0
00	01	0	0	0	1
00	10	1	0	0	0
00	11	0	1	0	1
01	00	0	1	0	0
01	01	1	0	0	0
01	10	0	1	0	0
01	11	0	0	1	0
10	00	1	0	0	0
10	01	0	0	0	1
10	10	1	0	0	0
10	11	0	1	0	0
11	00	0	0	0	1
11	01	0	0	1	0
11	10	0	0	0	1
11	11	1	0	0	0

		0x	1x
0x	0x	3	1
0x	1x	3	1
1x	0x	1	3
1x	1x	3	1

Table 4.1: The extended difference table of the example mapping (left), and the reduced version if only the most significant bit of input and output are visible (right). Variable x denotes an invisible bit. The difference is subtraction in $\mathbb{Z}$ modulo four.

uniform distribution of the absolute values implies:

$$\begin{aligned}
&\Pr(i, i^*, u' \mid K = k, G(x) = 1) \\
&\quad = \Pr(i \mid K = k, G(x) = 1) \Pr(i^* \mid K = k, G(x) = 1) \\
&\qquad \cdot \Pr(u' \mid i, i^*, K = k, G(x) = 1) \\
&\quad = \Pr(i) \Pr(i^*) \frac{ee_F(i + k, i^* + k, u')}{\sum_{v'} ee_F(i + k, i^* + k, v')} .
\end{aligned}$$

Here $\sum_{v'}$ means the sum over all distinguishable output differences, and likewise in the following, $\sum_l$ means the sum over all distinguishable input values.

$$\begin{aligned}
&\Pr(i, i^*, u' \mid G(x) = 1) \\
&\quad = \Pr(i \mid G(x) = 1) \Pr(i^* \mid G(x) = 1) \Pr(u' \mid i, i^*, G(x) = 1) \\
&\quad = \Pr(i) \Pr(i^*) \frac{\sum_l ee_F(i + l, i^* + l, u')}{\sum_l \sum_{v'} ee_F(i + l, i^* + l, v')} . \qquad (4.7)
\end{aligned}$$

If the difference operation corresponds to the key addition operation, (4.7) is

equivalent to:

$$\Pr(i, i^*, u', \mid G(x) = 1) = \Pr(i)\Pr(i^*)\frac{e_F(i\Delta i^*, u')}{\sum_{v'} e_F(i\Delta i^*, v')}. \tag{4.8}$$

Combining (4.6) and (4.8) gives

$$\begin{aligned}&\Pr(K = k \mid i, i^*, u', G(x) = 1)\\ &\quad = \Pr(K = k)\frac{ee_F(i + k, i^* + k, u')\sum_{v'} e_F(i\Delta i^*, v')}{e_F(i\Delta i^*, u')\sum_{v'} ee_F(i + k, i^* + k, v')}.\end{aligned} \tag{4.9}$$

Table 4.2 illustrates this for the example mapping.

inputs		output difference	
i	i^*	$0x$	$1x$
$0x$	$0x$	$1/2$	$1/2$
$0x$	$1x$	$3/4$	$1/4$
$1x$	$0x$	$1/4$	$3/4$
$1x$	$1x$	$1/2$	$1/2$

Table 4.2: $\Pr(K = 0 \mid i, i^*, u', G(x) = 1)$ calculated for the example mapping, assuming that $\Pr(K = 0)$ before the experiment equals $1/2$.

Equation (4.9) gives the conditional probability distribution after one good pair. Denoting the probability of the characteristic with p, the probability distribution after one pair becomes:

$$\begin{aligned}\Pr(K \mid X) &= \frac{\Pr(K)\Pr(X \mid K)}{\Pr(X)}\\ &= \Pr(K)\frac{\Pr(X \mid K, G(X) = 1)p + \Pr(X \mid K, G(X) = 0)(1 - p)}{\Pr(X \mid G(X) = 1)p + \Pr(X \mid G(X) = 0)(1 - p)}.\end{aligned} \tag{4.10}$$

In an extended differential attack, the probability distributions resulting from the separate experiments are combined. If sufficiently many experiments are combined then it is possible to determine the correct value of the key with high probability. The following propositions show how the results of different experiments are to be combined.

Proposition 4.1 *Let $q^1(K)$ and $q^2(K)$ denote the key probability distributions resulting from two experiments x^1, x^2. Let the a priori probability distribution*

of the key be uniform. The combined probability distribution $Q(K)$ is then given by:

$$Q(k) = q^1(k) \diamond q^2(k),$$

where the operation '$\diamond$' is defined by

$$q^1(k) \diamond q^2(k) \equiv \frac{q^1(k)q^2(k)}{\sum_l q^1(l)q^2(l)}.$$

To prove this proposition the following lemma is used several times.

Lemma 4.2 *The conditional distribution of the results of an experiment, conditional on the key, does not depend on the results of other experiments.*

$$\Pr(x^i \mid x^j, K) = \Pr(x^i \mid K) \tag{4.11}$$

No formal proof is given for the lemma because it is intuitively clear: if the key is known, then the results of the experiments are determined. Now Proposition 4.1 can be proven.

Proof: By definition,

$$Q(k) = \Pr(K = k \mid x^1, x^2).$$

With Bayes' rule this becomes

$$\begin{aligned} Q(k) &= \frac{\Pr(K = k) \cdot \Pr(x^1, x^2 \mid K = k)}{\Pr(x^1, x^2)} \\ &= \frac{\Pr(K = k) \cdot \Pr(x^1 \mid K = k) \cdot \Pr(x^2 \mid x^1, K = k)}{\Pr(x^1) \cdot \Pr(x^2 \mid x^1)}. \end{aligned}$$

Using (4.11) and the definition of q_1 this becomes

$$\begin{aligned} Q(k) &= q^1(k) \frac{\Pr(x^2 \mid K = k)}{\Pr(x^2 \mid x^1)} \\ &= \frac{q^1(k) \cdot \Pr(x^2 \mid K = k)}{\sum_l \Pr(x^2 \mid x^1, K = l) \cdot \Pr(K = l \mid x^1)} \\ &= \frac{q^1(k) \cdot \Pr(x^2 \mid K = k)}{\sum_l \Pr(x^2 \mid K = l) \cdot q^1(l)}. \end{aligned}$$

In these equations $\sum_l$ means the sum over all possible key values. Since the a priori probability distribution of the key is uniform,

$$\Pr(K = k) = \Pr(K = l), \forall l.$$

The combined probability distribution can now be expressed as:

$$\begin{aligned} Q(k) &= \frac{q^1(k) \cdot \Pr(x^2 \mid K = k) \Pr(K = k) / \Pr(x^2)}{\sum_l q^1(l) \Pr(x^2 \mid K = l) \Pr(K = l) / \Pr(x^2)} \\ &= \frac{q^1(k) \cdot q^2(k)}{\sum_l q^1(l) \cdot q^2(l)} . \end{aligned}$$

■

Let $\delta(k)$ denote the Kronecker delta function: $\delta(0) = 1$ and $\delta(k) = 0, \forall k \neq 0$. Let the uniform distribution be denoted by $\nu(k)$. A straightforward calculation shows that for any distributions q, r, s:

$$q(k) \diamond \nu(k) = q(k) \tag{4.12}$$

$$\delta(k \Delta k^0) \diamond q(k) = \delta(k \Delta k^0) \tag{4.13}$$

$$(q(k) \diamond r(k)) \diamond s(k)) = a(k) \diamond (r(k) \diamond s(k)) = \frac{q(k) r(k) s(k)}{\sum_l q(l) r(l) s(l)} \tag{4.14}$$

The first equation reveals that if an experiment gives no information about the key, then the conditional probability distribution remains unchanged. If the probability distribution equals $\delta(k \Delta k^0)$, then the correct key value is known to be k^0. The second equation states that the results of any additional experiment will not change this. The third equation is the associativity law.

These properties can be used to prove the following proposition:

Proposition 4.3 *The combined outcome from pairs* $1, 2, \ldots, j$ *can be calculated from* $q^1(k), q^2(k), \ldots q^j(k)$ *in the following recursive way:*

$$Q^j(k) = \frac{q^j(k) \cdot Q^{j-1}(k)}{\sum_l q^j(l) \cdot Q^{j-1}(l)} \qquad \textit{for } j = 1, 2, \ldots, M$$

and $Q^0(k) = \nu(k)$.

In an extended differential attack, pairs will be encrypted and the results will be processed with Formulae (4.9) and (4.3) until $Q^M(k)$ allows the value of k to be predicted with high probability. An estimate of the value of M that makes $Q^M(k)$ close to $\delta(k \Delta k^0)$ will be developed in the next section.

4.2 Cryptanalysis of the DES in the CFB mode

The extended differential attack can be applied to the DES in the CFB mode [105]. As the key addition in the DES is an exor, it is natural to use exor as the difference operation Δ. In the m-bit CFB mode, only m bits of the 64-bit

output of the DES are visible. As a consequence, only partial information about the input and the output of the last round is available, as indicated in Table 4.3. It is clear that a differential attack requires that information on both input and output bits of a single S-box is available. This means that in 1-bit CFB this approach is restricted to the trivial case of two rounds. If three rounds or more are used, then it follows from Table 4.3 that m has to be at least three. In the following, the differential attack will be described for 8-bit CFB. In this case most information is available on S-box three, namely one input bit and two output bits.

S-box	known inputs	accessible outputs
1	$a = 7$	
2	$e = 1$	
3	$a = 1$	$\alpha = 4$, $\beta = 6$
4	$e = 3$	
5	$a = 3$	$\alpha = 2$
6	$e = 5$	
7	$a = 5$	$\alpha = 8$
8	$e = 7$	

Table 4.3: Input bits of S-boxes that are known and output bits that are accessible in the case of m-bit CFB ($m \leq 8$); the six inputs bits of an S-box are denoted by a to f, the four outputs are denoted by α to δ, and the CFB bits are denoted by the digits 1 to 8.

4.2.1 Mode Specific Problems

Because only a limited subset of the input bits and output bits of the last round is visible, the exor-table has to be reduced. However, for the 8-bit CFB mode all values in the reduced exor-table are equal. Furthermore, when X and K are restricted to the visible bits and the key bits that are added with visible input bits, $\Pr(K \mid X)$ becomes uniform for all values of X, meaning that no information about the key can be extracted.

Part of the problem can be solved by using the differential characteristic to predict the input exor of the last round. This technique avoids the collapsing of rows in the exor-table. The characteristic predicts only differences, not absolute values. Since the key addition is linear, the input difference of the S-boxes is completely determined by the input difference of the round; it is independent of the value of the key bits that are added in the last round. Therefore it is only possible to recover key bits that are added to bits that are visible in the

output. Only for these bits can the absolute value of the corresponding bits at the input of the S-boxes be expressed as a function of the unknown key bits and the known output bits. For the case of the DES in 8-bit CFB, this means that only three key bits can be recovered: one that enters S3 in the last round, one that enters S5 and one that enters S7.

Input exor prediction makes a differential attack possible, because now for some values of X not all values of K are possible. This effect is visible in the reduced form of the extended exor-table: some zero entries appear (cf. Table 4.4). However, for the most probable experimental results x, the entries are non zero for all key values k (because only one input bit is visible, there are only two k-values). This means that using the conventional approach, most pairs that pass filtering will suggest both key values, resulting in a dramatic increase in the chosen plaintext requirements.

Note that Proposition 3.2 does not apply here. Only a part of the output of the cipher is visible, and a part of the invisible output is predicted and actually used to extract information about the key. Therefore, there will be pairs that do not have the predicted differences ('wrong pairs') but that do suggest the correct value for the key.

Table 4.4 gives a part of the reduced form of the extended exor-table of $S3$. We denote by i and i^*, the bits that enter $S3$ in the last round in position a. Bit 7 of the visible output equals $i \oplus k$, resp. $i^* \oplus k$. Only one input bit is visible, but the rest of the input difference can be predicted by the characteristic. When bit a of the input exor is zero, the two rows of the extended exor-table have different entries, only in this case, the conditional probability distribution is non-uniform. The fact that the two rows of the extended exor-table are equal when bit $a = 1$ can be easily understood. For every input pair (p, p^*) with $(i, i^*) = (0, 1)$, there is a corresponding pair (p^*, p) with $(i^*, i) = (1, 0)$. Since the exor operation commutes, the output exors of both pairs are the same. In the remaining analysis only input exors with the bit a equal to zero will be considered.

Table 4.4 contains all the information needed to calculate $\Pr(K \mid X, G(X) = 1)$ using (4.9). Before (4.10) can be used to calculate $\Pr(K \mid X)$, a remark should be made about the characteristic's probability. Both the input exor and the output exor are only known with a certain probability. These probabilities are close to one another, but are generally not equal; which of the two is best known depends on the filtering. However, to simplify the analysis it will be assumed that both exors are known with the same probability. This turns out to be a good approximation.

The extended differential attack can be applied to determine k. If $Q^M(0) > 0.5$, we decide that $k = 0$. In practice it is expected that, after a sufficient number of experiments, $1 - Q^M(0)$ will be a reliable estimate for k and will be close to 1 (or 0) with high probability. An important question is how the

input exor $S3'_E$	output exor $S3'_O$				quality H
(i, i^*)	$00xx_x$	$01xx_x$	$10xx_x$	$11xx_x$	Eq. (4.17)
01_x	2	18	18	26	0.719
$(0,0)$	2	14	10	6	
$(1,1)$	0	4	8	20	
02_x	2	10	26	26	0.688
$(0,0)$	0	8	16	8	
$(1,1)$	2	2	10	18	
04_x	4	8	24	28	0.688
$(0,0)$	4	8	8	12	
$(1,1)$	0	0	16	16	
10_x	4	24	12	24	0.688
$(0,0)$	0	8	8	16	
$(1,1)$	4	16	4	8	
20_x	0	16	8	40	0.5
$(0,1)$	0	8	4	20	
$(1,0)$	0	8	4	20	
24_x	26	22	10	6	0.5
$(0,1)$	13	11	5	3	
$(1,0)$	13	11	5	3	

Table 4.4: This table was formed by joining parts of the reduced forms of the exor-table and of the extended exor-table for $S3$. The rows are rearranged such that rows with a common input exor are near to one another. The only visible input bit is a, but the complete input exor is predicted by the characteristic. Variable $\mathtt{x}$ denotes an invisible bit, and the subscript $_\mathtt{x}$ indicates hexadecimal notation.

required number of pairs depends on the required error probability.

Denote by $N(z)$ the required number of right pairs to predict a key bit with an error probability equal to z. Let

$$q[i, u'] = \Pr(K = 0 \mid i, i^*, u')\,, \tag{4.15}$$

where i^* is left omitted from the notation because for a good pair it is determined by i and the characteristic. An estimation of $N(z)$ is given by $N'(z)$, where $N'(z)$ is defined by

$$N'(z) = 64 \cdot \frac{\ln\left(\frac{1}{z} - 1\right)}{\ln(\tau)}, \qquad \text{with } \tau = \prod_{v=0}^{3} \left(\frac{1}{q[0, v]} - 1\right)^{ee_F(0,0,v) - ee_F(1,1,v)}, \tag{4.16}$$

where v indexes the four columns of the extended exor-table.

This formula is motivated by the following. Suppose $k = 0$, and consider 64 right pairs distributed according to the entries in the extended exor-table. Then

$$\begin{aligned} Q^{64}(0) &= \frac{\prod q^i(0)}{\prod q^i(0) + \prod(1 - q^i(0))} \\ &= \frac{1}{1 + \prod(\frac{1}{q^i(0)} - 1)} \\ &= \frac{1}{1 + (\frac{1}{q[0,0]} - 1)^{ee_F(0,0,0)} \cdot \ldots \cdot (\frac{1}{q[1,3]} - 1)^{ee_F(1,1,3)}} \,. \end{aligned}$$

From (4.15) it follows that $q[0, v] + q[1, v] = 1$, therefore

$$\frac{1}{q[1,v]} - 1 = \frac{1}{1 - q[0,v]} - 1 = (\frac{1}{q[0,v]} - 1)^{-1} \,.$$

This gives

$$\begin{aligned} Q^{64}(0) &= \frac{1}{1 + (\frac{1}{q[0,0]} - 1)^{ee_F(0,0,0) - ee_F(1,1,0)} \cdot \ldots \cdot (\frac{1}{q[0,3]} - 1)^{ee_F(0,0,3) - ee_F(1,1,3)}} \\ &= \frac{1}{1 + \tau} \,. \end{aligned}$$

For an arbitrary number of right pairs n this becomes

$$Q^n(0) = \frac{1}{1 + \tau^{n/64}} \,.$$

To predict $k = 0$ with an error probability z, it is required that $Q^n(0) \geq 1 - z$. Equation (4.16) defines $N'(z)$ such that $Q^{N'(z)}(0) = 1 - z$.

If the wrong pairs are randomly distributed then their influences on Q cancel one another out. This follows from the fact that $q[0, v] = 1 - q[1, v]$ (the key bit is either 0 or 1). In this case

$$q[0,v] \diamond q[1,v] = \frac{q[0,v](1 - q[0,v])}{q[0,v](1 - q[0,v]) + (1 - q[0,v])q[0,v]} = \frac{1}{2} \,.$$

Thus it is expected that wrong pairs will produce the uniform distribution for $Q(0)$. The total number of required pairs $M(z)$ can then be estimated by $p^{-1}N'(z)$.

Note that (4.16) can be extended to the case where certain output exors are filtered: one can simply modify the corresponding table entries such that $ee_s(0, 0, u') = ee_s(1, 1, u')$, yielding $q[0, u'] = q[1, u'] = 0.5$.

The optimisation of this attack, or equivalently the minimisation of M is not easy, since it also depends on the properties of the characteristic: the probability p, the possibility of filtering and the non-uniformity in the extended exor-table. A good heuristic measure for the entries in the extended exor-table for a given input exor i is the expression

$$H = \frac{\sum_{v'=0}^{3} \max(ee_F(0,0,v'), ee_F(1,1,v'))}{\sum_{v'=0}^{3} ee_F(0,0,v') + ee_F(1,1,v')} . \tag{4.17}$$

This measure is indicated in Table 4.4. $H = 0.5$ means that the extended attack is not applicable.

4.2.2 An Attack on 4 Rounds

It follows from the previous section that the optimal input exor of S3 in the last round is $01_{\mathtt{x}}$. The characteristic has to predict only the input exor of S3 in the last round and the exor of two bits that are added to the output of S3 in the last round. This means that a truncated characteristic can be used, which has a higher probability. Figure 4.2 shows a differential characteristic with high probability that produces the required input exor in the last round. The input exor to the first round is equal to $(40\,08\,00\,00_{\mathtt{x}}, 04\,00\,00\,00_{\mathtt{x}})$. The characteristic has then a probability of $1/4$ in the first round. In the third round, it is sufficient that the input exor to $S3$ is correct. This gives a probability of $3/4$ for the truncated characteristic in the third round. A close inspection reveals that all pairs that go wrong in the third round result in an input exor of zero for S3 in the last round. If the pairs with output exor $00\mathtt{xx}_{\mathtt{x}}$ for $S3$ are filtered, these wrong pairs can be filtered out. Of the pairs that go wrong in the first round, approximately one quarter will produce an output exor $00\mathtt{xx}_{\mathtt{x}}$. Only a fraction of $\frac{2}{64}$ of the right pairs are lost. Further filtering of the wrong pairs can be done on the visible input bit of S3. From this it follows that the fraction of right pairs after filtering is equal to

$$\tilde{p} = \frac{p_1 p_3 \frac{62}{64}}{p_1 p_3 \frac{62}{64} + (1-p_1) \cdot \frac{3}{4} \cdot \frac{1}{2}} = 0.392 .$$

The following equation for $q_{0,v}$ the follows from (4.15) and (4.10):

$$q_{0,v} = \frac{1}{2} \cdot \frac{\tilde{p} \cdot \frac{ee_F(0,0,v)}{30} + (1-\tilde{p}) \cdot \frac{1}{3}}{\tilde{p} \cdot \frac{e_F(0,v)}{62} + (1-\tilde{p}) \cdot \frac{1}{3}} . \tag{4.18}$$

This assumes that the wrong pairs yield a uniform distribution of output exors, a fact which has been confirmed by computer experiments. This gives $q_{0,1} = 0.609$, $q_{0,2} = 0.527$, and $q_{0,3} = 0.383$.

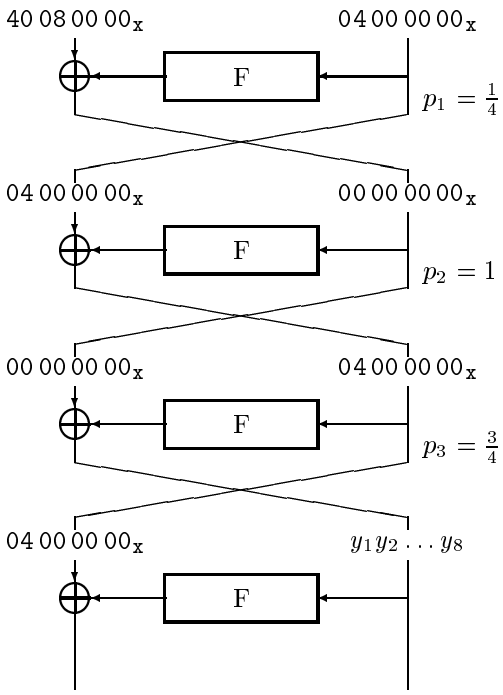


Figure 4.2: The optimal characteristic for an attack on 4 rounds.

For an error probability of 5% (or $z = 0.95$), (4.16) gives $M' = 16.6$ and M is estimated to be $M'/p = 89$, which was confirmed by computer simulations. A similar attack can be applied to determine a key bit entering S-boxes 5 and 7; it will only be discussed for a larger number of rounds.

If $m = 8$, a differential attack allows three key bits to be determined (namely one bit corresponding to $S3$, $S5$ and $S7$), and if $m = 16$ this increases to six key bits. Even in that case, the maximal number of output bits of a single S-box is equal to two.

4.2.3 5 Rounds and More

In order to develop an attack that is extendible to more rounds, an iterative characteristic will be used. Detailed calculation shows that the best result is obtained with the iterative characteristic ϕ ($s^{0'} = \mathtt{19\,60\,00\,00_x}, t^{0'} = \mathtt{0_x}$) [10]. For five rounds, the fraction of right pairs after filtering becomes $9.40 \cdot 10^{-3}$. For

an error probability of 5% (or $z = 0.95$), (4.16) predicts that 370,000 pairs are sufficient to obtain a key bit (only 1 characteristic has been used). Computer simulations show that the actual number of pairs is even smaller.

For S-boxes 6 and 7, a similar strategy can be followed. The best iterative characteristic for both S-boxes has input exor $s^{0\prime} = 00\,00\,1D\,40_{\mathtt{x}}, t^{0\prime} = 0$. The fraction of right pairs after filtering is equal to $5.92 \cdot 10^{-3}$ and $3.33 \cdot 10^{-3}$, from which it can be estimated that the number of required pairs is equal to 12 and 8.4 million.

To attack six rounds, the first round trick [10] can be used. The estimated number of pairs to find 1 and 3 key bits for 8-bit CFB are indicated in Table 4.5.

The attack for $R = 7$ (without the first round trick) was implemented as a distributed application on a heterogeneous, non-dedicated farm of 30 DEC workstations, using the PVM (Parallel Virtual Machine) software [42] for inter-process communication. The program was generated and run from the HeNCE (Heterogeneous Network Computer Environment) software [9]. The correct key bits were retrieved from $2^{35.2}$ pairs using a quartet structure; the attack took about 40 hours. Table 4.5 gives the required number of pairs for the DES reduced to six, eight or ten rounds, and for the full DES. Since there are only 2^{63} different pairs, the maximum number of rounds that can be broken, in theory, is ten.

# rounds	probability p		# pairs	
	1 bit	3 bits	1 bit	3 bits
6	$9.40 \cdot 10^{-3}$	$3.33 \cdot 10^{-3}$	$2^{18.5}$	$2^{23.0}$
8	$4.05 \cdot 10^{-5}$	$1.15 \cdot 10^{-5}$	$2^{34.2}$	$2^{39.4}$
10	$1.73 \cdot 10^{-7}$	$3.93 \cdot 10^{-8}$	$2^{50.0}$	$2^{55.8}$
16	$1.35 \cdot 10^{-14}$	$1.57 \cdot 10^{-15}$	$2^{97.2}$	$2^{104.7}$

Table 4.5: Probability of the characteristic and number of pairs to find 1 and 3 key bits in 8-bit CFB.

4.2.4 Discussion

Independent of our research, K. Ohta and M. Matsui developed a differential attack [99] that can be used against reduced versions of the DES in m-bit CFB mode. Their attack can be applied to the DES, reduced to eight rounds, if $m \geq 24$.

If m increases, our attack becomes also more efficient, and more key bits can be found (five if $m \geq 15$). If $m \geq 18$, three output bits of a single S-box are

known, which implies that a smaller reduction has to be applied to the exor-tables. This results in a reduction of the required number of chosen ciphertext pairs. If $m \geq 15$, two bits of $S8$ in the last round are known, and the input to the last round but one can be estimated. Only if $m \geq 28$ can information on both input and output of a single S-box be obtained in this way, thus allowing key bits in this round to be determined.

This differential attack would be impossible without IP^{-1}. In the absence of IP^{-1} only information on the output of S-boxes of the last round would be available. The security of the DES in 1-bit CFB could be improved if the bit is selected from the left half of the ciphertext. Selecting all the CFB bits from the left half of the ciphertext thwarts the proposed differential attacks for small values of m. Another way to strengthen the DES in the CFB mode against differential attacks could be a redesign of the S-boxes in the last round in order to decrease the difference between the $0-0$ and $1-1$ entries in the reduced exor-table. Finally, a completely different structure for the computation of the CFB bits from the inputs to the last round could be used.

4.3 Maximum Likelihood

Maximum likelihood techniques can be used to improve differential cryptanalysis of a large class of block ciphers. As explained in the previous sections, when the output of the cipher is only partially visible, the basic differential attack can recover only a small number of key bits. The maximum likelihood attack recovers more key bits.

The optimisations can be used for any block cipher of the Feistel type (cf. Section 2.1.1 and [37]). More generally, the analysis is applicable to all ciphers where the input of each round is first combined with a part of the key and subsequently transformed with a substitution.

The technique is also applicable to linear cryptanalysis. In that case however, the improvement is only marginal for the cases that have been studied.

4.3.1 General Idea

Consider a probabilistic relation μ between plaintext, ciphertext and the key. The probability of this relation depends on the actual values of the input of the substitution. It follows that

$$\mu(H, Z) = 0 \text{ with probability } p = p(I), \quad I = G \oplus K\,, \tag{4.19}$$

where

G, H are different sets of plaintext or ciphertext bits;

K, Z are different sets of key bits;

I is a set of input bits of the S-boxes in the first and/or last rounds.

Any kind of relation will suffice, provided that p is sufficiently large: for example a differential relation (such as in [10]) which requires a chosen plaintext attack, or a linear relation (such as in [77, 78]) which corresponds to a known plaintext attack.

Firstly the dependence of p on I is analysed. This off-line analysis gives the *theoretical pattern* $p_t(i)$.

Definition 4.2 *Let $\mu(H, Z)$ be a relation between Z, a set of key bits, and H, a set of plaintext and ciphertext bits, which holds with probability p. Let p depend on I. The table that for each value i of the variable I gives the value of $p(I)$ is called the* theoretical pattern *$p_t(I)$.*

Subsequently plaintext-ciphertext pairs for the secret key k are taken into account. Since the input of each round is added modulo 2 to the unknown key, the actual values of this input I are not known. However, for a constant value of the key k, identical g-values yield identical i-values. By combining plaintexts with the same value g and noting for which z $\mu(H = h, Z = z) = 0$, the *observed pattern* $p_p(G)$ is constructed.

Definition 4.3 *Let $\mu(H, Z)$ be a probabilistic relation between Z, a set of key bits and H, a set of plaintext and ciphertext bits. Assume that the probability p only depends on I, the actual input of certain S-boxes. Let $I = G \oplus K$, where K denotes unknown key bits, and G denotes bits that are visible in the plaintext or the ciphertext. Consider a number N of plaintext-ciphertext pairs. The table that gives, for each row g and for each column z, the fraction of text pairs for which $\mu(h, z) = 0$, is called the* observed pattern *p_p. The row g is denoted $p_p(g)$.*

Proposition 4.4 *Under the above mentioned conditions, the a posteriori distribution of K and Z only depends on the theoretical pattern $p_t(I)$ and the observed pattern p_p.*

Proof: Bayes' rule says:

$$\Pr(Z, K \mid p_p) = \frac{\Pr(p_p \mid Z, K) \Pr(Z, K)}{\Pr(p_p)} . \tag{4.20}$$

$\Pr(Z, K)$ is given by the a priori distribution of Z and K. Application of the sum rule gives

$$\Pr(p_p) = \sum_{z,k} \Pr(p_p \mid Z = z, K = k) \Pr(Z = z, K = k) ,$$

which is independent of the values of Z and K. Therefore (4.20) reduces to

$$\Pr(Z, K \mid p_p) \sim \Pr(p_p \mid Z, K)\,. \tag{4.21}$$

Application of the product rule gives

$$\Pr(p_p \mid Z, K) = \prod_g \Pr(p_p(G = g) \mid Z, K)\,.$$

Under the condition that p only depends on I, $\Pr(p_p(G) \mid Z, K)$ follows from the binomial statistic with probability $p = p_t(G \oplus K)$. ∎

Assuming that each value of G occurs with the same probability, one can conclude that the most probable value of Z, K is the one that gives the closest match for $p_p(G \oplus K)$ and $p_t(I)$, after a guess for Z. Figure 4.3 gives a graphical example where Z is a single bit and K can take values between 0_x and $3F_x$. A matching for p_p and p_t is good if peaks are mapped to peaks, and valleys to valleys.

An advantage of this approach is that it results in the complete a posteriori distribution of Z, K. This can be used to reduce the required number of texts. Instead of processing a fixed number of texts, one can look at the probability distribution after processing N texts and decide whether the attack requires more texts, or make a guess about the correct key value. The optimal value of N is determined by the relative cost of an encryption versus the cost of computing the distribution. However, the most important effect is the decoupling between the number of visible output bits and the number of recoverable key bits. This is now illustrated with two examples.

4.3.2 Application of the Maximum Likelihood technique

Two maximum likelihood based attacks are presented. The first attack was meant to improve the linear attack on the DES. However, the expected improvement was not observed. The second example improves the differential attack on the DES in CFB-mode.

4.3.3 The Linear Attack

Description of the attack

For $\mu(H, Z)$ the relations given by M. Matsui [78] are used. In order to analyse R rounds of the DES, an $(R-2)$-round linear relation is required. For the linear attack the parameters from (4.19) correspond to:

H**:** the exor of the plaintext and ciphertext bits that appear in the relation;

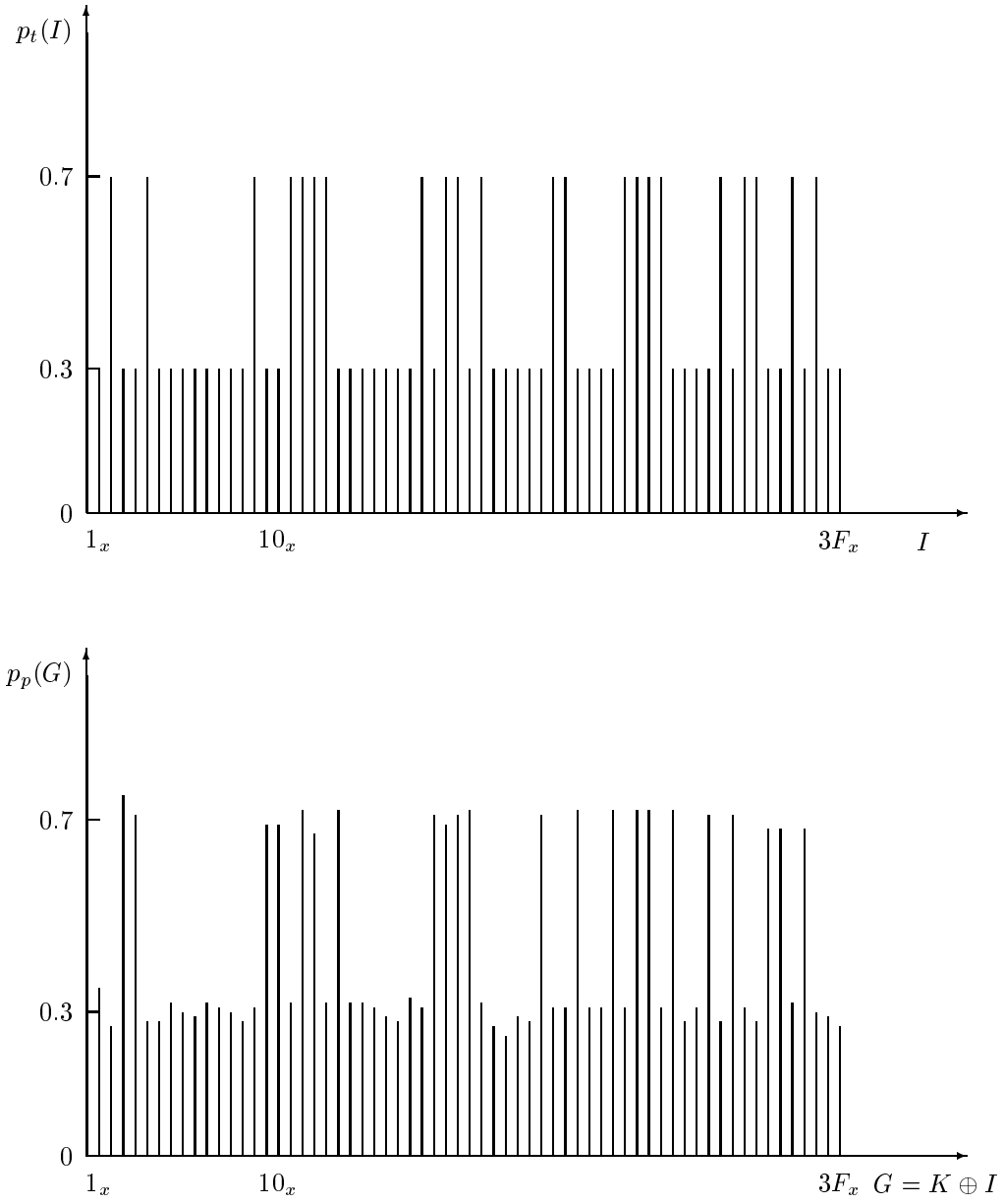


Figure 4.3: Graphical example of a match between p_t and p_p. The best match is given for $k = \texttt{01}_\texttt{x}$ and $z = 0$.

Z: the exor of the key bits that appear in the relation;

I: the 12 input bits that enter the S-boxes that are approximated in the first or last round;

G: the 12 corresponding plaintext and ciphertext bits;

K: the 12 corresponding key bits.

The relation can be divided into two parts: over the middle $R-2$ rounds it has a fixed probability p_{R-2}, over the first and the last round it holds with probability 1 or 0, depending on the value of I. The most probable value z, k is the value that gives the closest match for

$$p_p(G) \quad \text{and} \quad p'_t(G)\,,$$

where the pattern p'_t is defined as follows: for every g,

$$p'_t(g) = (1-z)p_t(g \oplus k) + z(1 - p_t(g \oplus k))\,.$$

Algorithm 4.1 can be used to determine the closest match.

Algorithm 4.1 Using maximum likelihood principles to optimize a linear attack.

```
calculate the theoretical pattern p_t
   p_t(i) = 1 if μ(i,0) = 0 with probability p_{R-2}
initialize p_p(h||g) = 0, for h||g = 0..2^13 - 1
repeat
    for i = 1..N do
        take an encryption
        increment counter p_p(g + 2^12 * h)
    calculate the a posteriori distribution of Z||K
until a posteriori distribution satisfies stop criterion
```

It is clear that the data collecting phase is identical to the one described in [78]. The difference is in the information extraction phase. The original linear attack assumes that wrong subkey guesses will yield a low probability for the relation μ. This assumption is known as *the hypothesis of wrong key randomisation* [45]. Since the maximum likelihood method looks at the complete pattern of the information, it will remain effective even when this hypothesis does not hold and two or more guesses for the subkey in a round yield about the same probability for the relation μ. Referring to the graphical example of

Figure 4.3, the original linear attack only examines whether the highest peak of p_p is mapped to the highest peak of p_t, while the maximum likelihood technique considers the mapping of all peaks and valleys.

Experimental results

Application to the DES: The algorithm was applied to the DES and implemented on a cluster of DEC and HP workstations with the tool PVM3.3.3 [42].

The results of the maximum likelihood method and Matsui's method are almost identical. Experiments were performed for the DES reduced to eight and twelve rounds with various numbers of plaintexts. Also the experimental probability that the right 13 bit key is among the t highest ranked candidates for both methods was measured, with $t = 1, 2, \ldots, 32$. This also gives very similar results for both methods.

Figure 4.4 gives the probability of success as a function of the number of plaintexts used for the case of the DES, reduced to eight rounds, for $t = 1$ and $t = 30$. Similar curves are obtained for the case of the DES, reduced to twelve rounds.

The probability of finding k does not increase significantly. A possible explanation is that for the linear relation which is used, the most likely subkey has a higher probability than the other ones (there is only one peak). Therefore, Matsui's straightforward approach is sufficient.

Akelarre: Akelarre [5] is a block cipher that was proposed at SAC'96. The cipher has a block length of 128 bits and a variable key length (always a multiple of 64 bits). Its round function is built from components of IDEA [69] and RC5 [115]. As can be seen in Figure 4.5, Akelarre uses modular addition, exors and rotations. The round function has the typical IDEA structure, where the Multiplication-Addition structure has been replaced by an Addition-Rotation structure. This AR-structure is based on RC5, but for the purposes of this attack, it can be considered as a black box.

The round function of Akelarre has a serious weakness. Denoting the four input words with u_1, u_2, u_3, u_4 and the four output words with v_1, v_2, v_3, v_4, the following invariant holds:

$$(v_1 \oplus v_3) \| (v_2 \oplus v_4) = \mathrm{rot}_{r \bmod 64}(u_1 \oplus u_3) \| (u_2 \oplus u_4), \tag{4.22}$$

where r denotes the rotation key. This invariant can be concatenated for *any* number of rounds (Akelarre was originally proposed with four rounds). The cipher can be broken with a linear attack, using only ciphertexts. This is done by guessing the least significant bytes of some keys in the input and output transformation and verifying whether (4.22) holds with high probability. Once the

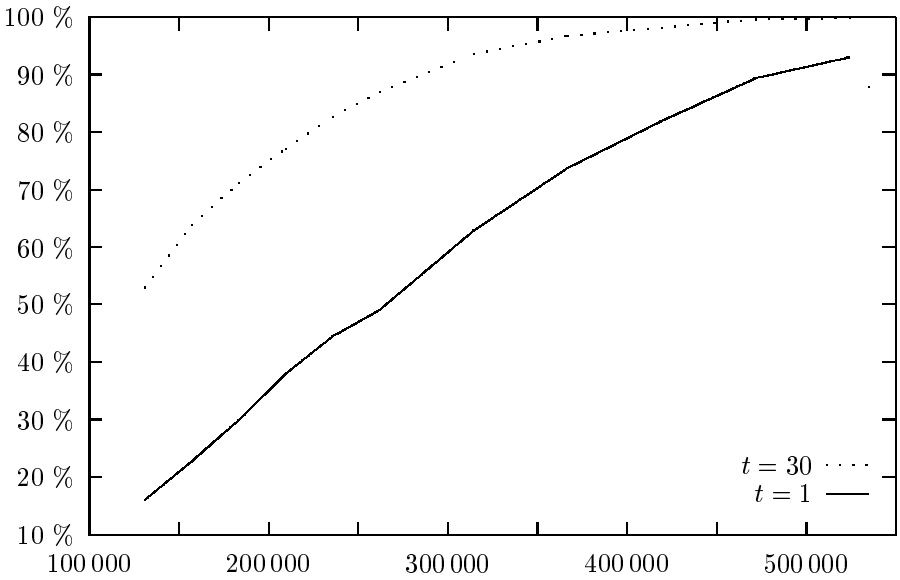


Figure 4.4: Probability of success as a function of the number of plaintexts used for the DES, reduced to eight rounds (t denotes the number of candidates for $z||k$).

least significant bytes are determined, we proceed with the next bytes. However, because the input and output transformation have no strong nonlinear elements, the hypothesis of wrong key randomisation does not hold. It has been experimentally verified that the standard linear attack does not work here, whereas the maximum likelihood technique allows the keys to be found very easily. The attack was tested by encrypting the LaTeX source of this text: 5000 ciphertexts (625 kbyte) suffice to break the cipher.

4.3.4 The Differential Attack on the m-bit CFB mode

Description

In order to simplify the discussion, only the attack on the DES reduced to 4 rounds in the 8-bit CFB mode will be explained. The extension to attacks on more rounds as described in Section 4.2 is straightforward.

In 8-bit CFB mode, the extended differential attack from Section 4.1 allows information on key bit k_{44}, which enters S3 in the last round, to be obtained.

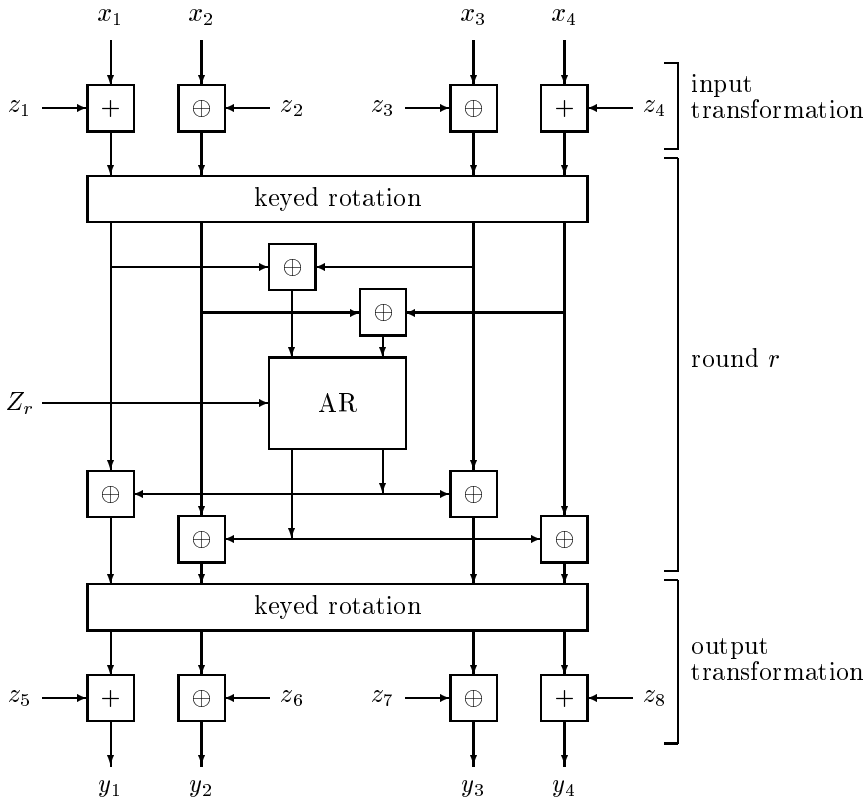


Figure 4.5: Computational graph of Akelarre.

In Section 4.2 it was shown that in order to achieve this goal, the characteristic of Figure 4.2 is optimal. This characteristic yields an input exor of $\mathtt{01}_{\mathtt{x}}$ for S3 in the last round.

Only a single input bit of S3 is visible in the output: in the extended differential, pairs for which this input exor equals 1 are filtered. As explained in Section 4.2.2, the attack can be improved by filtering in addition the pairs with output exor $\mathtt{00xx}_{\mathtt{x}}$ for S3. The fraction of right pairs after filtering is now equal to 0.392, assuming that the wrong pairs yield a uniform distribution of output exors.

In the first round, only S2 is active. The input exor is equal to $\mathtt{08}_{\mathtt{x}}$, and the corresponding right output exor is equal to $\mathtt{A}_{\mathtt{x}}$; it has a probability of 1/4. The corresponding row of the exor-table is the last row of Table 4.6. The upper rows of Table 4.6 are rows from the extended exor-table of S2. Only the rows with input exor $\mathtt{08}_{\mathtt{x}}$ are shown. Rows that differ only in the third or fourth input bit are merged. Due to the symmetry of the exor operation, the rows corresponding to the values $(a, a \oplus \mathtt{08}_{\mathtt{x}})$ and $(a \oplus \mathtt{08}_{\mathtt{x}}, a)$ must always be merged. This means that the value of the third input bit from the left has no influence. In this particular case, it turns out that the value of the fourth bit also has no influence on the output exor.

The pairs that follow the characteristic in the first round follow the characteristic in the last round with a probability of 3/4; in this case they are filtered with probability 2/64. If the pair does not follow the characteristic in the last round then it is filtered with probability 5/8. This yields a probability of filtering of

$$\frac{3}{4} \cdot \frac{2}{64} + \frac{1}{4} \cdot \frac{5}{8} = \frac{23}{128} .$$

For pairs which do not follow the characteristic in the first round, the probability of filtering is equal to $5/8 = 80/128$.

The filtering probability of a pair depends on the probability that it passes the first round of the characteristic. This probability depends only on the value of four input bits, that are given by the exor of four known plaintext bits and four unknown key bits. This means that the relation μ can be defined as follows:

$$\mu = 1 \Leftrightarrow \text{ the pair is filtered} .$$

Experimental results

Finding the 4 key bits with a success probability of 90% requires on average 280 pairs; in order to increase this probability to 98% the attack requires on average 330 pairs. For the 8-bit CFB mode one of these 4 key bits overlaps with the 3 bits found with the extended differential attack from Section 4.2. It is

	output exors (hexadecimal)															
input values	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
$00xx00_x \oplus 00xx00_x$	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0	0
$00xx01_x \oplus 00xx01_x$	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0
$00xx10_x \oplus 00xx10_x$	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
$00xx11_x \oplus 00xx11_x$	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2
$01xx00_x \oplus 01xx00_x$	0	0	0	0	0	2	0	2	0	0	0	0	0	0	0	0
$01xx01_x \oplus 01xx01_x$	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
$01xx10_x \oplus 01xx10_x$	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0
$01xx11_x \oplus 01xx11_x$	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2
$10xx00_x \oplus 10xx00_x$	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
$10xx01_x \oplus 10xx01_x$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0
$10xx10_x \oplus 10xx10_x$	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
$10xx11_x \oplus 10xx11_x$	0	0	0	2	0	0	0	2	0	0	0	0	0	0	0	0
$11xx00_x \oplus 11xx00_x$	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0
$11xx01_x \oplus 11xx01_x$	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0	0
$11xx10_x \oplus 11xx10_x$	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0	0
$11xx11_x \oplus 11xx11_x$	0	0	0	2	0	2	0	0	0	0	0	0	0	0	0	0
input exor = 08_x	0	0	0	4	0	4	0	8	0	10	16	6	6	0	6	4

Table 4.6: The rows with input exor 08_x from the exor-table and the extended exor-table of S2. Rows that only differ in the third or the fourth bit are merged.

possible to exploit this overlap to reduce the number of required pairs, but this optimisation has not been implemented.

It is clear that these results can be extended directly to the attacks on more rounds. The main difference is that these attacks use another characteristic, which activates different S-boxes. The number of key bits which can be found is equal to the number of input exor bits equal to 0 in the active S-boxes, minus the number of input bits which give no information (like the fourth input bit of S2 above). In addition overlap between key bits should be taken into account. Due to the nature of the attack, only probabilities in the first round are modified, resulting in an increase in the number of chosen ciphertext pairs with a constant factor between three and four. The attack still presents no serious threat for the full DES in the CFB mode, but the attack on eight rounds becomes more realistic, since it can find ten key bits with about 2^{41} chosen ciphertexts. Note that in some implementations of the CFB mode, the number of rounds was reduced from 16 to 8 in order to improve the performance. In comparison with the CBC mode of the full DES, the 8-bit CFB mode with the DES reduced

to eight rounds requires $64/8 \cdot 8/16 = 4$ times more work and gives a smaller security margin against a differential attack.

4.4 Differential Cryptanalysis of Hash Functions

In this section an improvement of the differential attack on hash functions based on block ciphers is presented [109]. In Chapter 2 it is explained how hash functions can be constructed from block ciphers. Twelve secure variants for constructions where the length of the hash result equals the block length are described in [103]. Only four of these variants can be attacked with the described differential attack, since it requires the cryptanalyst to have explicit control over the plaintext input of the block cipher, and that the key is fixed. The hash function studied here uses the DES as compression function in the following construction:

$$h^i = f(x^i, h^{i-1}) = \mathrm{DES}[h^{i-1}](x^i) \oplus x^i \,. \tag{4.23}$$

A differential attack on block cipher based hash functions is similar to an attack on the underlying block cipher, but there are some important differences. By using the specific properties of the collision attack on hash functions, the work factor to find a pair that follows the characteristic can be reduced. A new family of differential characteristics is proposed that are especially useful in combination with the improved attack. Attacks on a hash function based on DES variants reduced to 12, 13 or 15 rounds become faster than brute force collision attacks. These results have been published in [109]

4.4.1 Properties of the Hash Mode

An attack that is always applicable to hash functions is a brute force collision search. Because of the birthday paradox this attack requires approximately $2^{(l+1)/2}$ encryptions to produce a collision for a hash function with an l-bit output. For the case of the DES, $l = 64$ and about $2^{32.5}$ encryptions are required.

Differential cryptanalysis can be applied to hash functions in the same way as to the corresponding block ciphers, but there are some important differences [104].

- For the case of a collision attack, the plaintext input can be controlled. This a the differential attack on the hash function feasible. A differential attack on a block cipher used as an encryption device, in contrast, is mainly of theoretic interest.
- The key is known. Sometimes the key can be chosen freely, or the best alternative can be chosen out of a set of possible keys. This can be exploited in several ways. Firstly, when searching for a collision, input values

can be selected that follow the characteristic with probability one in certain rounds. Precomputation of a few tables allows the inputs of four or five consecutive rounds to be chosen (cf. Section 4.4.2). Secondly, the probability of some characteristics is key-dependent. If the key can be controlled then it is chosen to be optimal for the used characteristic, otherwise a characteristic with optimal probability for the given key can be selected. Thirdly, an early abort strategy can be used: as soon as it is clear that the pair is a wrong pair, it is discarded. For most inputs, only a few rounds have to be computed.

- There are more restrictions on the characteristic: in block cipher analysis the most probable characteristic can be used. For hash functions, the characteristic has to produce a collision, i.e., the output exor of the round function f must be zero. For the construction of (4.23), this means that the output exor of the block cipher has to match the input exor. Moreover, the characteristic must cover all the rounds: 1R-, 2R-, or 3R-attacks do not apply. This reduces the probability of the characteristic.

- Only one right pair is required to find a collision or a second preimage.

In the rest of the section only collision attacks are considered.

4.4.2 Choosing Inputs

In a collision attack, the input values (or messages) are chosen arbitrarily. This freedom is used to enhance the success probability. A naive approach is to select messages that will follow the proposed characteristic in the first two rounds with probability one. Algorithm 4.2 allows four rounds to be passed with probability one. A very simple extension of the algorithm allows five rounds to be passed by adding an extra round at the beginning. Figure 4.6 defines the notation for intermediate values of the hashing.

4.4.3 Good Characteristics

It has already been observed in [104, 59] that it is a non trivial problem to find good even-round characteristics for the hash function of (4.23). Since all known Feistel ciphers have an even number of rounds, the even-round characteristics are of most interest. One-round iterative characteristics can have an arbitrary number of rounds, but they have a very low probability of $\frac{1}{234}$ per round (cf. Figure 4.7). Two-round iterative characteristics have the highest probability for seven rounds or more [80], but in hash functions they can only be applied to DES variants with an odd number of rounds. This can be concluded from Figure 4.7. Each $0 \leftarrow x'$ round has on average a probability of $\frac{1}{234}$. Dependent on the round

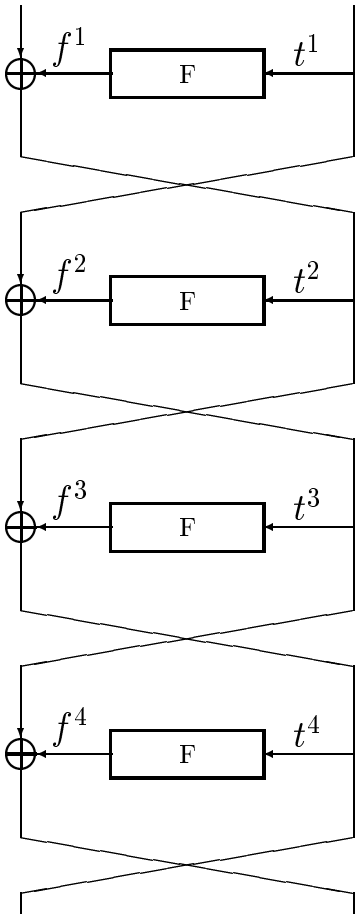


Figure 4.6: Four rounds of the DES.

Algorithm 4.2 An algorithm to easily find the good pairs of a given four round differential characteristic.

Step 1:
For a given key k calculate table T_1 that lists all values of t^1 that follow the characteristic in round 1. Repeat this for tables T_2 and T_3 that list the values of f^2 and t^3 respectively. Since in each round only a few S-boxes are active, these tables can be reduced in size by the use of 'don't cares': the inputs of non-active S-boxes are arbitrary and thus not specified.

Step 2:
Match these three tables and look for all possible values of (t^1, f^2, t^3).

Step 3:
Calculate table T_4 with all values for t^4. For every (t^1, f^2, t^3) 'invert' round two and try to match the possible values of t^2, $F[k](t^3)$ and t^4.

Step 4:
For each match found, calculate the inputs to the first round.

key, this probability is $\frac{1}{146}$ or $\frac{1}{585}$. For 13 rounds, the attack requires the same number of encryptions as the brute-force collision attack. However, since the DES has 16 rounds, any serious attack requires a characteristic with an even number of rounds.

$$\begin{array}{ccc} (x' & , & x') \\ 0 & \leftarrow & x' \\ 0 & \leftarrow & x' \\ 0 & \leftarrow & x' \\ 0 & \leftarrow & x' \\ 0 & \leftarrow & x' \\ (x' & , & x') \end{array} \qquad \begin{array}{ccc} (x' & , & 0) \\ 0 & \leftarrow & 0 \\ 0 & \leftarrow & x' \\ 0 & \leftarrow & 0 \\ 0 & \leftarrow & x' \\ 0 & \leftarrow & 0 \\ (x' & , & 0) \end{array}$$

Figure 4.7: One-round and two-round iterative characteristics.

In [104], B. Preneel proposes the search for an input value x' that is a good fixed point ($x' \leftarrow x'$) and a good building block for an iterative characteristic ($0 \leftarrow x'$). In [59], L.R. Knudsen shows that such a characteristic cannot have a high probability. The problem is that all x' with a high probability for $0 \leftarrow x'$ have low probability for $x' \leftarrow x'$, and vice versa. L.R. Knudsen therefore proposes the use of an iterative characteristic based on a special four round building block (cf. Figure 4.8). This building block has probability $2^{-23.6}$, averaged over all keys, and $2^{-23.0}$ for optimal keys. For a DES variant with eight rounds, the work factor is comparable to a brute-force collision search.

The following approach gives a better result, especially in combination with

$$
\begin{array}{ccc}
(\texttt{E0000004}_{\texttt{x}} & , & \texttt{E0000004}_{\texttt{x}}) \\
\texttt{00000002}_{\texttt{x}} & \leftarrow & \texttt{E0000004}_{\texttt{x}} \\
\texttt{00000002}_{\texttt{x}} & \leftarrow & \texttt{E0000006}_{\texttt{x}} \\
\texttt{00000002}_{\texttt{x}} & \leftarrow & \texttt{E0000006}_{\texttt{x}} \\
\texttt{00000002}_{\texttt{x}} & \leftarrow & \texttt{E0000004}_{\texttt{x}} \\
(\texttt{E0000004}_{\texttt{x}} & , & \texttt{E0000004}_{\texttt{x}})
\end{array}
$$

Figure 4.8: The 4-round iterative characteristic of L. Knudsen.

$$
\begin{array}{ccc}
(x' & , & 0) \\
0 & \leftarrow & 0 \\
0 & \leftarrow & x' \\
\cdots & & \cdots \\
x' & \leftarrow & x' \\
0 & \leftarrow & x' \\
\cdots & & \cdots \\
 & & \\
 & & \\
(x' & , & 0)
\end{array}
\qquad\qquad
\begin{array}{ccc}
(x' & , & 0) \\
0 & \leftarrow & 0 \\
0 & \leftarrow & x' \\
\cdots & & \cdots \\
a' & \leftarrow & x' \\
x' \oplus b' & \leftarrow & a' \\
x' \oplus a' & \leftarrow & b' \\
b' & \leftarrow & x' \\
0 & \leftarrow & 0 \\
(x' & , & 0)
\end{array}
$$

Figure 4.9: Two alternatives for the transient part of an iterative characteristic.

the algorithm of Section 4.4.2. Take a value x' with good probability for $0 \leftarrow x'$. Instead of inserting one $x' \leftarrow x'$ round, insert five 'transient' rounds (cf. Figure 4.9). These five rounds have a low probability that is, however, better than the fixed point construction. The low probability is not a problem since the input values of these transient rounds can be chosen in such a way that the rounds are passed with probability one. A computer search has indicated that the best transient rounds have a symmetrical pattern. The computer search considered the 50 x'-values with the best $0 \leftarrow x'$ probabilities. For each x', all possible a'-values and b'-values were examined. The best combination is $x' = \texttt{0019 6000}_{\texttt{x}}$ and $a' = b' = \texttt{0445 0180}_{\texttt{x}}$. The probabilities of the different rounds are given in Table 4.7. Note that there exist x'-values that yield a lower probability for which the optimal a' and b' are different.

The fact that the probability of these rounds depends on the key can be exploited to reduce the work/success ratio: eliminate the keys that give the characteristic a low (or zero) probability. For this choice of x' and a' (cf. Figure 4.7), 4.5% of the keys have a non-zero probability. Stronger selection criteria for the keys are possible. Table 4.8 gives the theoretical probabilities and work

structure	probability	comments
$0 \leftarrow x'$	2^{-8}	key independent
$a' \leftarrow x'$	$2^{-10.8}$	$2^{-9.95}$ for 50% of the keys
$x' \oplus a' \leftarrow a'$	$2^{-20.5}$	$2^{-18.1}$ for 4.7% of the keys

Table 4.7: The different rounds in the characteristic and their probabilities.

# rounds	probability ($\log_2$)	work factor ($\log_2$)
8	-65.8	8
9	-28.8	5
12	-81.8	21.4
13	-43.2	18.9
14	-89.8	29.2
15	-50.3	25.9
16	-97.8	37.0

Table 4.8: A survey of probabilities of the characteristics and theoretical work factors for reduced versions of the DES.

factors for DES variants with various number of rounds. The probability of the given characteristic, is averaged over the keys with non-zero probability only. The work factors are calculated as follows: the reciprocal of the probability of the rounds where the input values cannot be controlled multiplied by a reduction factor that takes the early abort strategy into account. The numbers for DES variants with an odd number of rounds are obtained by choosing input values for five arbitrarily chosen consecutive rounds. The characteristic is the best 2-round iterative characteristic of [10].

4.5 Conclusions

Three extensions to differential cryptanalysis were introduced and successfully applied in practical situations.

The first extension improves the differential attack on block ciphers in m-bit CFB mode (m small). The basic differential attack performs badly in this situation because most good pairs will suggest all key values and therefore the number of requested good pairs increases significantly. The extended attack allows the number of required plaintexts to be decreased. For an attack on the DES reduced to eight rounds, 2^{40} pairs are required to recover three key bits.

This result has been published in [105].

The second extension introduces the principle of maximum likelihood to improve differential and linear attacks. The number of recoverable key bits by a differential attack on encryption modes that hide a part of the output is raised. This is again very useful to extend the attack on block ciphers in m-bit CFB mode, where previously the number of recoverable key bits was upper bounded by m. For the case of the DES in 8-bit CFB mode, the number of recoverable key bits is increased from three to ten. The information extraction phase of a linear attack is also optimised. This makes it possible to extend the linear attack to cases where the 'hypothesis of wrong key randomisation' does not hold, e.g. in a ciphertext-only attack on Akelarre.

The third extension is an improvement of the differential attack on hash functions that are based on block ciphers. By making use of the specific advantages of the hash function context, a better attack is developed. A special kind of differential characteristic is introduced, optimally suited for this attack. When applied to a hash function that uses the DES as its round function, the differential attack becomes faster than the birthday attack if the DES is reduced to 12, 13 or 15 rounds. This result has been published in [109].

Chapter 5

Key Dependent Analysis

What we see depends on mainly what we look for.
John Lubbock

The hypothesis of stochastic equivalence (cf. Section 3.4 and the work of X. Lai [69]) assumes that the probability of a differential characteristic (or the deviation of a linear relation) for a given fixed key can be approximated by the average probability of the characteristic (or deviation of the relation) over all key values (cf. Section 3.4).

In this chapter the distinction between the unconditional probability of a differential and its probability conditioned on the key value is explained. A completely equivalent distinction can be made between unconditional and conditional input-output correlations.

By taking into account the key dependence of differential characteristics and input-output correlations it is possible to mount effective attacks, for instance on ciphers with nonlinear key addition. This is illustrated with two attacks on reduced IDEA, which have been published in [14], and an attack on MAA, which has been published in [106].

5.1 Probability of a Differential

One Round

Consider one round of an iterative block cipher. The stochastic variable that produces the text input is denoted by X, the key input by K and the output by Y; χ denotes the number of possible text inputs and κ the number of possible keys. The probability that the round transformation $\rho[k](x)$ transforms an input

difference x' into an output difference y' is given by:

$$\begin{aligned}\Pr(X' = x' \mapsto Y' = y') &= \frac{\#\{x, k \mid \rho[k](x * x')\Delta\rho[k](x) = y'\}}{\chi \cdot \kappa} \\ &= \frac{\sum_x \sum_k \sum_y \theta_{\rho[k]}(x * x', y * y') \cdot \theta_{\rho[k]}(x, y)}{\chi \cdot \kappa},\end{aligned}$$

where $\theta_{\rho[k]}$ is the characteristic function of the round transformation $\rho[k]$. Application of the sum rule leads to

$$\Pr(X' = x' \mapsto Y' = y') = \sum_k \Pr(X' = x' \mapsto Y' = y' \mid K = k) \cdot \Pr(K = k), \tag{5.1}$$

where

$$\begin{aligned}\Pr(X' = x' \mapsto Y' = y' \mid K = k) &= \frac{\#\{x \mid \rho[k](x * x')\Delta\rho[k](x) = y'\}}{\chi} \\ &= \frac{\sum_x \sum_y \theta_{\rho[k]}(x * x', y * y') \cdot \theta_{\rho[k]}(x, y)}{\chi}.\end{aligned}$$

Since determining $\Pr(X' \mapsto Y' \mid K = k)$ for all values k is often not feasible, and since the actual value of K is not known to the cryptanalyst, it is usually assumed that for most values k it holds that $\Pr(X' \mapsto Y' \mid K = k) \approx \Pr(X' \mapsto Y')$. This assumption is known as *'the hypothesis of stochastic equivalence'* [69]. The validity of this assumption depends on the structure of the round transformation. A necessary condition is that the key addition is linear with respect to the difference operation Δ. The next example shows a round transformation where $\Pr(X' \mapsto Y' \mid K = k) \equiv \Pr(X' \mapsto Y')$. For a generic round transformation, the conditional probability may heavily depend on the actual key value. For the round transformation of the DES, this was already observed in [57].

Example 5.1 *Let the round transformation be defined as $\rho[k](x) = \gamma(x * k)$, with γ an arbitrary substitution and $*$ the commutative operation that is used for Δ, then*

$$\begin{aligned}\Pr(X' = x' \mapsto Y' = y' \mid K = k) &= \chi^{-1} \cdot \#\{x \mid \rho[k](x * x')\Delta\rho[k](x) = y'\} \\ &= \chi^{-1} \cdot \#\{x \mid \gamma(x * x' * k)\Delta\gamma(x * k) = y'\} \\ &= \chi^{-1} \cdot \#\{x \mid \gamma(x * x')\Delta\gamma(x) = y'\}.\end{aligned}$$

Therefore $\Pr(X' \mapsto Y' \mid K = k) \equiv \Pr(X' \mapsto Y')$.

More Rounds

Since encryption algorithms necessarily involve nonlinear elements in the round transformation, the conditional probability of differentials over more than one round always depends on the key value. Example 5.2 illustrates this. Equivalent observations can be made about the difference between unconditional and conditional input-output correlations.

Example 5.2 *Consider the round transformation* $\rho[k](x) = x^3 \oplus k$*, defined over* $GF(2^3)$*. If* $\oplus$ *is taken as the difference operation, the conditional probability of the one-round differentials is key independent. The first column of Table 5.1 gives these probabilities for a fixed input difference* $\mathtt{1_x}$*, and all the possible output differences. The two-round transformation can be written as* $\rho[k^2](\rho[k^1](x)) = ((x \oplus k^1)^3 \oplus k^2)^3$*. The probability of the two-round differentials depends on the value of* k^2*, which is also shown in Table 5.1.*

y'	p_1	$p_2(k^2)$ k^2							
		$\mathtt{0_x}$	$\mathtt{1_x}$	$\mathtt{2_x}$	$\mathtt{3_x}$	$\mathtt{4_x}$	$\mathtt{5_x}$	$\mathtt{6_x}$	$\mathtt{7_x}$
$\mathtt{0_x}$	0	0	0	0	0	0	0	0	0
$\mathtt{1_x}$	0.25	1	0.25	0	0.25	0	0.25	0	0.25
$\mathtt{2_x}$	0	0	0	0	0.25	0.5	0	0	0.25
$\mathtt{3_x}$	0.25	0	0.25	0	0	0.5	0.25	0	0
$\mathtt{4_x}$	0	0	0	0	0.25	0	0.25	0.5	0
$\mathtt{5_x}$	0.25	0	0.25	0	0	0	0	0.5	0.25
$\mathtt{6_x}$	0	0	0	0.5	0	0	0.25	0	0.25
$\mathtt{7_x}$	0.25	0	0.25	0.5	0.25	0	0	0	0

Table 5.1: The conditional probabilities $\Pr(X' = \mathtt{1_x} \mapsto Y' = y' \mid K)$ for Example 5.2. The rows correspond to the different values of y'. The first column list the (key independent) probabilities for one-round differentials (p_1), the next columns list the probabilities for different values of the second round key ($p_2(k^2)$).

Markov Ciphers

Definition 5.1 (X. Lai [67]) *An iterated cipher is a* Markov cipher *if and only if their exists a difference operation* Δ *such that the round operation of the block cipher has the property that for all nonzero input differences the probability*

$$\Pr(X' \mapsto Y') = \Pr(X' \mapsto Y' \mid X = x),$$

if the round key is uniformly random.

The definition uses probability distributions that are not conditional on the value of the key; they are thus averaged over all the key values. This means that from the point of view of key dependent analysis, there is no difference between Markov ciphers and other ciphers.

Exploitation of Key Dependence

Four approaches are discussed to exploit the key dependence of characteristics and relations.

1. If the conditional probability distribution of a differential is strongly non-uniform it is sometimes possible to mount an attack that works only for keys with

$$\Pr(X' = x' \mapsto Y' = y' \mid K = k) \gg \Pr(X' = x' \mapsto Y' = y') .$$

   These keys are called *weak keys*. This approach is illustrated in [23] and in Section 5.3.

2. Another approach is to use a set of differentials $\{(x^{i'}, y^{i'})\}_{i=1}^{n}$. The success rate of an attack will typically be determined by the maximum of the probabilities of the differentials of the set. This approach is illustrated in Section 5.2.3.

3. In Section 5.2.4 an attack is presented that works if the conditional probability is either higher or lower than the unconditional probability. The success rate of the attack is determined by the difference between the probabilities. This is reminiscent of the key search method in a linear attack (cf. Section 3.3 and [77, 79]), where the correct key is identified by measuring the correlation between input bits and output bits.

4. A fourth approach is to measure experimentally the probability of a differential and use this to determine the used key.

The attacks presented in the next sections show that key dependent differentials and/or linear approximations can be used successfully to cryptanalyse ciphers that have no useful key independent differentials or input-output correlations.

5.2 Application to IDEA

5.2.1 Description of IDEA

The block cipher IDEA (International Data Encryption Algorithm) was proposed by X. Lai, J. Massey and S. Murphy in [69] as a strengthened version of PES (Proposed Encryption Standard) proposed in [68]. IDEA is an iterated block cipher, consisting of 8 rounds followed by an output transformation. It has a 128-bit key and operates on data blocks of 64 bits. The round transformation divides the data into four 16-bit blocks and uses three different operations: bitwise exor, addition modulo 2^{16} and multiplication modulo $2^{16}+1$ ($\texttt{0000}_\texttt{x}$ representing the element 2^{16}). *"Mixing operations from three different algebraic groups,"* is an important design concept of the cipher. Figure 5.1 shows the computational graph of IDEA. The two multiplications and the two additions in the middle of the round transformation correspond to the F-function in a Feistel Network and are called the MA-structure. The four 16-bit blocks that enter round r ($r = 0, \ldots, 7$) are denoted x_i^r, where $i = 0, 1, 2, 3$. The inputs of the output transformation are denoted x_i^8, and the ciphertext blocks are x_i^9. Every round uses six 16-bit subkeys z_i^r that are derived from the key with a simple rotating selection scheme.

The following expression will be used to denote one round of IDEA:

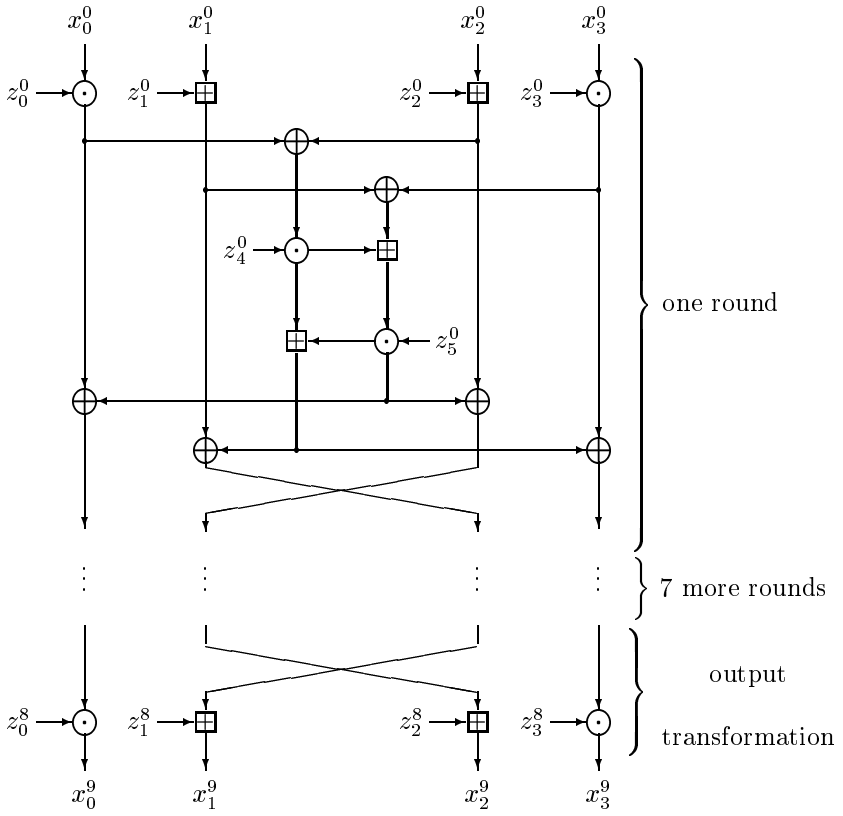
$$(a,b,c,d) \quad \rightarrow \quad (e,f,g,h) \quad \xrightarrow{(e\oplus g, f\oplus h)\rightarrow(k,l)} \quad (e\oplus l, g\oplus l, f\oplus k, h\oplus k)\,.$$

Here (a,b,c,d) denotes the input, consisting of four 16-bit words. The key addition transforms the input to (e,f,g,h). The MA-structure has input $(e\oplus g, f\oplus h)$ and output (k,l). The output of the round is then given by $(e\oplus l, g\oplus l, f\oplus k, h\oplus k)$. A differential characteristic is denoted in the same way as:

$$(a,b,c,d) \quad \overset{p_1}{\rightarrow} \quad (e,f,g,h) \quad \xrightarrow{(e\oplus g, f\oplus h)\overset{p_2}{\rightarrow}(k,l)} \quad (e\oplus l, g\oplus l, f\oplus k, h\oplus k)\,.$$

Now a,b,c,d,e,f,g,h,k,l denote differences of 16-bit words. Let p_1 and p_2 denote the probabilities of the transition. Note that the given probabilities will always be calculated assuming independent transitions.

IDEA was developed to resist differential cryptanalysis. In [67] X. Lai argues that for 3 rounds of IDEA there are no useful differentials and concludes that IDEA is resistant against a differential attack after 4 of its 8 rounds. In his work, differences are defined corresponding to the modular addition and multiplication operations of IDEA, in order to allow Markov theory to be applied (IDEA is a Markov cipher). However, in the analysis presented here the difference operation will always be the exor operation, to which the Markov theory does not apply here.



x_i^r : 16-bit text sub-block
z_i^r : 16-bit key sub-block
$\oplus$: bit-by-bit exor of 16-bit sub-blocks
$\boxplus$: addition modulo 2^{16} of 16-bit integers
$\odot$: multiplication modulo $2^{16}+1$ of 16-bit integers with the zero sub-block corresponding to 2^{16}

Figure 5.1: Computational graph for the encryption process of the IDEA cipher.

5.2.2 Key Dependent Differentials and Relations

Modular multiplication and, to a lesser extent, modular addition, have the property that the unconditional probability of the differentials is low. Table 5.2 shows this for multiplication modulo $2^8 + 1$ and addition modulo 2^8. However, from Table 5.2 it is also clear that the conditional probability of a differential more often takes a high value.

	Multiplication		Addition	
p	unconditional	conditional	unconditional	conditional
0	0.8	80.2	85.9	96.5
$0 < \cdot < 2^{-8}$	2.1	0	0	0
2^{-8}	94.7	0	0	0
2^{-7}	2.3	7.3	4.4	0
2^{-6}	0.07	6.6	4.7	0.3
2^{-5}	0.008	4.6	3.0	0.8
2^{-4}	0.005	1.1	1.4	1.0
2^{-3}	0.003	0.2	0.5	0.8
2^{-2}	0.002	0.04	0.1	0.4
2^{-1}	0	0.01	0.02	0.2
1	0	0.003	0.002	0.03

Table 5.2: Distribution of the conditional and unconditional probabilities of differentials for multiplication modulo $2^8 + 1$ and addition modulo 2^8. The entries are in percentages.

Table 5.3 gives the distribution of input-output correlations of these operations.

It can be observed that modular multiplication has no differentials with unconditional probability one (except for the trivial $0 \mapsto 0$) and neither does it have input-output correlations with value 0.5. Modular addition has one differential with probability one (change the most significant bit) and one linear relation with correlation one (in the least significant bit, because of the absence of a carry bit). In [23] the differentials with conditional probability one are chained to build differentials with (conditional) probability one for seven and eight rounds of IDEA. The key values for which this characteristic holds are called weak keys. There are 2^{51} key values that exhibit a seven-round differential with probability one.

In the following a differential-linear and a truncated differential attack will be presented that use key dependent differentials.

	Multiplication		Addition	
c	unconditional	conditional	unconditional	conditional
0	0.8	18.0	66.7	77.8
2^{-7}	0.001	0	3.0	0
2^{-6}	1.9	25.8	7.6	6.3
2^{-5}	57.8	17.9	9.9	3.6
2^{-4}	39.5	25.9	7.8	5.9
2^{-3}	0.003	10.5	3.9	3.8
2^{-2}	0	1.7	1.1	1.8
2^{-1}	0	0.1	0.09	0.6
1	0	0.008	0.002	0.1

Table 5.3: Distribution of the conditional and unconditional input-output correlations of multiplication modulo $2^8 + 1$ and addition modulo 2^8. The entries are in percentages.

5.2.3 Differential-Linear Cryptanalysis

In this section a differential-linear attack on IDEA, reduced to three rounds, is presented [14]. The final swap of the second and the third output block are omitted. The notation is defined in Figure 5.1.

A Set of Relations

Firstly we define the words $e^i = 2^{15-i}$, $i = 0, \ldots, 15$. We now consider the following set of linear relations:

$$e^{15} \bullet ((k \odot x) \oplus (k \odot (x \oplus e^i))) = 0\,, \tag{5.2}$$

with unconditional input-output correlation c_i and conditional correlations $c_i(k)$. Table 5.4 lists the values of c_i and the distributions of the $c_i(k)$ for some values of i. The table also gives the distribution of $c(k) = \max_i c_i(k)$, which is of the greatest interest for the purposes of this attack. Observe that $c(k) \gg c_i$ for all i and almost all k.

This set of relations will be used in the linear-differential attack. The next section gives a differential characteristic that produces the difference e^i at the input of a multiplication in the second round. The attack produces pairs for every value of e^i. The relation that has the highest correlation determines the data requirements of the attack.

					e^0	e^2	e^8	e^{15}	$c(k)$
c_i					2^{-1}	2^{-1}	2^{-1}	2^{-1}	
		$c_i(k)$	$=$	0	0.02	0.003	0.06	0	0
0	$<$	$c_i(k)$	$\leq$	2^{-9}	0.3	0.1	0.2	0	0
2^{-9}	$<$	$c_i(k)$	$\leq$	2^{-6}	1.9	2.0	1.8	2.1	0
2^{-6}	$<$	$c_i(k)$	$\leq$	2^{-4}	6.6	6.7	6.7	6.7	0
2^{-4}	$<$	$c_i(k)$	$\leq$	2^{-3}	8.8	8.8	8.8	8.9	0
2^{-3}	$<$	$c_i(k)$	$\leq$	2^{-2}	17.7	17.7	17.7	17.7	0.03
2^{-2}	$<$	$c_i(k)$	$\leq$	2^{-1}	35.3	35.3	35.3	35.3	2.1
2^{-1}	$<$	$c_i(k)$	$\leq$	1	29.3	29.3	29.3	29.3	97.9

Table 5.4: The correlation of Relation (5.2) for a subset of $\{e^i\}_{i=0}^{15}$. The distributions for different e^i-values are close to one another. The first row gives the unconditional correlations c_i. The next rows give the distribution of the conditional correlations $c_i(k)$. The last column gives the distribution of $c(k) = \max_i c_i(k)$. The distribution entries are in percentages.

The Differential Characteristics

The attack starts with guessing the value of z_3^0. The first round of the characteristic looks like the following:

$$(0, e^i, 0, \alpha^i) \overset{0.5}{\rightarrow} (0, e^i, 0, e^i) \xrightarrow{(0,0)\overset{1}{\rightarrow}(0,0)} (0, 0, e^i, e^i)\,.$$

With the (assumed) knowledge of z_3^0 it is possible to choose the values of x_3^0 and α^i such that $(x_3^0 \odot z_3^0) \oplus ((x_3^0 \oplus \alpha^i) \odot z_3^0) = e^i$. The probability of the first round then equals the probability of the transition $e^i \mapsto e^i$ in modular addition, which is 0.5. The first half of the second round of the characteristic can be described as follows:

$$(0, 0, e^i, e^i) \overset{0.5}{\rightarrow} (0, 0, e^i, \beta) \xrightarrow{(e^i,\beta)\overset{p_2}{\rightarrow}(\cdot,\cdot)} (\cdot, \cdot, \cdot, \cdot)\,.$$

The value of β is not important for the attack and is not specified. This means that the differential becomes a truncated differential. The probability of the truncated differential at the MA-structure of the second round equals 0.25. For good pairs the input exor of the first multiplication in the MA-structure of the second round equals e^i. It follows from (5.2) that for every key there exists at least one e^i such that the value of the least significant bit of the output of this multiplication is highly biased. From now on the differential is truncated even further by specifying only the least significant bit of the 16-bit words. The modular addition now reduces to an exor operation. Denoting the output exors

of the multiplications of the MA-structure with γ and δ, the second round of the characteristic can be completed:

$$(0,0,e^i,e^i) \xrightarrow{0.5} (0,0,e^i,\beta) \xrightarrow{(e^i,\beta)\xrightarrow{p_2}(\gamma\oplus\delta,\delta)} (\delta, e^i\oplus\delta, \gamma\oplus\delta, \gamma\oplus\delta\oplus\beta)\,.$$

Figure 5.2 shows the second round of the characteristic.

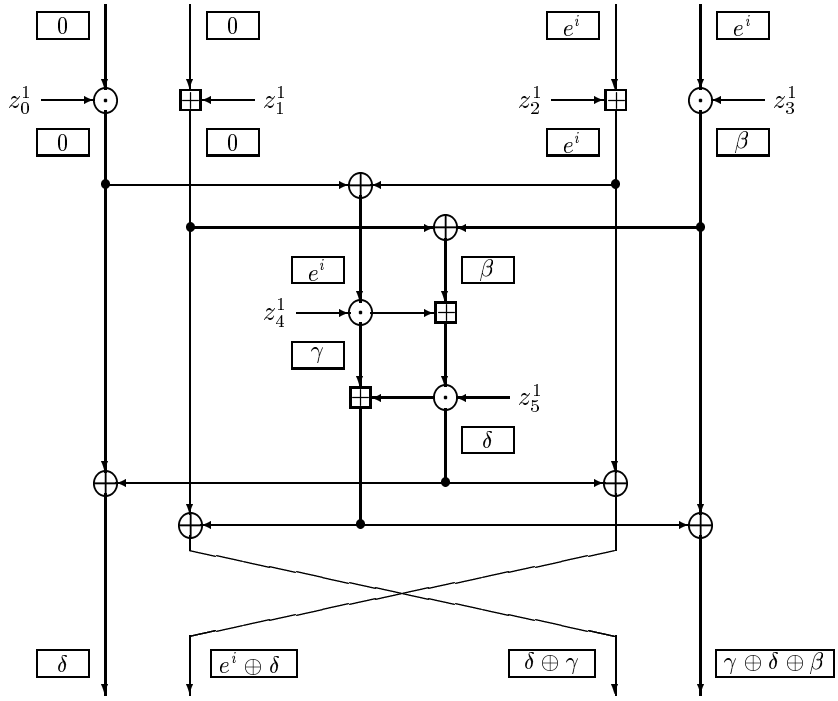


Figure 5.2: Second round of the characteristic for the linear-differential attack on three rounds of IDEA. At the end, only the least significant bits of the words are considered, such that the addition operation becomes equal to the exor operation.

Recovering z_4^2

The key input of the first multiplication of the MA-structure in the last round is z_4^2. The text input of this multiplication can be calculated as $x_1^3 \oplus x_3^3$. These values are visible in the ciphertext because the ciphertext is formed by the outputs of the third round. Denote the least significant bit of the output of this

multiplication ζ, and denote the least significant bit of the output of the second multiplication of the MA-structure with η. Then ζ can be expressed as follows.

$$x_2^{3\prime} = \delta \oplus e^i \oplus \zeta \oplus \eta \tag{5.3}$$

$$x_3^{3\prime} = \delta \oplus \gamma \oplus \eta \tag{5.4}$$

$$x_2^{3\prime} \oplus x_3^{3\prime} = e^i \oplus \zeta \oplus \gamma\,. \tag{5.5}$$

By assuming a value z_4^2 it is possible to calculate ζ, and thus also γ.

The attack proceeds as follows. A counter is initialised for every possible z_4^2. For every possible z_3^0 and e^i, a set of plaintext pairs is enciphered. Filtering of the wrong pairs is not possible. For every pair the value $x_1^3 \oplus x_3^3$ is calculated and by assuming a value for z_4^2 it is possible to calculate γ using (5.5). If $\gamma = 0$, the counter corresponding to the key value is incremented. The calculation is repeated for every value z_4^2. If z_3^0 was guessed correctly, then only two values for z_4^2 will be suggested after enough pairs have been encrypted: the correct z_4^2 and its additive inverse modulo $2^{16}+1$. Experimental results confirm that for wrong guesses of z_3^0 no key value will be suggested.

Since the best e^i is not known, the attack is tried with all e^i-values. Computer experiments show that for a correct guess of z_3^0, the attack requires at most 2^{14} chosen plaintext pairs. On average, a correct guess is made after 2^{15} trials, resulting in 2^{29} required plaintext pairs. Examining one plaintext pair takes a few exor operations and 2^{16} table look-ups, one for each value of z_4^2. Since 16 differentials are examined, the attack requires about 2^{20} simple operations, i.e., addition or exor, for each pair. The workload is therefore about 2^{49} simple operations. For a rough estimate of the equivalent number of encryptions, the required time for an addition is taken to be equal to the required time for an exor. The required time for a modular multiplication is estimated to be 3.5 times this time. This results in a workload of about $0.75 \cdot 2^{44}$ 3-round IDEA encryptions.

5.2.4 Truncated Differential Cryptanalysis

In this section a truncated differential attack on a reduced version of IDEA is presented [14]. The reduced version consists of three rounds and the output transformation. The notation is the same as in the previous section. Note that the ciphertext consists now of the words x_i^4.

Probability of the Truncated Characteristic

The three-round truncated differential characteristic used in the attack is the following:

$$\begin{array}{llll} (a,0,b,0) & \stackrel{2^{-16}}{\rightarrow} & (c,0,c,0) & \xrightarrow{(0,0)\stackrel{1}{\rightarrow}(0,0)} (c,c,0,0) \\ (c,c,0,0) & \stackrel{1}{\rightarrow} & (d,e,0,0) & \xrightarrow{(d,e)\stackrel{2^{-32}}{\rightarrow}(e,d)} (0,d,0,e) \\ (0,d,0,e) & \stackrel{2^{-16}}{\rightarrow} & (0,f,0,f) & \xrightarrow{(0,0)\stackrel{1}{\rightarrow}(0,0)} (0,0,f,f) \\ (0,0,f,f) & \stackrel{1}{\rightarrow} & (0,g,0,h) & \end{array} . \tag{5.6}$$

In each intermediate result, only the zero differences are predicted. The differences denoted with the letters a to h can take any non-zero value. The differential has a mirror image with the same probability:

$$\begin{array}{llll} (0,a,0,b) & \stackrel{2^{-16}}{\rightarrow} & (0,c,0,c) & \xrightarrow{(0,0)\stackrel{1}{\rightarrow}(0,0)} (0,0,c,c) \\ (0,0,c,c) & \stackrel{1}{\rightarrow} & (0,0,d,e) & \xrightarrow{(d,e)\stackrel{2^{-32}}{\rightarrow}(e,d)} (d,0,e,0) \\ (d,0,e,0) & \stackrel{2^{-16}}{\rightarrow} & (f,0,f,0) & \xrightarrow{(0,0)\stackrel{1}{\rightarrow}(0,0)} (f,f,0,0) \\ (f,f,0,0) & \stackrel{1}{\rightarrow} & (g,0,h,0) & \end{array} . \tag{5.7}$$

To estimate the unconditional probability of the differential, all operations are assumed to have independent inputs. The estimated unconditional probability of the truncated differential is 2^{-64}.

Determination of the distribution of the conditional probability of the differential requires too much computational power. Therefore, the analysis is applied to reduced versions of IDEA. Firstly IDEA(16) is analysed. This cipher uses the same operations as IDEA, but operates on four nibbles (four bit quantities) instead of four 16-bit words [67]. For this cipher, the estimated unconditional probability of (5.6) becomes 2^{-16}. The conditional probability of the differential is determined by encrypting the 2^{16} different plaintexts under each of the 2^{32} possible keys. The resulting distribution is shown in Table 5.5. The main cause for the non-uniformity in the distribution is the second round of the differential, where a difference (d,e) in the inputs to the MA-structure must result in difference (e,d) in the outputs of the MA-structure. The correct value of the unconditional probability of the differential can be calculated using (5.1); it is equal to $2^{-16.5}$, which is slightly less than the first estimate.

For IDEA(32), operating on four bytes, it is no longer possible to determine the conditional probability for all keys. The conditional probability distribution of Table 5.6 is an estimate, based on the encryption of the 2^{32} possible plaintexts under 160 randomly selected key values. Based on these results, the

Probability					#Keys/All keys
		0			13 %
0	$<$	p	$\leq$	2^{-18}	12 %
2^{-18}	$<$	p	$\leq$	2^{-17}	21 %
2^{-17}	$<$	p	$\leq$	2^{-16}	30 %
2^{-16}	$<$	p	$\leq$	2^{-15}	14 %
2^{-15}	$<$	p	$\leq$	1	10 %

Table 5.5: Distribution of the conditional probability of the differential (5.6) for IDEA(16).

unconditional probability is estimated to be $2^{-32.7}$, which should be compared with the 2^{-32} that results from the first estimate.

Probability					#Keys/All keys
0	$\leq$	p	$\leq$	2^{-35}	14 %
2^{-35}	$<$	p	$\leq$	$2^{-33.0}$	10 %
$2^{-33.0}$	$<$	p	$\leq$	$2^{-32.5}$	31 %
$2^{-32.5}$	$<$	p	$\leq$	1	45 %

Table 5.6: Distribution of the conditional probability of the differential (5.6) for IDEA(32).

It can be expected that the behavior of IDEA can be extrapolated from these results.

Using the Truncated Differential

The attack uses *structures* of chosen plaintexts. A structure consists of 2^{32} texts: x_1^0 and x_3^0 are fixed, x_0^0 and x_2^0 take on all possible values. From such a structure it is possible to generate $2^{32} \cdot (2^{32}-1)/2 \approx 2^{63}$ pairs that have the correct truncated input exor for (5.6). Filtering is possible since the differential requires that $x_0^{4\prime} = x_2^{4\prime} = 0$. On average, only one out of 2^{32} pairs will survive this test, or $f = 2^{-32}$.

Each surviving pair suggests values for two 16-bit words of the first round key. A value (z_0^0, z_2^0) is suggested if

$$(x_0^0 \odot z_0^0) \oplus (x_0^{0*} \odot z_0^0) = (x_2^0 + z_2^0) \oplus (x_2^{0*} + z_2^0)\,. \tag{5.8}$$

The expected number of suggested values (z_0^0, z_2^0) by a pair can be estimated to be 2^{16}. Each pair also suggests values for two 16-bit words of the key in the

output transformation. A value (z_1^3, z_3^3) is suggested if

$$(x_1^4 \odot (z_1^3)^{-1}) \oplus (x_1^{4^*} \odot (z_1^3)^{-1}) = (x_3^4 - z_3^3) \oplus (x_3^{4^*} - Z_3^3)\,. \tag{5.9}$$

In total, each pair that survives the filtering is expected to suggest 2^{32} 64-bit key values, or s, the probability for a wrong key to be suggested, equals $2^{32}/2^{64} = 2^{-32}$ (cf. Section 3.2.2). Since $f \cdot s = 2^{-64}$ and every structure contains 2^{63} pairs, it can be expected that every structure will suggest 2^{63} keys and every value of the key will on average be suggested 0.5 times per structure used.

An important side remark can be made about the number of key bits at a time the attack has to search for. The key scheduling of IDEA consists of a simple rotation. This causes an overlap of 14 bits between (z_0^0, z_2^0) and (z_1^3, z_3^3). Furthermore, because of the absence of a carry bit after the highest order bit of the modular addition, key values z_2^0 and z_1^3 that differ only in the highest order bits are indistinguishable. These two observations are very important to reduce the memory requirements of the attack. Instead of 64, only 48 key bits are searched simultaneously, reducing the number of counters by a factor of 2^{16}. The actual attack has only been implemented for the reduced versions of IDEA previously defined. The key schedule for these reduced versions was adapted to produce an overlap of relatively as many key bits. (For IDEA(32) and IDEA(16), seven and three bits overlap, respectively.) This means that in these cases only 23 bit and 11 bit key values are searched for. To find other key bits, a similar attack with the second differential (5.7) can be performed.

Exploitation of the Key Dependence

The signal-to-noise ratio of the attack can be calculated as:

$$S/N = \frac{p}{f \cdot s} = \frac{p}{2^{-64}}.$$

Previous it was believed [10] that a differential attack can only be successful if $S/N > 1$. However, in Section 3.2 it was already mentioned that a differential attack can also find the used key if $S/N < 1$, provided that the signal-to-noise ratio is calculated with p equal to the probability of the maximal differential.

From the filtering, and from the Equations (5.8) and (5.9), it is clear that the maximal differential only specifies the values a, b, c, f, g and h. The maximal

differential containing (5.6) is then given by:

$$\begin{array}{cccccc} (a,0,b,0) & \overset{2^{-16}}{\rightarrow} & (c,0,c,0) & \xrightarrow{(0,0)\overset{1}{\rightarrow}(0,0)} & (c,c,0,0) \\ (c,c,0,0) & \overset{1}{\rightarrow} & (d,e,0,0) & \xrightarrow{(d,e)\rightarrow(\cdot,\cdot)} & (\cdot,\cdot,\cdot,\cdot) \\ (\cdot,\cdot,\cdot,\cdot) & \rightarrow & (\cdot,\cdot,\cdot,\cdot) & \xrightarrow{(\cdot,\cdot)\rightarrow(\cdot,\cdot)} & (0,0,f,f) \\ (0,0,f,f) & \overset{1}{\rightarrow} & (0,g,0,h) & & \end{array} \quad . \tag{5.10}$$

The problem of how to determine the probability of this maximal differential arises. Further analysis shows, however, that all the pairs that follow the maximal differential (5.10) also follow the truncated differential (5.6). Therefore the probability of the maximal differential equals the probability of (5.6).

Since the unconditional probability of the maximal differential is estimated to be 2^{-64}, the signal-to-noise ration is one, and a differential attack seems impossible. For the case of IDEA it turns out that the conditional probability of the differential strongly depends on the actual key value. Thus for many keys, the signal-to-noise ratio will either be significantly higher than one, or it will be significantly lower than one. Since the cryptanalyst does not know beforehand whether $S/N > 1$ or $S/N < 1$, both the least suggested and the most suggested key value will be outputs of the analysis. The more the conditional probability of the differential deviates from $f \cdot s$, the easier it becomes to recover the key. It is interesting to see that for IDEA(16), about 1 in every 8 possible values of the key result in a zero conditional probability for the used differential.

The numbers in Table 5.5 also indicate that the attack will not work for some classes of keys, namely the classes of keys for which the probabilities are too close to $f \cdot s$.

The relation between number of required plaintexts, the workload of the attack and the fraction of recoverable keys, was determined experimentally for the two reduced versions of IDEA.

IDEA(16): Table 5.7 lists the results of 1000 runs of the attack on IDEA(16) for an increasing number of chosen plaintexts. The key ranking technique [79] was used: the attack was considered successful if the correct key value was among the eight least and eight most suggested values. Thus the attack returns 16 suggestions for 11 bits of the secret key. If all the plaintexts are used, the correct value of the key is among those 16 values in about 67 % of all cases. Note that there are a total of 2^{16} plaintexts of IDEA(16) and that an exhaustive search for the key will take about 2^{32} encryptions. The workload is the estimated number of operations required to perform the attack, measured as the number of encryptions of the cipher. For each pair that survives the filtering process, the 2^4 possible values of the affected keys of each side of (5.8) are tried.

The attack can be speeded up by pre-calculation of a table to avoid the expensive multiplication operation. This table has a size of 2^8 nibbles. Under the assumption that a multiplication takes the equivalent of 3.5 additions, and that an addition, an exclusive-or and a table-lookup take about the same time, the workload becomes about 18 encryptions of IDEA(16) for every pair. In total, the workload is about 2^8 encryptions for every structure. Due to the overlap of key bits in this first round test with the key bits in the output transformation, the second part of the key search, i.e. using (5.9), is much faster than the first and can be ignored in the workload estimation.

#Keys/All keys	# Structures	# Chosen plaintexts	Workload
25%	16	2^{12}	2^{12}
40%	32	2^{13}	2^{13}
51%	64	2^{14}	2^{14}
59%	128	2^{15}	2^{15}
67%	256	2^{16}	2^{16}

Table 5.7: Fraction of recovered keys with an increasing number of chosen plaintexts for the attack on IDEA(16) with 3.5 rounds. (1000 runs of the attack were performed.)

IDEA(32): The attack on IDEA(32) has a much higher workload. Table 5.8 lists the results of 50 runs of the attack, using up to 2^{11} structures. In order to reduce the workload of the tests, 7 bits of the key were assumed to be already known, leaving only 16 key bits to search for. The attack was considered successful if the correct value was among the 12 least and four most suggested values. Thus the attack returns 16 suggestions for 16 bits of the secret key.

IDEA: The above results on reduced versions of IDEA allow an estimate of the complexity of the attack on IDEA to be made. Table 5.7 shows that it is possible to find 25 % and 51 % of the keys using $2^{16\times 3/4}$ and $2^{16\times 7/8}$ chosen plaintexts respectively for IDEA(16). Table 5.8 shows that it is possible to find 28 % and 64 % of the keys using $2^{32\times 3/4}$ and $2^{32\times 13/16}$ chosen plaintexts respectively for IDEA(32). The estimated complexity of the attack on IDEA is given in Table 5.9, using the results on the reduced versions. For at least one key out of every hundred, 2^{40} chosen plaintexts and an effort of about 2^{51} encryptions suffice to recover the key. More than 83 % of the keys can be recovered with 2^{56} chosen plaintexts and an effort of about 2^{67} encryptions. Note that an exhaustive key search has an expected workload of 2^{127} encryptions.

#Keys/All keys	# Structures	# Chosen plaintexts
1 %	16	2^{20}
7 %	64	2^{22}
15 %	128	2^{23}
31 %	256	2^{24}
54 %	512	2^{25}
65 %	1024	2^{26}
83 %	2048	2^{27}

Table 5.8: Fraction of recovered keys with an increasing number of chosen plaintexts for the attack on IDEA(32) with 3.5 rounds. (50 runs of the attack were performed.)

#Keys/All keys	# Chosen plaintexts	Workload
> 1 %	$2^8 \cdot 2^{32}$	2^{51}
> 31 %	$2^{16} \cdot 2^{32}$	2^{59}
> 83 %	$2^{24} \cdot 2^{32}$	2^{67}

Table 5.9: Estimated fraction of recovered keys with an increasing number of chosen plaintexts for the attack on IDEA with 3.5 rounds.

Finding Additional Key Bits

The attack outlined above finds 48 bits of the 128-bit key of IDEA. However, once these key bits have been found, a similar attack using the second truncated differential can be performed. As noted earlier, the key-dependency of the probability of the first differential comes mostly from the second round of the differentials. Since the second round is the same for the two differentials, it can be expected that for a fixed key, the probabilities of the two differentials are very close. After doing the attack with the second differential all 64 key bits in the beginning of the first round and all 64 key bits of the output transformation are obtained. Subsequently, similar attacks can be done on a further reduced version of IDEA with a negligible complexity.

5.3 Application to MAA

In [107], B. Preneel and P.C. van Oorschot presented a generic attack on MAC algorithms; in [108] this attack was applied to MAA, and a class of weak keys for MAA was identified. This section shows that some keys exhibit clusters of collisions, which is an undesirable property. In theory, a collision cluster could be used to recover the used key. These results on MAA have been published in [106].

5.3.1 The Message Authenticator Algorithm MAA

MAA is one of the primary MAC algorithms used historically, the other one being CBC-MAC [49, 51]. The Message Authenticator Algorithm (MAA) is an ISO standard [49] which dates back to 1984 [31]. It is currently being used by several large financial institutions. While it was originally designed for use on mainframe computers, it is very fast on present PCs and workstations (about the same speed as SHA-1 [41]). A complete description of MAA can be found in [32]. All variables in the description are 32-bit words. The 64-bit key is split into two 32-bit words: j and k. The algorithm consists of three parts. During the *prelude* the 64-bit key is used to initialise the *chaining variables* x^0 and y^0 and to calculate four parameters v^0, w, t, and s. The input m to MAA is of arbitrary length; it is divided into q 32-bit words denoted by x^1 through x^q. The *main loop* takes the ith 32-bit message word m^i, the chaining variables x^{i-1} and y^{i-1}, and the parameters v and w as input, and produces as output the updated chaining variables x^i and y^i. The last part is the *coda*; it simply adds s and t as final message blocks ($m^{q+1} = s$ and $m^{q+2} = t$) and computes the 32-bit MAC as $\mathrm{MAA}[j,k](m) = x^{q+2} \oplus y^{q+2}$.

The ith iteration of the main loop ($1 \leq i \leq q+2$) performs the following operations:

$$
\begin{aligned}
v^i &= \texttt{rol}(v^{i-1});\\
x^i &= (x^{i-1} \oplus m^i) \otimes_1 M_1((v^i \oplus w) + (y^{i-1} \oplus m^i));\\
y^i &= (y^{i-1} \oplus m^i) \otimes_2 M_2((v^i \oplus w) + (x^{i-1} \oplus m^i));
\end{aligned}
\quad .
$$

Here $\otimes_1$ denotes multiplication modulo $2^{32}-1$, $\otimes_2$ denotes multiplication modulo $2^{32}-2$, $+$ is addition modulo 2^{32}, and $\oplus$ is bitwise exor. Functions $M_1(\cdot)$ and $M_2(\cdot)$ are masking operations that each fix eight bits (four to zero and four to one):

$$M_1(x) = (x \vee A) \wedge C \qquad\qquad M_2(x) = (x \vee B) \wedge D\,,$$

where $A = \texttt{02040801}_\texttt{x}$, $B = \texttt{00804021}_\texttt{x}$, $C = \texttt{BFEF7FDF}_\texttt{x}$, and $D = \texttt{7DFEFBFF}_\texttt{x}$. In the following, $v^i \oplus w$ is denoted by z^i (or z, when the superscript is clear from the context).

During the prelude the six variables x^0, y^0, v^0, w, s, and t are checked for bytes that are equal to $\texttt{00}_\texttt{x}$ or $\texttt{FF}_\texttt{x}$. The prelude calls a special procedure to alter these bytes, the BYT procedure, because they are considered to be worth avoiding. If no $\texttt{00}_\texttt{x}$ or $\texttt{FF}_\texttt{x}$ values are encountered, x^0, v^0 and s depend only on j, whereas y^0, w and t depend only on k.

ISO 8731 [49] limits the size of the messages to $4 \cdot 10^6$ bytes (≈ 3.8 Mbyte). Also, the standard defines a special mode for messages longer than 1024 bytes (256 blocks). In this mode, MAA is applied to the first 1024 bytes, and the corresponding 4-byte MAC is concatenated to the next 1024 bytes of the message to form the new input of MAA. This procedure is repeated with the next 1024-byte block, until the end of the message is reached.

5.3.2 Known classes of weak keys

In [108] two classes of weak keys for MAA have been identified. The first class of weak keys are external keys which result in an internal key v^0 of rotational period < 32. For such keys, rotating v^0 over 2, 4, 8, or 16 positions will yield v^0 again (there are no keys v^0 of period 1 since the all zero and all one values are eliminated). There are respectively 2, 14, 254, and 64516 values of v^0 for which this holds. Exhaustive examination can be used to determine which values of the first 32-bit word of the input key (j) yield such values of v^0. The the number of weak keys is independent of the second input key word (k), and is thus 2^{32} times larger. If v^0 has period r then a forgery attack can be mounted, requiring $r \cdot (2^{31}-2)$ zero blocks. Verifying the forgery allows a cryptanalyst to obtain information on v^0, which is undesirable since it leaks partial key bits. It is relatively easy to detect whether a key is weak, leading to a key recovery attack on these keys [108].

Proposition 5.1 *For MAA, one can detect, using 2^{27} chosen messages of about 1 Kbyte each, whether a key belongs to a subclass of 2^{48} weak keys.*

The second class of weak keys allows the size of the message required for another existential forgery attack described in [108] to be reduced. These weak keys are not a problem if the long message mode is used.

5.3.3 Collision Clusters for MAA

Consider two different messages m and m^* of length q. If $x^q = x^{q*}$ and $y^q = y^{q*}$ an *internal collision* is said to have occurred. If no internal collision occurs, but nonetheless $\mathrm{MAA}(m) = \mathrm{MAA}(m^*)$, then an *external collision* is said to have occurred. If the two messages have the first $q-1$ blocks in common, then $x^{q-1} = x^{q-1*}$ and $y^{q-1} = y^{q-1*}$. Denote the difference in the last message block by $d = m^{q*} \oplus m^q$. The two messages form an internal collision iff

$$\begin{aligned}(x^{q-1} \oplus m^q) \otimes_1 M_1((v^q \oplus w) + (y^{q-1} \oplus m^q)) = \\ (x^{q-1} \oplus m^q \oplus d) \otimes_1 M_1((v^q \oplus w) + (y^{q-1} \oplus m^q \oplus d)) \qquad (5.11a)\\ (y^{q-1} \oplus m^q) \otimes_2 M_2((v^q \oplus w) + (x^{q-1} \oplus m^q)) = \\ (y^{q-1} \oplus m^q \oplus d) \otimes_2 M_2((v^q \oplus w) + (x^{q-1} \oplus m^q \oplus d))\,. \qquad (5.11b)\end{aligned}$$

For a given value of d,

$$\Pr\left((x^q, y^q) = (x^{q*}, y^{q*})\right) \approx 2^{-64}\,.$$

The expected number of internal collisions among the 2^{32} messages that have the first $q-1$ blocks in common and take on all possible values for the last block, is equal to $1/2$. In this section the existence of large classes of keys ($\geq 2^{33}$ elements) for which there exists a value of d such that

$$2^{-21} \leq \Pr\left((x^q, y^q) = (x^{q*}, y^{q*})\right) \leq 2^{-13}\,,$$

will be demonstrated. For these keys, between 2^{11} and 2^{19} collisions occur between all q-block messages that differ only in the last block. All collisions have the same value of d. In this context a key is called 'weak' if it has a number of collisions that is substantially larger than two. Several classes of weak keys can be distinguished. The first class described here in detail is the most easy to find. Afterwards other types of weak keys are mentioned.

This property allows the detection and subsequent recovery of weak keys using about 2^{23} chosen texts. Also, a small subset of the messages can be forged using this property. Precisely which messages are forgeable depends on the actual key value.

Simple Weak Keys

In the following, ϵ^i denotes $x^i \oplus y^i$. In the analysis, the superscript of z, ϵ, x, y, v and w is omitted. A sufficient set of conditions for a collision is:

$$\begin{aligned} x \oplus m &= M_1(z + (x \oplus m \oplus \epsilon \oplus d)) && \text{(5.12a)} \\ x \oplus m \oplus d &= M_1(z + (x \oplus m \oplus \epsilon)) && \text{(5.12b)} \\ x \oplus m \oplus \epsilon &= M_2(z + (x \oplus m \oplus d)) && \text{(5.12c)} \\ x \oplus m \oplus \epsilon \oplus d &= M_2(z + (x \oplus m))\,. && \text{(5.12d)} \end{aligned}$$

The Equations (5.12) can be written at bit level. In this analysis, the bits of the words are numbered from the right to the left (the least significant bits get number 0, the most significant bits number 31). The carry bits of addition modulo 2^{32} ('+') into bit i are denoted by u_i, v_i, w_i, and t_i. The equations for the carry bits become (with $u_0 = v_0 = w_0 = t_0 = 0$):

$$\begin{aligned} u_{i+1} &= u_i(z_i \oplus x_i \oplus m_i \oplus \epsilon_i \oplus d_i) \oplus z_i(x_i \oplus m_i \oplus \epsilon_i \oplus d_i) \\ v_{i+1} &= v_i(z_i \oplus x_i \oplus m_i \oplus \epsilon_i) \oplus z_i(x_i \oplus m_i \oplus \epsilon_i) \\ w_{i+1} &= w_i(z_i \oplus x_i \oplus m_i \oplus d_i) \oplus z_i(x_i \oplus m_i \oplus d_i) \\ t_{i+1} &= t_i(z_i \oplus x_i \oplus m_i) \oplus z_i(x_i \oplus m_i)\,. \end{aligned}$$

The masking of bit i determines the rest of the bit level equations. Four cases can be distinguished:

0. When both M_1 and M_2 leave bit i unchanged:

$$u_i = v_i = w_i = t_i = z_i \oplus \epsilon_i \oplus d_i\,.$$

1. When M_1 sets or clears bit i, but M_2 does not:

$$\begin{aligned} w_i = t_i &= z_i \oplus \epsilon_i \\ d_i &= 0 \\ x_i \oplus m_i &= M_{1i}\,. \end{aligned}$$

2. When M_2 sets or clears bit i, but M_1 does not:

$$\begin{aligned} u_i = v_i &= z_i \oplus \epsilon_i \\ d_i &= 0 \\ x_i \oplus m_i \oplus \epsilon_i &= M_{2i}\,. \end{aligned}$$

3. When both M_1 and M_2 determine bit i:

$$\begin{aligned} d_i &= 0 \\ \epsilon_i &= M_{1i} \oplus M_{2i} \\ x_i \oplus m_i &= M_{1i}\,. \end{aligned}$$

The next step is to write down these equations for the 32 bits. The result is three sets of equations:

- A set of conditions on z and ϵ: these conditions determine the class of weak keys. The class of keys where z^1 and ϵ^0 satisfy the equations is called the base class of weak keys. If z^1 and ϵ^0 do not satisfy the equations then it is still possible that z^i and ϵ^{i-1} $(1 < i)$ do. These keys can be attacked by using messages consisting of $i+1$ blocks, where the first i blocks are common. This means that by doing more work, the probability of success can be enhanced. In the generic case there are 32 different values for z (because of the rotation operation). By varying the choice of common blocks, all values of ϵ can be created (but the values cannot be controlled).
- Equations that give d as a function of z and ϵ.
- Conditions on $x \oplus m$ that determine for given x a vector space of messages.

These equations were solved for a reduced version of MAA that operates on 14-bit words instead of 32-bit words. For this version of MAA there are about $2^{14.6}$ keys in the base weak key class. Each weak key has an associated difference d and an associated vector space. Two messages that have difference d will produce a collision with a probability of 2^{-4}, 2^{-5}, or 2^{-6}. The probability depends on the size of the vector space of messages associated with the weak key. The expected number of basic weak keys for the full MAA is 2^{33}. For each key there exists an associated difference d such that between 2^{11} and 2^{19} block messages with this difference will collide.

To demonstrate the existence of weak keys, a key that produces a z^2 satisfying the equations was searched for. Afterwards, a first message block m^1 was searched for such that ϵ^1 is also a solution.

Example 5.3 *Let* $j = \texttt{e8813bb2}_\texttt{x}$, $k = \texttt{45cfb69c}_\texttt{x}$, *and* $m^1 = \texttt{56e2}_\texttt{x}$. *There exist* 2^{18} *pairs of two block messages with the first message block equal to* m^1 *and the second message blocks differing by* $d = \texttt{1c081098}_\texttt{x}$, *that produce an internal collision.*

The Use of a Collision Cluster in an Attack

The existence of a collision cluster of size 2^p can be used in a differential attack to recover the key. Two messages with difference d will produce a collision with probability 2^{p-32}. The first step is to recover d. Thirteen bits of d are known to be zero. The message space can be divided into 2^{13} subspaces, with constant values for these 13 bits. The collision cluster will be situated in one of these subspaces. In this subspace the difference will have a collision probability equal to 2^{p-19}. Since it is not known beforehand which subspace is the correct one,

the attack has to be repeated for all of them. The 19 unknown bits of d can be found by encrypting randomly selected messages from the same subspace. With 2^n texts, 2^{2n-1} pairs can be generated. A pair with the correct difference d will generate a collision with probability 2^{p-19}. A collision will occur with high probability when

$$\begin{aligned} 2^{2n-1} &\geq \left(2^{-19} \times 2^{p-19}\right)^{-1} \\ &\Updownarrow \\ n &\geq \frac{39-p}{2}. \end{aligned}$$

For the largest clusters, $p = 19$, and only $2^{13} \cdot 2^{10} = 2^{23}$ texts are required. When $p = 11$, 2^{27} texts are required. Once d is known the bit equations can be used to determine bits of $x^0, y^0, x^0 \oplus y^0$, and z^0. Off-line built tables (with size 2^{32}) of $x^0(j), v^1(j), y^0(k)$, and $w(k)$ allow j and k to be determined. The following proposition summarises this result.

Proposition 5.2 *A key that has an associated difference with a collision cluster of size 2^p, can be detected with a differential attack using $2^{13} \times 2^{(39-p)/2} = 2^{(65-p)/2}$ chosen messages. The value of the difference and the colliding messages gives sufficient information for recovery of the key with a lookup table.*

The collision cluster can be used for forging messages that lie in the subspace of the cluster. As explained above, a message m^* will produce the same MAC as the message m with probability $2^{p-19} \cdot 2^{-19} = 2^{p-38}$. This probability varies between 2^{-19} and 2^{-27}. However, which of the 2^{13} subspaces is forgeable, depends on the actual key value.

Involved Weak Keys

The set of collisions with the same difference d that occurs for weak keys, can be seen as a 'burst' of collisions. Knowledge of one collision enables a cryptanalyst to very easily create the whole vector space of collisions. The interaction between masking and the two modular multiplications causes many other 'bursts' of collisions. Below, one other example is presented.

Example 5.4 *Consider a reduced version of MAA, operating on 14-bit words instead of 32-bit words. Let $j = \mathtt{72d}_\mathtt{x}$, $k = \mathtt{3a39}_\mathtt{x}$. There exist 2^9 one block message pairs $(m, m \oplus \mathtt{31d8}_\mathtt{x})$ that produce a collision.*

To explain the phenomenon, the example is analysed in some detail. Equation (5.11a) gives:

$$(\mathtt{1dce}_\mathtt{x} \oplus m) \otimes_1 M_1(1 + (\mathtt{13cb}_\mathtt{x} \oplus m)) = (\mathtt{2c16}_\mathtt{x} \oplus m) \otimes_1 M_1(1 + (\mathtt{2233}_\mathtt{x} \oplus m)). \tag{5.13}$$

The solution $m = \mathtt{c22}_\mathtt{x}$ is called the base solution. Filling in gives:

$$\begin{aligned} \mathtt{11ec}_\mathtt{x} \times M_1(1 + \mathtt{1fe9}_\mathtt{x}) &= \mathtt{11ec}_\mathtt{x} \times \mathtt{1fcb}_\mathtt{x} = \mathtt{130b}_\mathtt{x} + \mathtt{8e7}_\mathtt{x} \times (2^{14} - 1) \\ \mathtt{2034}_\mathtt{x} \times M_1(1 + \mathtt{2e11}_\mathtt{x}) &= \mathtt{2034}_\mathtt{x} \times \mathtt{2e13}_\mathtt{x} = \mathtt{130b}_\mathtt{x} + \mathtt{172f}_\mathtt{x} \times (2^{14} - 1) \end{aligned}$$

Now consider the slightly modified Equation (5.13):

$$(\mathtt{11ec}_\mathtt{x} + m') \otimes_1 (\mathtt{1fcb}_\mathtt{x} + m'') = (\mathtt{2034}_\mathtt{x} - m'') \otimes_1 (\mathtt{2e13}_\mathtt{x} - m')\,, \qquad (5.14)$$

which can be rewritten as

$$\mathtt{11ec}_\mathtt{x} \times \mathtt{1fcb}_\mathtt{x} - \mathtt{2034}_\mathtt{x} \times \mathtt{2e13}_\mathtt{x} + (\mathtt{1fcb}_\mathtt{x} + \mathtt{2034}_\mathtt{x})m' + (\mathtt{11ec}_\mathtt{x} + \mathtt{2e13}_\mathtt{x})m'' = 0 \bmod (2^{14} - 1)\,.$$

The base solution corresponds to $m' = m'' = 0$. Since $\mathtt{11ec}_\mathtt{x} + \mathtt{2e13}_\mathtt{x} = 2^{14} - 1$ and $\mathtt{1fcb}_\mathtt{x} + \mathtt{2034}_\mathtt{x} = 2^{14} - 1$, all values of m', m'' will satisfy Equation (5.14). It can be concluded that every m^* for which an m' and an m'' can be found such that

$$\begin{aligned} \mathtt{1dce}_\mathtt{x} \oplus m^* &= \mathtt{11ce}_\mathtt{x} + m' \\ M_1(1 + (\mathtt{13cb}_\mathtt{x} \oplus m^*)) &= \mathtt{1fcb}_\mathtt{x} + m'' \\ \mathtt{2c16}_\mathtt{x} \oplus m^* &= \mathtt{2034}_\mathtt{x} - m'' \\ M_1(1 + (\mathtt{2233}_\mathtt{x} \oplus m^*)) &= \mathtt{2e13}_\mathtt{x} - m'\,, \end{aligned}$$

will be a solution of (5.13). A similar equivalence between addition and the interaction of M_2, z, and exor has to exist for Equation (5.11b). The example shows that there exist keys and m-values in practice with a large cluster of m^* values.

Conclusion

For some values of z it is possible to find collision clusters that can be used in a differential attack. For the 'weakest' values of z the attack requires only 2^{23} chosen texts to recover the key. The previous best known attack on MAA was a forgery attack which requires about 2^{24} chosen texts with 250 trailing blocks [108]. The problem could be eliminated in several ways: applying the *BYT* function to z, updating z in a more complex way or having different values of z in the equations for x^i and y^i.

5.4 Conclusions

In this chapter it was demonstrated that the hypothesis of stochastic equivalence [69] does not always hold. The consequence is the existence of differential characteristics with key dependent probabilities and key dependent input-output

correlations. The keys for which these relations lead to an attack on the cipher, are called 'weak keys'. By considering key dependent relations and characteristics it is possible to mount attacks that perform better than the previously known attacks.

Two attacks on reduced versions of IDEA were presented. The first attack works on IDEA reduced to three rounds and recovers the key by using a set of linear relations. The relations of the set are chosen such that, for each key, at least one relation has high correlation values. The attack requires at most 2^{29} chosen plaintexts and has a workload of about 2^{44} encryptions. The second attack uses truncated differentials and works on IDEA reduced to three rounds, followed by the output transformation. For 1% of the keys, the attack requires only 2^{40} chosen plaintexts and has a workload of 2^{51} encryptions. By using more plaintexts the attack will recover more keys. Both attacks demonstrate that successful attacks can be mounted, even if the resistance of the cipher is good *on average*. As stated in Section 3.4, the average probability of a characteristic or a relation is only of approximate value. The attacks have been published in [14].

The analysis of MAA leads to a similar conclusion. The value of the key dependent parameter z is essential for the security of the algorithm. Weak values of z lead to collision clusters that can be used in a key recovery attack. A new key recovery attack has been presented which requires fewer chosen plaintexts than the best previously known attack on MAA, which is only a forgery attack. The results on MAA have been published in [106].

Chapter 6

Non-Surjective Attack

If it 's provably secure,
it 's probably not.
L.R. Knudsen

An important design criterion for block ciphers is performance. In order to build on the experience gathered from the cryptanalysis of DES, most designers preserve the structure of a Feistel network, but suggest new structures for the round function that exploit in a more efficient way the present day computer architectures. In this way they try to achieve a better trade-off between security and speed. Examples of such block ciphers are FEAL [88], LOKI91 [17], Blowfish [121], and the CAST cipher family [2, 3, 48]. One approach is the use of large highly nonlinear (or random) S-boxes. This allows the designers to reduce the number of rounds and optimise the speed of the algorithm, while maintaining or improving the resistance against differential and linear cryptanalysis. However, reducing the number of rounds may introduce new vulnerabilities.

This chapter focuses on a new attack on Feistel ciphers that exploits a weakness that is introduced by the use of non-surjective or, more generally, non-uniform round functions. The attack demonstrates that the round function of a Feistel cipher with six to eight rounds needs to be surjective and sufficiently uniform. The attack is explained in Section 6.1, and applied in Section 6.2 to some members of the CAST family and LOKI91. These results have been published in [112, 113].

6.1 General Principle

Firstly the notation is introduced. Next, the attack is presented, followed by an extension. Afterwards, a chosen plaintext variant is explained.

6.1.1 Notation

Consider a Feistel cipher with R rounds (R even and $R \geq 4$), operating on text blocks of width l. The values s^r, t^r and k^r, $r = 0, \ldots R$ are defined in Section 2.1.1. A function is called *unbalanced* or *non-uniform* if it does not take all the outputs in its range equally often. Define β_τ as:

$$\beta_\tau(s^0, t^0, k) = \bigoplus_{r=1}^{\tau/2} F(k^{2r} \oplus t^{2r-1}), \quad \tau \text{ even and} \geq 2\,. \tag{6.1}$$

Under the assumption of independent and uniformly distributed round keys, it holds that for an unbalanced round function F, the sum β_τ will be unbalanced. It can be expected that this also holds for most key schedulings. Setting τ equal to R, (6.1) becomes:

$$\beta_R(s^0, t^0, k) = \bigoplus_{r=1}^{R/2} F(k^{2r} \oplus t^{2r-1}) = t^0 \oplus s^R\,. \tag{6.2}$$

If not all the values of β_R have the same probability, a cryptanalyst can gather statistical information about the plaintext by analysing the ciphertext. In a known plaintext setting, the value of β_{R-2} can provide information about the key, as will be explained in the following section.

6.1.2 Basic Attack

In the following, the concept of a random function is required. A random function is defined as a function that is selected at random from the set of all possible mappings from the domain to the range of the function.

Taking the last round out of the sum, (6.2) becomes

$$\beta_{R-2}(s^0, t^0, k) = \bigoplus_{r=1}^{R/2-1} F(k^{2r} \oplus t^{2r-1}) = t^0 \oplus s^R \oplus F(k^R \oplus t^R)\,. \tag{6.3}$$

Random non-surjective round functions F will result in a non-surjective β_{R-2} for small R. This is quantified in the following lemmas.

Lemma 6.1 *A random function with equal input and output size takes on average a fraction of* $1 - e^{-1}$ *of the possible outputs.*

Proof: Denote the number of inputs and outputs by p. The problem can be described in terms of balls and bins. Every input corresponds to a ball thrown into a randomly chosen bin (there are 2^p balls and bins). Of interest is the

number of empty bins after all the balls have been thrown. The probability of a bin remaining empty after one ball is thrown is given by $1 - 2^{-p}$. After 2^p balls, this probability is $(1 - 2^{-p})^{2^p}$. This value quickly approximates e^{-1} if p is large (e.g., for $p = 16$ the error is less than .001 percent). ∎

Lemma 6.2 *Denote by f the fraction of p-bit vectors that are possible outputs of the round function, and by f_t the fraction of possible values for β_t. If the round function is a random function and has independent inputs in different rounds then:*

$$f_2 = f \quad \textit{and} \quad f_{\tau+2} = 1 - (1 - f_\tau \cdot f)^{2^p} , \quad \tau \geq 2 . \tag{6.4}$$

Proof: From the definition of β_τ:

$$\beta_{\tau+2} = \beta_\tau \oplus F(k^{\tau+2} \oplus t^{\tau+1}) .$$

Variable $\beta_{\tau+2}$ can take the value x if there exists at least one y such that y is a possible value for β_τ and $y \oplus x$ is a possible output of F. Conversely, x is an impossible value for $\beta_{\tau+2}$ if and only if there exists no such y. For a random round function with independent inputs, the product rule can be applied to obtain

$$1 - f_{\tau+2} = (1 - f_\tau \cdot f)^{2^p} ,$$

from which (6.4) follows. ∎

A non-surjective β_{R-2} makes the following attack possible. For all k^R, calculate the right hand side of (6.3) by use of the known plaintext (s^0, t^0) and the ciphertext (s^R, t^R). Check whether this is a possible value for β_{R-2}. Wrong key guesses will eventually produce an impossible value for β_{R-2}. If there are 2^κ possible round keys k^R, on average $-\kappa / \log_2(f_{R-2})$ plaintext/ciphertext pairs are required to determine the right value of k^R. Indeed, the number m of known plaintexts can be solved from the equation

$$2^\kappa \cdot f_{R-2}^m = 1 .$$

The work factor of the attack can be calculated as follows. Start with 2^κ possible keys and verify for each key whether it could produce the first known plaintext-ciphertext pair. A fraction of f_{R-2} of the keys survives this test. For these $2^\kappa f_{R-2}$ keys, test whether they could produce the second plaintext-ciphertext pair, etc. This leads to a work factor of

$$\sum_{j=0}^{m-1} 2^\kappa \cdot f_{R-2}^j \approx \frac{2^\kappa}{1 - f_{R-2}} . \tag{6.5}$$

For small values of κ, or when consecutive round keys are strongly related, several round keys can be searched for at once. In this way, f_{R-j} can be used (where $j > 2$) instead of f_{R-2}, which will make the attack more effective, as can be seen from (6.4) and (6.5). In general, let $\kappa(j)$ denote the number of key bits that have to guessed if j rounds are peeled off. Let $w(R-j)$ denote the work factor of the attack for one key guess, and $\kappa(R)$ the total number of key bits. The maximum value for j for which the attack is less effort than an exhaustive key search, is then determined by

$$2^{\kappa(j)} w(R-j) < 2^{\kappa(R)} ,$$

or

$$\kappa(j) + \log_2(w(R-j)) < \kappa(R) . \tag{6.6}$$

6.1.3 Statistical Attack

Equation (6.4) shows that for larger values of R, f_{R-2} approaches 1 very quickly, but β_{R-2} will not be uniformly distributed: all outputs are possible, but they do not occur with the same probability. For still larger values of R, β_{R-2} becomes close to a 'random function,' which should be a design goal. The attack can be modified to deal with surjective but non-uniform functions β_{R-2}. Firstly the extension of the basic attack is described. The computation of the distribution of β_{R-2} and the expected number of known plaintexts is then explained.

Outline

The first step consists of computing the relative probabilities of each value of β_{R-2}. Then the right hand side of (6.3) is computed for every k^R and for every known plaintext-ciphertext pair. It is then possible to calculate the a posteriori probability of the key candidates.

By Bayes' rule, the probability $\Pr(k^R|t^0, s^R, t^R)$ that k^R is the right key, given t^0, s^R, and t^R, can be expressed as:

$$\Pr(k^R|t^0, s^R, t^R) = \frac{\Pr(k^R)\Pr(t^0, s^R, t^R|k^R)}{\Pr(t^0, s^R, t^R)} = \frac{\Pr(k^R)\Pr(\beta_{R-2})}{\Pr(\beta_R)} .$$

Denote by $\Pr^i(k^R)$, the a posteriori probability that k^R is the right key after the processing of the ith known plaintext ($\Pr^0(k^R) = 1/2^\kappa$). Then

$$\Pr{}^i(k^R) = \frac{\Pr^{i-1}(k^R)\Pr(\beta^i_{R-2})}{\Pr(\beta^i_R)} = \frac{1}{2^\kappa} \prod_{j=1}^{i} \frac{\Pr(\beta^j_{R-2})}{\Pr(\beta^j_R)} . \tag{6.7}$$

This expression can be evaluated for each key candidate; it assigns to each round key a probability that can be used to rank the keys according to decreasing probability. In a practical implementation the logarithms of the probabilities can be added, rather than multiplying the values.

Distribution of β_{R-2}

The calculation of the probability of each β_{R-2} turns out to be a non-trivial step. One strategy is to count the occurrences of each β_{R-2} for each possible input of the round functions, but this is infeasible for realistic values of R. A more convenient strategy uses the Walsh transform to compute the distribution of β_{R-2} from the distribution of its components. Suppose the Boolean vector y depends on the Boolean vectors v and w in the following way: $y = v \oplus w$. Denote by $f_k(x)$ the distribution of k, i.e. the number of times that k equals x. The Boolean vector y is equal to x if $v = s$ and $w = s \oplus x$, and this holds for all possible values of s. Therefore

$$f_y(x) = \sum_s f_v(s) f_w(x \oplus s)\,,$$

which means that f_y is the convolution of f_v and f_w. The convolution of functions of Boolean vectors can be calculated by multiplying their Walsh transforms [8]. The Walsh transform of p-bit functions can be computed in $\mathcal{O}(p\,2^p)$ integer operations.

To find the distribution of β_{R-2}, first the Walsh transform of the distribution of the round function (or equivalently of β_2) is calculated. This distribution can be obtained from a counting operation, or it can be calculated by a Walsh transform itself if the round function exors the outputs of several S-boxes (e.g., the CAST round function, see Section 6.2.1). Since $(R-2)/2$ round functions contribute to β_{R-2}, the distribution of β_{R-2} is equal to the inverse Walsh transform of the $(R-2)/2$th power of the transformed distribution of β_2.

In this way, a probability for each value of β_{R-2} is obtained. However, to estimate the number of plaintexts, is convenient to have a more compact representation. Define $d(r) = d(\Pr(\beta_{R-2}))$ as the frequency distribution of β_{R-2}, with mean value 2^{-p}. Replacing the variable r by its logarithm $l = \ln(r) = \ln(\Pr(\beta_{R-2}))$, the distribution $d(l)$ is obtained, with mean value $\approx \ln(2^{-p})$. The standard deviation is a measure of the imbalance of β_{R-2}. A large standard deviation means that there are values of β_{R-2} that occur much more, or much less frequently, than on average. In the subsequent sections, $d(l)$ will be called 'the distribution of β_{R-2}.'

Estimation of the Number of Plaintexts

Equation (6.7) enables a cryptanalyst to calculate the a posteriori probability of each round key. If only a small number of known plaintexts are available, it is very unlikely that the right round key has the highest rank, i.e., the largest probability. As the number of plaintexts increases the probability that the right round key gets the highest rank will increase. Now the number m of known plaintexts that are required for the right key to have the highest rank with probability $1/e$ (e is the base of the natural logarithm) will be estimated. Instead of multiplying probabilities in (6.7), the logarithms of the values will be added.

In order to estimate this number, it is examined in detail what happens for a candidate round key $\tilde{k}^R$. For each $\tilde{k}^R$,

$$\tilde{\beta}_{R-2} = \tilde{\beta}_{R-2}(\tilde{k}^R) = t^0 \oplus s^R \oplus F(\tilde{k}^R \oplus t^R)$$

is calculated from the plaintext and the ciphertext and $l = \ln(\Pr(\tilde{\beta}_{R-2}))$ is obtained from the precalculated table and added to $\tilde{k}^R$'s counter. The idea is that for the right round key, values of $\tilde{\beta}_{R-2}$ with a higher (logarithm of) probability, will occur more frequently. Therefore, high values of l will be added to $\tilde{k}^R$'s counter more frequently than low values of l. In contrast, for a wrong $\tilde{k}^R$, there is no correlation between the probability of $\tilde{\beta}_{R-2}$ and the value added to the counter; the increment is chosen more or less at random from the distribution of β_{R-2}. After many plaintexts, the difference between a right and a wrong key is likely to become clear.

If all $\tilde{\beta}_{R-2}$'s occur with the same probability then this whole operation corresponds to adding stochastic variables with a distribution equal to the distribution of β_{R-2}. If $\tilde{k}^R$ is not the right round key then this is actually what happens, because choosing the wrong round key can be thought of as adding an extra round instead of peeling one off [46]. This means that in fact $\tilde{\beta}_{R-2} = \beta_{R+2}$, which is almost uniform compared to β_{R-2}, since the imbalance is strongly reduced as the number of rounds increases. However, if $\tilde{k}^R$ is the right round key then $\tilde{\beta}_{R-2} = \beta_{R-2}$, so that values of $\tilde{\beta}_{R-2}$ with a higher probability, and thus with a higher $l = \ln(\Pr(\tilde{\beta}_{R-2}))$, will occur more frequently. This means that the adding operation corresponds to adding a stochastic variable with a similar distribution, but slightly distorted to higher values of l.

For one plaintext, there will be a large overlap between these two distributions, which makes it almost impossible to distinguish between them. However, if the procedure is repeated a number of times, each of the two distributions is convoluted with itself. Let μ_w, σ_w and μ_r, σ_r be the mean and standard deviation of the distributions for a wrong and the right round key respectively. After m plaintexts, the distributions can be approximated by normal distributions. The mean values are multiplied by m, but the standard deviations only

by $\sqrt{m}$, which implies that the distributions will be easier to distinguish. The probability that a counter c_w for a wrong round key is greater than the counter c_r for the right round key after m plaintexts is $\Pr(c_w > c_r) = \Pr((c_r - c_w) < 0)$. The probability that c_r is the largest of all the counters is

$$(1 - \Pr((c_r - c_w) < 0))^{2^p} .$$

If $\Pr((c_r - c_w) < 0) = 2^{-p}$ then this probability equals e^{-1}. The distribution of $c_r - c_w$ has mean $m\mu_r - m\mu_w$ and standard deviation $\sqrt{m\sigma_r^2 + m\sigma_w^2}$. Hence,

$$\begin{aligned}
\Pr((c_r - c_w) < 0) &= 2^{-p} \\
&\Updownarrow \\
\Phi\left(\frac{m(\mu_r - \mu_w)}{\sqrt{m(\sigma_r^2 + \sigma_w^2)}}\right) &= 2^{-p} \\
&\Updownarrow \\
m &= \left(\Phi^{-1}(2^{-p})\frac{\sqrt{\sigma_r^2 + \sigma_w^2}}{\mu_r - \mu_w}\right)^2 ,
\end{aligned}$$

where $\Phi(x) = (1 + \mathrm{erf}(x/\sqrt{2}))/2$ is the integral from $-\infty$ to x of the normal probability density function, and $\Phi^{-1}(x)$ is its inverse. The parameters μ_w and σ_w can be obtained from the distribution of β_{R-2}, but for μ_r and σ_r the 'distorted' distribution for the right round key is required. A straightforward way to obtain an approximation of this distribution is to simulate it. However, the probability that the right round key 's counter is augmented by a value l as the result of one plaintext can also be theoretically calculated.

$$\begin{aligned}
\Pr(l) &= \sum_{\beta_{R-2}} \Pr(l|\beta_{R-2}) \Pr(\beta_{n-2}) \\
&= \sum_{\beta_{R-2}:\ln(\Pr(\beta_{R-2}))=l} \Pr(\beta_{R-2}) \\
&\sim \mathrm{d}(l) \Pr(\beta_{R-2}) \\
&\sim \mathrm{d}(l) \exp(l) ,
\end{aligned}$$

where $\mathrm{d}(l)$ is the distribution of β_{R-2}, as defined in Section 6.1.3. This implies that for the right round key, the distribution of β_{R-2} is multiplied by an exponential function and a constant factor. The parameters μ_r and σ_r can be calculated from this new distribution.

6.1.4 A Chosen Plaintext Variant

The principles of the known plaintext attack can be used to mount a chosen plaintext attack if the round function of the cipher consists of an addition of the

outputs of the S-boxes. By a careful choice of plaintexts, it is possible to fix the input to some S-boxes in the first rounds. S-boxes with fixed input are called *inactive*. The distribution of the output of an inactive S-box is an impulse.

The attack then works as follows. Guess a part of the subkey in the first round, such that for given inputs x and $x \oplus x'$, it becomes possible to calculate the output difference $y' = F(x) \oplus F(x \oplus x')$. Typically this means that the part of the subkey that enters one or two S-boxes has to be guessed. Encrypt several plaintexts that have the following properties: the only active S-boxes in the first round are the boxes from which the inputs can be calculated by guessing the key (this gives a condition on $t^{0\prime}$); the resulting y' is compensated for by choosing appropriate $s^{0\prime}$. This ensures that all S-boxes in round two are inactive, and that in round three the same S-boxes as in round one are active. The resulting distribution of β_R is less uniform than for random plaintexts, provided that the guess for the round key was correct. By comparing the distributions for all possible key guesses, the correct key can be determined.

The main advantage of this attack is that it no longer requires extensive precalculations and storage of huge β_R-tables. Moreover, it allows the subkey to be attacked part by part, thereby greatly reducing the required disk space, the memory usage, and the workload per plaintext. However, the attack requires about the same number of plaintexts as the known plaintext attack.

6.2 Application to CAST and LOKI91

The CAST design procedure was introduced in [2]. The common feature of all known members of the CAST family is that the round function uses S-boxes with fewer input bits than output bits. In [2] it was suggested that the S-boxes be suggested from bent functions. Later, CAST with random S-boxes was proposed [48]. For the purposes of the presented attack, this makes no difference.

There are also several varieties of key schedulings for CAST ciphers. The ciphers from [2, 3] use a 64-bit key, and round keys that have an entropy of 16 bits (this is explained in the next section). The key scheduling of [2] adds the round key *after* the S-boxes of the round function. This feature probably weakens the cipher with respect to the proposal of [3]. In the remainder of this section the cipher from [3] will be called CAST_{16}. Other versions of CAST may use round keys with 32 bit entropy and will be denoted by CAST_{32}.

The attack is applied to CAST_{16} and CAST_{32}: the complexity of the attack and the number of required known plaintexts is estimated and verified with experimental results.

Finally it is explained how to apply the attack to LOKI91.

# S-boxes	f
1	5.96×10^{-8}
2	1.53×10^{-5}
3	3.90×10^{-3}
4	6.32×10^{-1}

Table 6.1: Fraction f of possible output values for the combination of 1 to 4 typical CAST S-boxes.

6.2.1 CAST

The round function of a CAST cipher is constructed as follows. Define four tables S_1, S_2, S_3, and S_4, with eight input and 32 output bits. If $b_1b_2b_3b_4$ denotes the four byte input, the output is obtained by adding the output of the four S-boxes:

$$F(b_1b_2b_3b_4) = S_1[b_1] \oplus S_2[b_2] \oplus S_3[b_3] \oplus S_4[b_4]\,.$$

Since each S-box has only eight input bits, its output can only take 256 values in G^{32}. If the four S-boxes are selected at random, Lemma 6.1 states that the expected number of possible outputs is $(1 - e^{-1}) \times 2^{32}$. This value can also be computed from (6.4), since adding the outputs of the S-boxes is equivalent to concatenating rounds. Table 6.1 gives the fraction f of possible output values for the combination of 1, 2, 3 and 4 S-boxes.

Some CAST_{16} S-boxes are constructed from 8-bit bent functions that are the Walsh transforms of the concatenation of four 6-bit bent functions. S-boxes following this design principle were constructed. Typically they have the same value of f.

The CAST_{16} key scheduling is characterised by the following procedure: for each round, firstly an 'initial value' of two bytes is calculated from the master key. This calculation is simple for the first rounds, and more complicated for the last rounds. These two bytes are expanded in a nonlinear way to the 32-bit round key. The entropy of each round key is therefore at most 16 bits. This enables a cryptanalyst to perform an exhaustive key search for three round keys at once (see (6.6)).

The basic attack can be applied to a CAST_{16} variant, reduced to six rounds. Equation (6.3) becomes:

$$\begin{aligned} \beta_2 &= F(k^2 \oplus t^1) \\ &= t^0 \oplus s^6 \oplus F(k^4 \oplus t^6 \oplus F(k^5 \oplus s^6 \oplus F(k^6 \oplus t^6))) \oplus F(k^6 \oplus t^6). \end{aligned}$$

Value t^0 is part of the plaintext, s^6 and t^6 form the ciphertext, and k^4, k^5, and k^6 are the round keys that are searched. Note that by swapping plaintext and

	6 rounds	8 rounds
4×16	2^{15}	2^{23}
5×20	2^{19}	2^{27}
6×24	2^{23}	2^{38}
8×32	2^{32}	2^{62}

Table 6.2: Estimates for the number of plaintexts for various reduced versions of CAST_{32}.

ciphertext, the same attack can be applied to find k^1, k^2, and k^3. The work factor of the attack is then 1.5×2^{48}. The number of required texts is only $-\log(2^{48})/\log(1-e^{-1}) \approx 82$. Note that in [48] it is estimated that at least 2^{18} known plaintexts are required to break CAST_{16} reduced to six rounds with a linear attack. For CAST_{32} it is not feasible to search for several round keys at once.

Since the sum of two CAST round functions is surjective, the basic attack is not applicable to more than six rounds. The statistical attack requires a table of size 2^{32}. Although this is not infeasible, an implementation of this attack is very demanding. Therefore the attack was implemented for several mini-versions of CAST_{32} that use S-boxes of size 4×16, 5×20, and 6×24 respectively. The properties of the 8×32 S-boxes were approximated by using bent functions or (for the 5×20 case) random functions with bent function-like imbalance; the experiments indicate that the imbalance of the individual functions has very little influence on the distribution of β_{R-2} and consequently on the effectiveness of the attack.

Since it is possible to search for three round keys of CAST_{16} at once, an attack on R rounds of CAST_{16} would require about the same number of known plaintexts as an attack on $R-2$ rounds of CAST_{32}. Presumably, however, a much higher work factor is involved in the former case.

The precalculations for β_{R-2} composed of six up to twelve S-boxes were carried out, $4r$ S-boxes being equivalent to $2(r+1)$ rounds. In Figure 6.1, the distribution of β_{R-2} is shown for various numbers of 4×16 S-boxes (the distribution is translated over $\ln(2^p)$ such that it has zero mean). Figure 6.2 shows the resulting estimates for the number of required plaintexts. The results are summarised in Table 6.2. Based on Figure 6.2, it can be extrapolated that for CAST_{16} with 8×32 bit S-boxes and eight rounds, 2^{32} known plaintexts are required, and about 2^{62} known plaintexts for CAST_{32}.

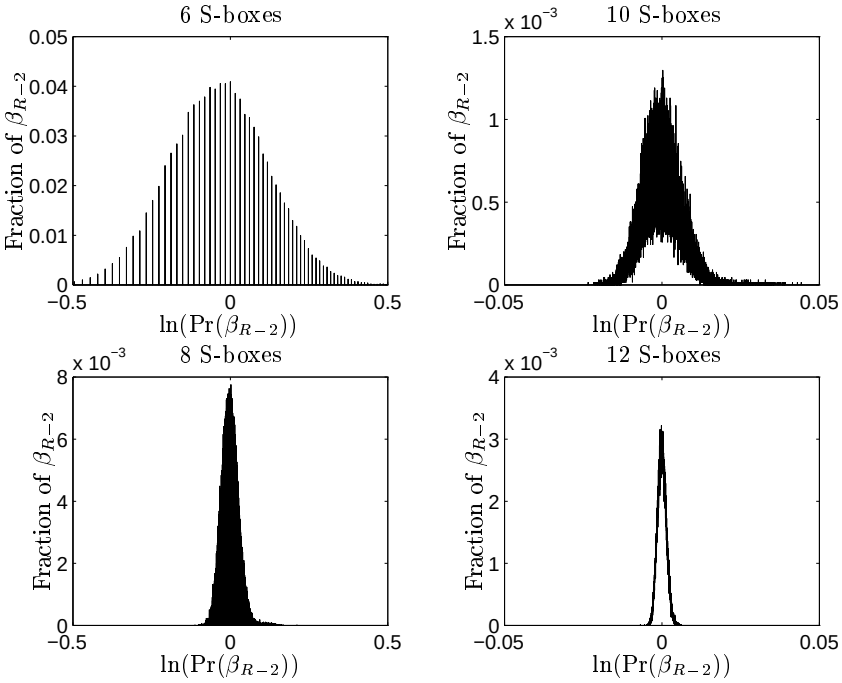


Figure 6.1: Fraction of values of β_{R-2} with the same probability as a function of the natural logarithm of that probability (translated over $\ln(2^p)$) for reduced versions of CAST_{32} with 4×16 S-boxes.

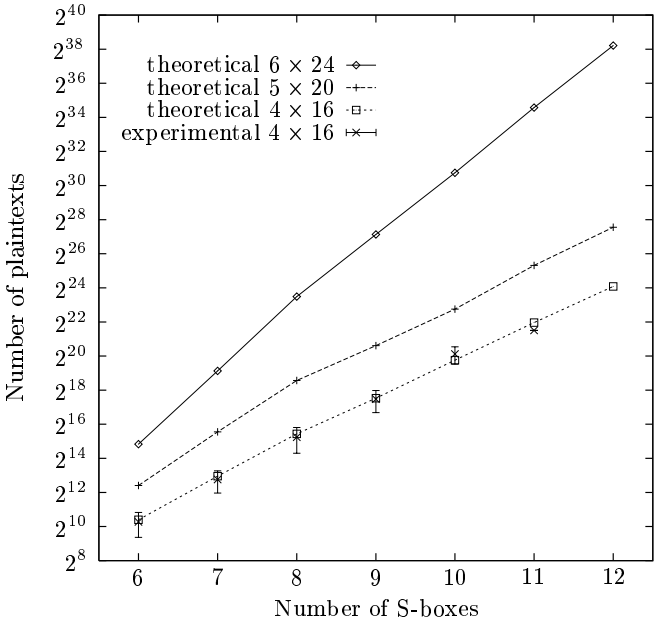


Figure 6.2: Number of known plaintexts required to find the CAST_{32} round key with probability e^{-1} for 4×16, 5×20 and 6×24 S-boxes, theoretical calculations and experimental results.

Verification

To verify the estimates, the attack was implemented. The attack is very suitable for a parallel implementation: each slave handles a set of plaintexts, and the master collects the results from all slaves and draws conclusions. The idle cycles of 50 workstations were used.

The reduced version with 4×16 S-boxes and β_{R-2} composed of 6 S-boxes, i.e., 6 rounds with only 2 S-boxes in the fourth round, was verified extensively, and also the case of eight 4-bit S-boxes. Some results of the former are collected in Figure 6.3.

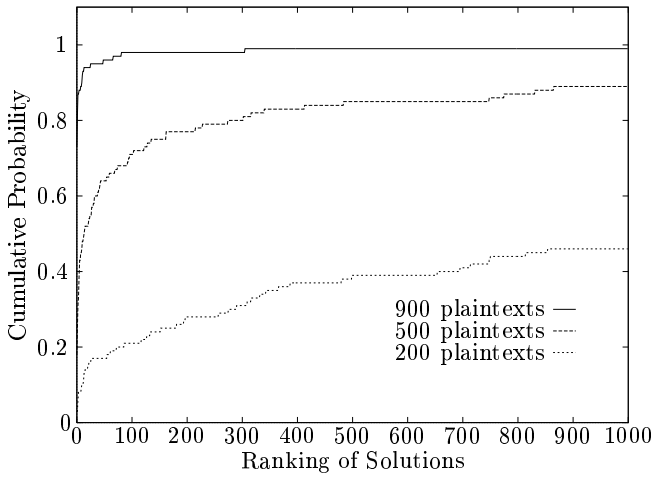


Figure 6.3: Cumulative probability for the rank of the right round key with, as parameter, the number of known plaintexts (for reduced CAST with six 4×16 S-boxes).

Also, a number of attacks for ten 4-bit S-boxes, twelve 4-bit S-boxes, six 5-bit S-boxes, eight 5-bit S-boxes and six 6-bit S-boxes were executed. The attack on eight 5-bit S-boxes took about one day on the 50 workstations. The major conclusion of all the experiments is that the estimates made in the previous paragraph seem quite accurate.

This can also be seen from Figure 6.2, where some experimental results for four bits are plotted, together with the theoretical estimates. The numbers plotted are the mean values (and standard deviations) of the number of plaintexts required to get the first ranking for the right round key. For six and seven S-boxes, the mean is taken over about 100 experiments; for eight, nine and ten S-boxes it is taken over 30 and for eleven S-boxes taken over only two

experiments.

Complexity

A first parameter is the disk space required for the calculation and storage of the table of β_{R-2}. For $l = 32$, this calculation requires 256 Kbyte, for $l = 48$ this increases to 64 Mbyte, to become 16 Gbyte for $l = 64$.

Other limiting factors are the memory usage and execution time per plaintext of the actual attack, because of the large number of counters (one for each key candidate) that have to be in memory and updated for every plaintext. Memory can be traded off for time by dividing the attack into multiple passes, considering only part of the key candidates and/or β_{R-2}-values in one pass.

6.2.2 LOKI91

The round function of LOKI91 takes a 32-bit message input and exors this with a 32-bit round key. These 32 bits are expanded to 48 bits and split into four parts. Each part enters the 12×8-bit S-box. This produces the $8 \times 4 = 32$ output bits. Note that of the 48 input bits to the nonlinear part, 32 bits are pairwise equal. In [59] L.R. Knudsen observed that this implies that the output can only take a fraction of $\frac{8}{13}$ of the possible values.

Each round key consists of 32 bits. The key scheduling of LOKI91 is such that k^{2r} is obtained by rotating k^{2r-1} over 12 positions to the left. Therefore it is possible to search for the round keys of two rounds at once, and the basic attack can be applied to five rounds of LOKI91. Since f is about the same for LOKI91 and CAST, comparable results for the extended attack are expected, except for the fact that only two rounds can be peeled off. In [127] the strength of various reduced versions of LOKI91 against linear and differential attacks is examined. The results are summarised in Table 6.3, together with the estimates of the strength against the new attack. For 9 rounds or more, the new attack becomes less efficient. Note that a differential attack requires *chosen* plaintexts, while the two other attacks require only *known* plaintexts.

	5 rounds	7 rounds	9 rounds
linear attack	2^{23}	2^{40}	2^{50}
differential attack	2^{8}	2^{16}	2^{30}
non–uniform attack	2^{6}	2^{32}	2^{62}

Table 6.3: Comparison of the data complexity of a linear attack, and estimates for the data complexity of a differential attack and the new attack for reduced versions of LOKI91.

6.3 Conclusions

A new attack has been presented that is applicable to Feistel ciphers with a small number of rounds ($R \approx 8$), that use a non-surjective round function. The attack easily breaks six rounds of CAST_{16}, requiring 2^{32} known plaintexts. By peeling off two additional rounds, the number of required plaintexts can be lowered to 82, or eight rounds can be broken with the same number of plaintexts, but with a much higher work factor. It is estimated that the attack also breaks eight rounds of CAST_{32} if 2^{62} known plaintexts are available. The attack leads to the following design criterion. Feistel ciphers with non-surjective round functions should use a number R of rounds that is large enough to make β_{R-2} surjective, where the sum β_r is defined in (6.1). In order to counter the statistical attack, β_{R-2} should have a distribution which is close to uniform.

There exist block ciphers that are based on the Luby-Rackoff [73] construction. The construction uses pseudo-random round functions to obtain a pseudo-random permutation. It is believed that a pseudo-random permutation is a good block cipher. Designers argue that the pseudo-randomness of the round function allows the number of rounds of the block cipher to be reduced to four. However, pseudo-random functions are non-surjective and the attack of this section also applies to these ciphers. It imposes a lower bound on the number of rounds that should be used.

With respect to the key scheduling of CAST [3], it can be seen that round keys with 16 bit entropy are inadequate. The computational cost for a cryptanalyst to peel off several rounds is too low. This makes CAST_{16} more vulnerable to the new attack than LOKI91. CAST_{32} with R rounds achieves the same resistance against the attack as $R+1$ rounds of LOKI91. The attack on CAST has been published in [112, 113].

Part II

Block Cipher Design

Chapter 7

Design Strategy & Components

The protection provided by encryption is based on the fact that most people would rather eat liver than do mathematics.

Bill Neugent

This chapter starts with explaining the Wide Trail design strategy [26]. The strategy is extended to include resistance to related-key attacks [55] and the interpolation attack [54]. Since the nonlinear components of block ciphers have already been studied extensively in the cryptographic literature, this chapter mainly focuses on the linear components. An important advantage of the Wide Trail strategy is that it allows block ciphers to be designed with a high and uniform resistance to linear and differential cryptanalysis. The last section shows how block ciphers can be designed that have exactly one selection of input and output bits with a strong correlation, while the correlation for the other selections are weak. This property is actually a trapdoor. This result has been published in [114].

7.1 Wide Trail Design Strategy

The Wide Trail design strategy was introduced by J. Daemen [26] as a means to construct cryptographic algorithms that resist differential and linear cryptanalysis. After a short historic overview the strategy is explained and the proposed extensions are formulated.

7.1.1 Historic Overview

The first theory behind the design of encryption algorithms can be found in [122], where C.E. Shannon proposes the building of strong ciphers by alternating simple substitutions with mixing transformations. The result of the combination is that *"any significant statistics from the encryption algorithm must be of a highly involved and very sensitive type—the redundancy has been both* diffused *and* confused *by the mixing transformation."* He observes that while it is easy to 'move' known quantities through the mixing transformations, this is quite difficult for unknown key dependent quantities, which are produced by the substitutions.

Feistel proposes in [36] the connection of small substitution boxes with a *"properly chosen wire-crossing pattern"* to provide diffusion. Confusion is provided by the substitution boxes. The design principles of the DES (partially revealed to the public world in [21]) give a first specification of the requirements for the S-boxes and the permutation P in order to achieve adequate confusion and diffusion. Since then, a number of different approaches has been developed. X. Lai, J. Massey and S. Murphy use a mix of operations over different algebraic groups to achieve strong ciphers [68, 69]. L.R. Knudsen and K. Nyberg have built ciphers with a provable resistance to various forms of cryptanalysis [96, 97]. The Wide Trail strategy was developed by J. Daemen in [26].

7.1.2 Description

In the Wide Trail strategy, the round transformation is composed of a number of uniform transformations. Let x denote the text input and k the (round) key input of the round transformation $\rho[k](x)$. The input x is divided into q p-bit blocks x_i, $0 \leq i < q$. The transformations are:

1. a nonlinear substitution layer γ operating separately on each p-bit block,
2. a linear diffusion layer θ mixing the p-bit blocks, and
3. an affine key addition $\sigma[k]$.

Here, only nonlinear substitution layers using $p \times p$ S-boxes are considered. The S-boxes are selected in such a way that there are no nonzero input exors that lead with high probability to certain output exors, and the input-output correlations are small. The linear diffusion layer is chosen so that there are no differential characteristics or linear relations over a small number of active S-boxes. The key addition specifies how the round keys are mixed with the input of every round. A fourth important component of a block cipher is the key schedule. It specifies how the round keys are derived from the key. The last two operations

have no role in the original Wide Trail strategy, as described in [26]. The round transformation is then given by

$$y = \rho[k](x) = \theta \circ \gamma \circ \sigma[k](x)\,. \tag{7.1}$$

The ordering of the different transformations can vary in specific designs.

The use of the term 'linear' implies a choice concerning the operation that will be considered as the linear operation: exor, modular addition, division, ... The Wide Trail strategy works for any choice of operation, but a choice must be made. The results obtained with respect to differential cryptanalysis are then only valid for the difference operation corresponding to the chosen linear operation. Since the exor operation is by far the most common choice of operation for successful cryptanalysis, in what follows it will be used as the linear operation.

By considering each component separately, a Wide Trail design becomes robust. The diffusion layer is selected to have uniform and good diffusion properties; the S-boxes are selected to have uniform nonlinear properties. The properties of the components are evaluated without taking the details of their interaction into account: there is no attempt made to compensate for weaknesses in the nonlinear layer by additional properties of the linear diffusion layer, or vice versa.

7.1.3 Differential and Linear Cryptanalysis

The probability of the best differential characteristic is often used as a measure of the resistance of the cipher to differential cryptanalysis. As a rule of thumb, the number of chosen plaintexts that are required for a differential attack is proportional to the inverse of the probability of the used differential [10]. The resistance to linear cryptanalysis is often measured by the highest input-output correlation [77]. The number of known plaintexts required for a linear attack is proportional to the inverse of the squared input-output correlation. For most proposed block ciphers it is difficult to calculate this probability or correlation. Usually it is not feasible to calculate all probabilities in order to select the best one. For some ciphers the search for the best characteristic or linear relation can be performed by using techniques from artificial intelligence (such as branch and bound or pruning) [7, 80], but in general these techniques are not sufficient to make the search practical. This is a problem for a designer who wants to verify the resistance of his design to these two general attacks. On the other hand, it opens up the possibility of constructing a trapdoor cipher by deliberately inserting a linear relation that has high deviation (cf. Section 7.5).

The Wide Trail strategy solves this problem by giving an upper bound for the probability of a differential characteristic and for the input-output correlation. The upper bound for the probability of a differential characteristic is

determined as follows. Firstly observe that for the linear diffusion layer and the affine key addition, the output difference is determined uniquely by the input difference. The only place where the probability of a differential characteristic is lower than one, is in the inactive S-boxes of the nonlinear layer. If δ is the maximal probability of a nonzero input difference leading to a certain output difference, and B is a lower bound for the number of active S-boxes in a differential characteristic, then the probability of a differential characteristic is clearly upper bounded by:

$$p \leq \delta^B \,. \tag{7.2}$$

Since the S-boxes are selected to have a low value for δ, and the linear diffusion layer is selected to have a high value for B, this upper bound can be very low. Note that this upper bound assumes that the rounds are independent (cf. Section 3.4). While in practice this assumption usually gives a good approximation, it has to be handled with care, and the following remarks have to be made. Firstly, if l is the block length in bits of the cipher then for any value of the key, the probability of any differential will either be zero, or an integer multiple of 2^{1-l}. Secondly, statistical techniques allow the following upper bound for the probability of the best differential characteristic of a general l-bit block cipher to be determined [98]:

$$p \leq l2^{1-l} \,.$$

For a reasonable number of rounds, this bound is usually higher than (7.2). A prudent approach might be to regard the highest of the bounds as an estimate of the probability of the best differential characteristic. The designer should keep in mind that the fact that all differential characteristics have low probabilities, does not guarantee that the cipher resists differential cryptanalysis, because the success probability actually depends on the probability for a fixed key of the differential.

The input-output correlation can be bounded in the same way. For the linear layers, every selection of input bits has correlation one with exactly one selection of output bits, and correlation zero with all other selections. If the maximal input-output correlation of an active S-box is given by λ and B is a lower bound for the number of active S-boxes in a linear relation, then the input-output correlation for the cipher is upper bounded by:

$$c \leq \lambda^B \,. \tag{7.3}$$

7.1.4 Differentials, Linear Hulls & Truncated Differentials

The Wide Trail strategy gives upper bounds for the probability of differential characteristics and the input-output correlation of a chain of linear relations

over the rounds of a block cipher, assuming that the round transformations act independently. In fact the strategy gives no strict guarantees of the resistance of the cipher to differential and linear attacks, because then the probability of differentials and linear hulls has to be used and the dependency of the rounds has to be considered. The Wide Trail approach differs from the designs with provable security to linear and/or differential cryptanalysis by L.R. Knudsen en K. Nyberg [96, 97]. It should also be noted that the 'provably secure' designs assume independent rounds and that the security proofs consider only a limited set of the possible attacks. At the moment neither the Wide Trail strategy, nor the provably secure approach take truncated differentials or new attacks into account.

However, the Wide Trail strategy emphasises the importance of diffusion as well as nonlinearity, which are, according to C.E. Shannon [122], both necessary for a strong algorithm.

7.1.5 Extensions

Mathematical structure

The building blocks of the round transformation can be selected in several ways. A first approach is to put forward a selection criterion and perform a random search until a candidate has been found that meets the criterion. A second approach is to use a mathematical construction that guarantees the required properties of the mappings. The second approach is usually much faster. However the inherent mathematical structure can also be exploited by the cryptanalyst. The designer has to ensure that the mathematical structure of the mappings does not translate to a mathematical structure for the cipher. An example of an attack that exploits the mathematical structure in the cipher is the interpolation attack [54]: if the operation of the cipher can be described as a mathematical function with a small number of key dependent coefficients then the cryptanalyst can collect a number of plaintext-ciphertext pairs and solve for the unknown coefficients. Once the coefficients are known, encryption and/or decryption of other messages can be done at will.

This means that during the design, the different components of the round function cannot be considered independently: if both exhibit mathematical structure then the design has to ensure that the structures are not compatible.

Key schedule

The key schedule is an important component for the resistance of the cipher to related-key attacks or attacks in which part of the key is known. If the block

cipher is used in the compression function of a hash function, the cryptanalyst can choose the key. Neglecting the key schedule will almost certainly lead to weaknesses.

The next sections discuss the required properties and construction methods for the separate round transformation components.

7.2 Diffusion Layer

In Chapter 3, basic definitions are introduced of functions and S-boxes that operate on bit vectors, i.e. mappings from G^n to G^m. These definitions are now extended to mappings from $G^{p\times n}$ to $G^{p\times m}$. ($G^{p\times n}$ is the vector space of dimension n, where the vectors have as components p-bit tuples.) The p-bit tuples are considered as elements from the finite field $GF(2^p)$. Since the vectors of $G^{p\times n}$ can also be considered as elements of $G^{(pn)}$, the vector space of binary (pn)-tuples, mappings with domain $G^{p\times n}$ can be studied from two different viewpoints. Sometimes it will be more convenient to consider the mappings as Boolean mappings, other times the representation with p-bit elements will be preferred.

From now on the *Hamming weight* of a vector means its number of non-zero components, where a component now has p bits. The *Hamming distance* of two vectors is still defined as the Hamming weight of their difference.

The addition of two elements from $GF(2^p)$ can be performed by exoring the individual bits of the elements. This leads to the following lemma.

Lemma 7.1 *All functions that are linear over $GF(2^p)$ can be considered as S-boxes with component functions that are linear over $GF(2)$, and vice versa.*

Proof: The proof follows from the definition of a linear function.

$$
\begin{array}{l}
f : (GF(2^p))^n \to GF(2^p) : x \mapsto f(x) \text{ is linear} \\
\qquad\qquad \Updownarrow \\
\forall x, y \in (GF(2^p))^n : f(x \oplus y) = f(x) \oplus f(y) \\
\qquad\qquad \Updownarrow \\
\forall x, y \in (GF(2))^{pn} : f_i(x \oplus y) = f_i(x) \oplus f_i(y), i = 1, \ldots, m \\
\qquad\qquad \Updownarrow \\
f_i : (GF(2))^{pn} \to GF(2) : x \mapsto f_i(x) \text{ is linear}, i = 1, \ldots, m\,.
\end{array}
$$

■

This lemma is used later, when diffusion layers are constructed that are linear in $GF(2)$, based on codes that are linear over $GF(2^p)$. An important difference between linear functions from $(GF(2))^n$ to $GF(2)$ and linear functions from $(GF(2^p))^n$ to $GF(2^p)$ is the following. If a function f from $(GF(2))^n$ to $GF(2)$

is linear then it can be described as a vector product. There always exists a vector $a \in (GF(2))^n$ such that

$$f(x) \equiv a \bullet x = \bigoplus_{i=1}^{n} a_i \cdot x_i \,.$$

This property does not hold for linear functions from $(GF(2^p))^n$ to $GF(2^p)$. The simplest counterexample is the squaring operation. In a field with characteristic q, it holds that $(x \oplus y)^q = x^q \oplus y^q$ [66]. Thus, in $GF(2^p)$ $(x \oplus y)^2 = x^2 \oplus y^2$; but there exists no $a \in GF(2^p)$ such that $x^2 \equiv a \cdot x$.

The extension of differential cryptanalysis to functions operating on p-bit values is straightforward. A meaningful extension of linear cryptanalysis is less obvious. The key element for linear cryptanalysis is the concept of correlation. It is not clear how this concept can be extended and used in $GF(2^p)$. Therefore only bitwise linear relations will be considered. To simplify the notation, the correlation between two functions from $(GF(2^p))^n$ to $GF(2^p)$ will be defined as the maximum of the correlations between all linear combinations of their component functions:

$$c(f(x), g(x)) = \max_{\alpha,\beta \in GF(2^p)} c(\alpha \bullet f(x), \beta \bullet g(x)) \,. \tag{7.4}$$

7.2.1 Measuring Diffusion

A necessary condition for a block cipher is *completeness*: every ciphertext bit has to depend on every plaintext bit and on every key bit. This objective is met by the block cipher in two steps:

1. The most efficient way to implement a nonlinear function is usually by means of table lookups. Since a function with n input bits and m output bits requires a table of $m2^n$ bits, n and m are always chosen to be smaller than l. If the nonlinear layer is implemented with p-bit S-boxes, each of the output bits is only influenced by the p input bits of the S-box from which it is an output, which depend on a restricted set of key bits and plaintext bits.

2. The diffusion layer ensures the mixing of the sets, such that after a few rounds completeness is reached. In the Wide Trail strategy, the p-bit output from an S-box is considered as one input value for the diffusion layer. Completeness for the diffusion layer means then that every p-bit output depends on every p-bit input.

The key addition is not used to provide diffusion.

The requirement for completeness is a very weak and easily met requirement. To provide resistance to cryptanalysis it is necessary that every ciphertext bit is a complex function of all the plaintext bits and all the key bits. In the context of differential cryptanalysis this means that small input changes should cause large output changes, and conversely, to produce a small output change, a large input change should be necessary. Here a 'small change' means a change in only a few p-bit values, whereas a 'large change' means that many p-bit values are changed. To provide resistance to linear cryptanalysis there should be no correlations between linear combinations of a small set of (p-bit) inputs and linear combinations of a small set of (p-bit) outputs.

In the literature several measures of the diffusion of a mapping have been proposed.

A mapping $\theta(x)$ displays the *avalanche effect* [36] if on average one half of the component functions of $\theta(x)$ are complemented when one component of x is complemented.

If complementing one component of x causes every component of $\theta(x)$ to complement with probability 0.5, θ obeys the *strict avalanche criterion (SAC)* [132].

A function f satisfies the *propagation criterion of degree k, $PC(k)$* ($1 \leq k \leq n$), if $f(x)$ changes with probability 0.5 whenever i ($1 \leq i \leq k$) bits of x are complemented. The function satisfies the propagation criterion of degree k and order m if any function obtained from f by keeping m input bits constant satisfies $PC(k)$ [102].

Three problems with the avalanche effect and SAC are evident from the definitions. A first problem is that only input changes of Hamming weight one are considered. When larger changes are applied, the situation might be far worse.

Example 7.1 *Consider the mapping*

$$\theta(x) = y \Leftrightarrow y_i = \bigoplus_{j=1, j\neq i}^{n} x_j \cdot x_i \,.$$

It is easy to verify that θ satisfies the SAC. However, if two components x_t, x_u are changed at the same time, only y_t and y_u change with probability 0.5. All the other components of y remain constant.

A second problem is that the properties are probabilistic. If the diffusion is good *on average*, cryptanalysts may be able to exploit an occasion where the diffusion is much lower than the average value. A third problem is that the definitions only deal with resistance to differential attacks, and not with linear or other correlation-based attacks. The propagation criterion allows the first two problems to be solved by raising k and m. The third problem remains.

The *branch number* $\mathcal{B}$ of a mapping is a measure that does not suffer from any of these problems [26]. The branch number is especially useful to measure the performance of the diffusion layer in a design that follows the Wide Trail strategy. In order to get a more general result the definition of the branch number of a mapping presented here deviates slightly from the definition given by J. Daemen [26]. It will be discussed under what conditions the definitions are equivalent.

7.2.2 Branch Numbers: Three Definitions

J. Daemen defines the branch number $\mathcal{B}(\theta)$ of a linear mapping θ as follows:

Definition 7.1 (J. Daemen [26]) *The* branch number *of a linear mapping is given by*

$$\mathcal{B}(\theta) = \min_{a \neq 0} \{ w_h(a) + w_h(\theta(a)) \} . \tag{7.5}$$

Here the branch number of a general mapping will be defined. It is necessary to make a distinction between the differential and the linear branch number.

Definition 7.2 *The* differential branch number *of a mapping (linear or non-linear) is defined by*

$$\begin{aligned} \mathcal{B}_d(\theta) &= \min_{a,b \neq a} \{ w_h(a \oplus b) + w_h(\theta(a) \oplus \theta(b)) \} & (7.6) \\ &= \min_{a \neq 0, b, e_\theta(a \| b) \neq 0} \{ w_h(a) + w_h(b) \} . & (7.7) \end{aligned}$$

The following proposition relates the value of $\mathcal{B}_d(\theta)$ to the worst case diffusion of θ in a differential context.

Proposition 7.2 *For any iterated block cipher with S-boxes, the number of active S-boxes in a two-round differential characteristic is lower bounded by the differential branch number of the mapping that is used in the diffusion layer.*

Proof: Consider two rounds of a block cipher, with diffusion layer θ.

$$\rho[k^2] \circ \rho[k^1](x) = \theta \circ \gamma \circ \sigma[k^2] \circ \theta \circ \gamma \circ \sigma[k^1](x)$$

Let a, b be two inputs, where $w_h(a \oplus b) = d$. Since σ and γ do not mix different p-bit values, the number of active S-boxes in γ of the first round is given by $w_h(\sigma[k^1](a) \oplus \sigma[k^1](b)) = d$, and the number of active inputs of θ is

$$w_h(\gamma(\sigma[k^1](a)) \oplus \gamma(\sigma[k^1](b))) = d .$$

Let the number of active outputs of θ be denoted by e, where

$$e = w_h(\theta(\gamma(\sigma[k^1](a))) \oplus \theta(\gamma(\sigma[k^1](b)))) .$$

The number of active S-boxes in γ of the second round is then also e. Equation (7.6) guarantees that $d + e \geq \mathcal{B}_d(\theta)$. ∎

For a linear mapping $\theta(a) \oplus \theta(b) = \theta(a \oplus b)$, and (7.6) reduces to (7.5).

Definition 7.3 *The* linear branch number *of a mapping is defined as*

$$\mathcal{B}_l(\theta) = \min_{\alpha,\beta,c(\alpha \bullet x, \beta \bullet \theta(x)) \neq 0} \{w_h(\alpha) + w_h(\beta)\} \tag{7.8}$$

This definition is the equivalent of Definition 7.2 in linear cryptanalysis. It can be shown that $\mathcal{B}_l(\theta)$ gives a lower bound for the number of active S-boxes in a linear relation. If θ is a linear mapping, characterised by the matrix Ω (cf. Section 3.1.3), then (7.8) can be reduced in the following way.

$$\mathcal{B}_l(\theta) = \min_{\alpha \neq 0} \{w_h(\alpha) + w_h(\theta^t(\alpha))\} , \tag{7.9}$$

where θ^t is the mapping characterised by transposed matrix Ω^t (cf. Section 3.1.3). Note that in general (7.9) is not equivalent to (7.5).

7.2.3 Branch Numbers and Coding Theory

The branch numbers of linear mappings can be studied using the framework of linear codes over $GF(2^p)$ [74, 101, 130].

Definition 7.4 *A* linear $[n, k, d]$ code *over $GF(2^p)$ is a k-dimensional subspace of the vector space $(GF(2^p))^n$, where any two different vectors of the subspace have a Hamming distance of at least d (and d is the largest number with this property).*

The distance d of a linear code equals the minimum weight of any nonzero codeword. A linear code can be described by each of the two following matrices:

- A *generator* matrix G for an $[n, k, d]$ code $\mathcal{C}$ is a $k \times n$ matrix whose rows form a vector space basis for $\mathcal{C}$ (only generator matrices of full rank are considered). Since the choice of a basis in a vector space is not unique, a code has many different generator matrices that can be reduced to one another by performing elementary row operations. If permutation of the coordinate positions is allowed, it is always possible to find a generator matrix having the form

  $$G_e = [I_{k \times k} \; A_{k \times (n-k)}] .$$

  This form is called the *echelon form*, or *standard form* of the generator matrix. The code is then in *systematic* form.

- A *parity check* matrix H for an $[n, k, d]$ code $\mathcal{C}$ is an $(n-k) \times k$ matrix with the property that a vector x is a codeword of $\mathcal{C}$ if and only if

$$Hx^t = 0\,.$$

If G is a generator matrix and H a parity check matrix of the same code, then

$$GH^t = 0\,.$$

Moreover, if $G = [I\ C]$ is a generator matrix of a code, then $H = [-C^t\ I] = [C^t\ I]$ is a parity check matrix of the same code.

The *dual code* $\mathcal{C}^\perp$ of a code $\mathcal{C}$ is defined as the set of vectors that are orthogonal to all the vectors of $\mathcal{C}$;

$$\mathcal{C}^\perp = \{x \mid x \bullet y = 0, \forall y \in \mathcal{C}\}\,.$$

It follows that a parity check matrix of $\mathcal{C}$ is a generator matrix of $\mathcal{C}^\perp$ and vice versa.

Codes can be associated with mappings in the following way.

Definition 7.5 *Let θ be a mapping from $(GF(2^p))^n$ to $(GF(2^p))^n$. The* associated code *of θ, $\mathcal{C}_\theta$, is the code that has codewords given by the vectors $(x||\theta(x))$. The code $\mathcal{C}_\theta$ has 2^n codewords and has length $2n$.*

If θ is defined as $\theta(x) = x \cdot A$, then $\mathcal{C}_\theta$ is a linear $[2n, n, d]$ code. Code $\mathcal{C}_\theta$ consists of the vectors $(x||(x \cdot A))$, where x takes all possible input values. Equivalently, the generator matrix G_θ of $\mathcal{C}_\theta$ is given by

$$G_\theta = [I\ A]\,,$$

and the parity check matrix H_θ is given by

$$H_\theta = [-A^t\ I] = [A^t\ I]\,.$$

It follows from Definition 7.6 that the differential branch number of a mapping θ equals the distance of the associated code $\mathcal{C}_\theta$. The theory of linear codes addresses the problem of determining the distance of a linear code extensively. This theory can be used to determine the differential branch number of a linear mapping.

Proposition 7.3 (The Singleton bound) *If $\mathcal{C}$ is an $[n, k, d]$ code, then $d \leq n - k + 1$.*

A code that meets this bound, is called a *Maximal Distance Separable (MDS) code*. Applied to an n-dimensional linear mapping, this proposition states that

$$\mathcal{B}_d \leq n+1\,.$$

The upper bound is easily explained. For every mapping θ it holds that $w_h(\theta(x)) \leq n$. Since it is possible to choose x such that $w_h(x) = 1$, it follows that $\mathcal{B}_d(\theta) \leq n+1$. Mappings that reach this bound are called *optimal (differential) diffusion mappings*.

Proposition 7.4 *A linear code $\mathcal{C}$ has distance d if and only if every $d-1$ columns of the parity check matrix H are linearly independent and there exists some set of d columns that are linearly dependent.*

An MDS-code has distance $n-k+1$, thus every $n-k$ columns of the parity check matrix are linearly independent. This property can be translated to a requirement for the matrix A [74]:

Proposition 7.5 *An $[n,k,d]$-code with generator matrix $G = [I_{k\times k}\ A_{k\times(n-k)}]$ is a maximum distance separable (MDS) code if and only if every square submatrix of A is nonsingular.*

Corollary 7.6 *All linear optimal (differential) diffusion mappings are invertible.*

This follows from the fact that A is nonsingular.

The following proposition relates the linear branch number of a linear mapping to the dual of the associated code.

Proposition 7.7 *If $\mathcal{C}_\theta$ is the associated code of the linear mapping θ, then the linear branch number of θ is equal to the distance of the dual code of $\mathcal{C}_\theta$.*

Proof: Lemma 3.1 states that the correlation between two linear functions can take only two values: if the linear functions are different, the correlation is zero, otherwise it is one. Therefore in (7.8) only the values for (α,β) where $c(\alpha \bullet x, \beta \bullet \theta(x)) = 1$ have to be considered. From (3.9) it follows that the correlation is one if

$$\sum_x (-1)^{\alpha\bullet x \oplus \beta\bullet\theta(x)} = 2^n\,.$$

This is equivalent to the requirement that for all codewords $(x\|\theta(x))$ of $\mathcal{C}$

$$\alpha \bullet x \oplus \beta \bullet \theta(x) = (x\|\theta(x)) \bullet (\alpha\|\beta) = 0,$$

or $(\alpha\|\beta) \in \mathcal{C}^\perp$. The linear branch number of s is defined to be the minimum weight of the vectors (α,β), which is by definition the distance of $\mathcal{C}^\perp$. ∎

Corollary 7.8 *The linear branch number of the linear mapping $\theta : x \mapsto x \cdot A$ is equal to the differential branch number of the mapping $\theta^t : x \mapsto x \cdot A^t$.*

Example 7.2 shows that the differential and the linear branch number of a linear mapping need not to be equal.

Example 7.2 *Consider the mapping $\theta : x \mapsto x \cdot A$ over $GF(4)$, with*

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} .$$

Since $H = [A^t\ I]$ has two equal columns, $\mathcal{B}_d(\theta) = 2$. For $[A\ I]$, all sets of two columns are independent, and there are three linearly dependent columns. Thus $\mathcal{B}_l(\theta) = 3$.

There do exist classes of mappings with equal differential and linear branch number. An obvious sufficient condition is the requirement that A be symmetric. A second class is the mappings that have an associated code that is MDS. Indeed, if all submatrices of A are nonsingular, then this holds also for A^t. Thus all mappings with optimal differential diffusion also have optimal diffusion against linear attacks. A third important class is the mappings with a circulant matrix A.

Definition 7.6 *An $n \times n$ matrix A is* circulant *if there exist n constants $a_1, \ldots, a_n$ and a 'step' $c \neq 0$ such that for all i, j $(0 \leq i, j < n)$*

$$a_{i,j} = a_{i+cj \mod n} .$$

If $\gcd(c, n) = 1$ then the branch numbers of θ are equal.

7.2.4 MDS-Codes

A well-known subclass of MDS-codes is formed by the Reed-Solomon codes (RS-codes). RS-codes over the field $GF(2^p)$ can have lengths of up to $2^p - 1$ [74]. RS-codes can be constructed very efficiently as follows.

Let a codeword b correspond to a polynomial

$$b(x) = \bigoplus_{i=1}^{n} b_i x^{i-1} .$$

Choose α as a primitive element in $GF(2^t)$. Then the polynomial

$$g(x) = (x \oplus \alpha) \cdot (x \oplus \alpha^2) \cdot \ldots \cdot (x \oplus \alpha^n)$$

generates a $(2n, n, n+1)$-Reed-Solomon code. The codewords are formed by the polynomials of degree $< 2n$ that are multiples of $g(x)$.

The standard form of the generator matrix can be constructed in the following way [101]. Let $a_i(x)$ be the remainder after dividing x^i by $g(x)$:

$$x^i = g(x) \cdot q(x) \oplus a_i(x) \,.$$

Then,

$$x^i \oplus a_i(x) = g(x) \cdot q(x)$$

is a codeword. We take these codewords for $i = 2n-1, 2n-2, \ldots, n$, as rows of G. It follows that

$$G = [I_{n \times n} \; A_{n \times n}]$$

is the standard form of the generator matrix.

MDS codes can also be generated by a random search for the matrix A. Due to the large number of square submatrices and the non-negligible probability (2^{-p}) that a determinant is 0, this approach is only suitable for small values of n and becomes computationally infeasible as n grows. The number of square submatrices is

$$n_m = \sum_{i=1}^{n} \binom{n}{i}^2 ,$$

and the expected number of trials is

$$n_t = (1 - 2^{-p})^{-n_m} \,.$$

Table 7.1 gives the values of n_m and n_t for various values of n, if $p = 8$.

Note that n_t can be reduced by looking for codes for which the matrix A is circulant. In a circulant matrix many of the submatrices are equivalent. This results in a significant reduction of n_t. A closed formula for the number of equivalence classes is difficult to obtain. Table 7.1 shows the numbers for $n = 1, \ldots, 8$, which were all determined experimentally.

7.2.5 Multi-Level Diffusion

In the mappings discussed in the previous sections, every output block can depend on every input block. Under this condition it is possible to construct mappings with a high branch number, that guarantee many active S-boxes in every two-round differential characteristic or linear relation. It is possible to use a combination of incomplete mappings and still obtain adequate diffusion. This

n	n_m		n_t	
	generic	circulant	generic	circulant
1	1	1	1	1
2	5	3	1	1
3	19	7	1.1	1
4	69	17	1.3	1.1
5	251	41	2.7	1.2
6	923	111	37	1.5
7	3431	309	$7 \cdot 10^5$	3.4
8	12869	935	$7 \cdot 10^{21}$	39

Table 7.1: The number of square submatrices in a generic matrix of order n, the number of non-equivalent determinants in a circulant matrix of the same order and the expected number of trials to find a matrix which has all submatrices nonsingular.

section deals with the most simple extension: two-level diffusion. Only the case of a differential characteristic is considered, since the linear case is equivalent.

Two-level diffusion in the n-dimensional vector space proceeds in several stages. The n inputs of the mappings are divided into n_2 *classes* of n_1 inputs each, $n = n_1 n_2$. There are two mappings θ_1, θ_2, both incomplete, that will be used alternately to produce the strong diffusion:

- Mapping θ_1 mixes only inputs of the same class; it can be considered as the parallel application of n_2 mappings h_i, each having n_1 inputs.

  $$\theta_1(x) = (h_1(x_1, \ldots, x_{n_1}), h_2(x_{n_1+1}, \ldots, x_{2n_1}), \ldots, h_{n_2}(x_{(n_2-1)n_1+1}, \ldots, x_n))$$

  Mapping θ_1 is clearly incomplete and its differential branch number is the minimum of the differential branch numbers of the mappings h_i.

- Mapping θ_2 mixes the inputs of different classes. The diffusion of θ_2 is measured in terms of classes rather than in terms of the n individual inputs. If one class is considered as one input, then the diffusion of θ_2 can also be measured by the differential branch number. In the following, $\mathcal{B}_d^*$ will denote the differential branch number with respect to classes.

Proposition 7.9 bounds the diffusion in a construction with two-level diffusion.

Proposition 7.9 *Consider the following transformation:*

$$\mathcal{A}_{\theta_1,\theta_2}(x) = \gamma \circ \theta_1 \circ \gamma \circ \theta_2 \circ \gamma \circ \theta_1 \circ \gamma(x)\,,$$

where θ_1 and θ_2 are diffusion mappings and γ is a nonlinear layer, implemented with S-boxes. The total number of active S-boxes in a four-round differential over A is lower bounded by $\mathcal{B}_d(\theta_1) \times \mathcal{B}_d^*(\theta_2)$.

Proof: Define $\mathcal{A}_{\theta_1} = \gamma \circ \theta_1 \circ \gamma$ and $\mathcal{A}_{\theta_1,\theta_2} = \mathcal{A}_{\theta_1} \circ \theta_2 \circ \mathcal{A}_{\theta_1}$. Proposition 7.2 implies that the number of active S-boxes in the transformation $\mathcal{A}_{\theta_1}$ is lower bounded by $\mathcal{B}_d(\theta_1)$. Since θ_1 does not mix inputs of different classes, the number of active classes stays invariant under $\mathcal{A}_{\theta_1}$ and the number of active S-boxes is lower bounded by $\mathcal{B}_d(\theta_1)$ times the number of active classes. Thus, the number of active S-boxes in a characteristic over $\mathcal{A}_{\theta_1,\theta_2}$ is lower bounded by $\mathcal{B}_d(\theta_1)$ times the sum of the number of active classes before and after θ_2. By definition, the latter sum is lower bounded by $\mathcal{B}_d^*(\theta_2)$, which proves the proposition. ∎

Application in a Design

For some applications, the implementation of a diffusion layer that is based on a complete mapping is not acceptable because of the computational effort that is required to evaluate it. Mappings with good two-level diffusion can provide a good alternative. An example is given in Chapter 8: the block cipher SQUARE has a block length of 128 bits, which makes the computation of the result of a complete mapping very expensive.

The most obvious way to apply the result of Proposition 7.9 is to define a block cipher with two different round operations, that are used alternately. In principle this construction is no longer an iterated block cipher. Since most fast implementations of block ciphers unroll loops, it might be that there are no practical objections. However, there is a way to unify the two different round transformations again. Define an extra transformation π that permutes the p-bit values. Let $\theta = \pi \circ \theta_1$. The round transformation becomes:

$$\rho[k](x) = \pi \circ \theta_1 \circ \gamma \circ \sigma[k] \circ \pi \circ \theta_1 \circ \gamma \circ \sigma[k](x)\,. \tag{7.10}$$

Since γ operates on each p-bit value separately, it commutes with π:

$$\pi \circ \gamma(x) = \gamma \circ \pi(x)\,.$$

Also, under $\sigma[k]$ the tuples are transformed independently of one another:

$$\pi \circ \sigma[k](x) = \sigma[\pi(k)] \circ \pi(x)\,.$$

Consider now the following transformation, where σ is left out to simplify notation:

$$\begin{aligned}
\mathcal{T} &= \rho \circ \rho \circ \rho \circ \rho \\
&= \pi \circ \theta_1 \circ \gamma \circ \pi \circ \theta_1 \circ \gamma \circ \pi \circ \theta_1 \circ \gamma \circ \pi \circ \theta_1 \circ \gamma \\
&= \pi \circ \theta_1 \circ \pi \circ \gamma \circ \theta_1 \circ \gamma \circ \pi \circ \theta_1 \circ \pi \circ \gamma \circ \theta_1 \circ \gamma \\
&= \pi \circ \theta_1 \circ \pi \circ \mathcal{A}_{\theta_1,\pi\circ\theta_1\circ\pi}\,.
\end{aligned}$$

The transformation A is defined as in Proposition 7.9, where θ_2 equals $\pi \circ \theta_1 \circ \pi$. The diffusion of the transformation $\mathcal{T}$ is then bounded by the results of Proposition 7.9. SQUARE (cf. Section 8.2) is an example of a cipher with two-level diffusion, but identical round operations.

In general, t-level diffusion can be accomplished by defining t mappings θ_i: θ_1 operates separately on classes of n_1 inputs, θ_2 operates on super-classes of n_2 classes, and so on for $\theta_3, \ldots$ This can be implemented efficiently if the round functions are permitted to vary; if all the round functions have to be equal then the analysis becomes difficult.

7.3 Nonlinear Layer

The nonlinear layer provides resistance to linear and differential cryptanalysis. It is implemented with S-boxes. A large number of S-box criteria, sometimes contradictory, have been published (see, for example, [21, 33, 56, 92, 93]).

As explained above, the Wide Trail strategy uses $p \times p$ S-boxes where δ, the maximal probability that a non-zero input exor leads to a certain output exor, and λ, the maximal input-output correlation, are low. Only construction methods for $p \times p$ S-boxes will thus be discussed.

As in the case of the diffusion layer, there are essentially two approaches possible for generating S-boxes: either generate a set of random S-boxes and select the instances that perform best with respect to the criterion, or search for explicit constructions. Section 7.3.1 discusses a method of constructing S-boxes with optimal nonlinearities. Section 7.3.2 discusses a method of removing regularities that may result from explicit construction methods, and Section 7.3.3 discusses the results that can be achieved by random search methods.

7.3.1 Explicit Construction

The construction of nonlinear S-boxes has already been discussed extensively in the cryptographic literature [1, 56, 93]. For S-boxes with the same number of inputs and outputs, the lowest values for δ and λ are obtained using the construction methods presented in [93]. The following mappings are presented:

- $s(x) = x^{2^k+1}$ in $GF(2^p)$:
  This mapping has the following properties: $\delta = 2^{-p} \cdot \gcd(k, p)$; if $p/\gcd(k, p)$ is odd, then s is invertible and $\lambda = 2^{(\gcd(p,k)-p)/2}$. Note that $p/\gcd(p, k)$ can only be odd if p has an odd factor (p cannot be a power of two). The component functions and all the linear combinations of the component functions then have nonlinear degree 2.

- $s(x) = x^{-1}$ in a finite field:
  If p is odd, then $\delta = 2^{1-p}$, otherwise $\delta = 2^{2-p}$. Also, $\lambda \leq 2^{(2-p)/2}$. The nonlinear degree of the component functions and all the linear combinations of the component functions is $p - 1$.

- $s(x) = u^x$ in a prime field $GF(q)$:
  If the order of u is t, then $\delta = q^{-1}(1 + (q-1)/t)$.

The disadvantage of any explicit construction is that the resulting S-boxes contain some structure which could be exploited by a cryptanalyst in an interpolation attack [54]. One technique of destroying this structure is to apply an invertible bitwise affine transformation to the output bits of the S-box. In this way, the description of the mapping s as a polynomial over $GF(2^p)$ can be made very complex. Note that any p-bit mapping can be represented as a polynomial or a rational form in $GF(2^p)$. Therefore it seems plausible that the application of an affine transformation can make the constructed S-box as secure as a random S-box against attacks that exploit existing structure.

7.3.2 Modifications

Another approach to destroying the structure of a mapping that is constructed as in the previous section is to apply some random changes to the output. The disadvantage of this approach is that δ and/or λ will increase.

Some experiments were performed starting from the mapping $s(x) = x^{-1}$ over $GF(2^8)$ as described in Section 7.3.1. This mapping has $\delta = 4 \cdot 2^{-8}$ and $\lambda = 8 \cdot 2^{-6}$. In this mapping a number of entries were randomly swapped and the resulting values of δ and λ were recorded. For the swapping of two entries, all possibilities were tried; for the swapping of more entries, the best results out of 300 000 samples were recorded for each case. Table 7.2 shows the results of these experiments.

7.3.3 Random Search

Among many cryptographers there is a strongly held belief that the existence of any structure in the components of a block cipher should be avoided. According to this line of thought, a good block cipher is a block cipher with as many randomly selected elements as possible. A consequence of this belief is that the best S-box is a random S-box. In [98] the average differential properties of permutations are investigated and a bound for the expected value of δ is given. For an m-bit permutation,

$$\lim_{m \to \infty} \frac{E[\delta 2^m]}{2m} \leq 1 \,.$$

modifications (%)	δ	λ
0	$4 \cdot 2^{-8}$	$8 \cdot 2^{-6}$
0.8	$4 \cdot 2^{-8}$	$9 \cdot 2^{-6}$
1.6	$4 \cdot 2^{-8}$	$9 \cdot 2^{-6}$
3.1	$6 \cdot 2^{-8}$	$10 \cdot 2^{-6}$
4.7	$6 \cdot 2^{-8}$	$10 \cdot 2^{-6}$
6.3	$6 \cdot 2^{-8}$	$11 \cdot 2^{-6}$
12.5	$6 \cdot 2^{-8}$	$12 \cdot 2^{-6}$
25	$8 \cdot 2^{-8}$	$13 \cdot 2^{-6}$
50	$8 \cdot 2^{-8}$	$15 \cdot 2^{-6}$

Table 7.2: Experimental values of δ and λ obtained by randomly swapping entries from the mapping $s(x) = x^{-1}$ over $GF(2^8)$. For the second row, all possibilities were tried. For the other rows, the best results out of 300 000 samples were recorded.

Often a collection of random S-boxes are generated and evaluated against some design criteria. The S-box that performs best is selected. In order to get an idea of what the typical δ and λ values are for a random S-box, 1.5 million samples were generated and evaluated. Table 7.3 shows the results. The S-boxes with the highest resistance to both linear and differential cryptanalysis have $\delta = 10 \cdot 2^{-8}$ and $\lambda = 15 \cdot 2^{-6}$. The experimental results give

$$\frac{E[\delta 2^m]}{2m} = \frac{11.3 \cdot 2^{-8} \cdot 2^{-8}}{2 \cdot 8} = 0.71 \,.$$

It is an open question as to whether the approach of generate-and-test results in S-boxes without structure: it might well be that the evaluation criteria implicitly impose a certain structure on the S-boxes. If this is the case, then the 'randomly' selected S-boxes with the highest performance values with respect to the criteria would also have the most structure.

Randomly selected S-boxes have a second important problem. If the designer does not present the user with an unambiguous construction method for the S-boxes, then the user has to trust the designer that the S-boxes do not exhibit some hidden properties (trapdoors). In Chapter 8 it is explained how trapdoor ciphers can be built by using S-boxes that are computationally indistinguishable from randomly selected S-boxes.

7.3.4 Branch Numbers

The branch numbers of nonlinear mappings can also be calculated and used as a design criterion. For example, in DES-like ciphers, where the individual

$\lambda \cdot 2^{-6}$	$\delta \cdot 2^{-8}$						
	8	10	12	14	16	18	20
15	0	0.07	0.07	0.006	0.0001	0	0
16	0.0003	4.77	5.58	0.58	0.04	0.002	0
17	0.002	15.63	20.55	2.24	0.15	0.007	0.0004
18	0.0002	12.21	17.17	1.96	0.13	0.007	0.0005
19	0.0004	4.91	7.31	0.87	0.05	0.003	0
20	0	1.52	2.34	0.28	0.02	0.001	0
21	0	0.41	0.64	0.08	0.004	0.001	0

Table 7.3: Maximum input-output correlation and difference propagation probability of randomly generated 8-bit permutations. The entries denote the percentage of the generated mappings that have the indicated λ and δ.

output bits are permuted to the inputs of different S-boxes in the next round, it makes sense to have S-boxes with high branch numbers in order to ensure good diffusion properties.

In the framework of the Wide Trail strategy, the branch numbers of the S-boxes used in the nonlinear layer are not important. However, it is possible to use a nonlinear diffusion layer instead of a linear layer. By doing so, the diffusion layer can be selected from a much larger set of mappings. Also, it can be argued that a nonlinear mapping is more random-like than a linear mapping. As argued in the previous section, branch numbers are important parameters for a diffusion layer.

As already mentioned, the linearity of a mapping is not an absolute notion, but always used in relation to a specific choice of 'linear operation'. A diffusion layer that is linear for one choice will be nonlinear for other choices. However, without the support from linear coding theory, not much can be said about the branch numbers of a mapping. There are no general and efficient methods known to determine the branch numbers of nonlinear mappings, or to construct nonlinear mappings with high branch numbers. The only remaining result is given in the following proposition.

Proposition 7.10 *If a mapping s from $(GF(2^p))^n$ to $(GF(2^p))^n$ is not invertible, then its differential and linear branch number are upper bounded by n.*

Proof: A mapping s is not invertible if and only if there exist at least two different vectors x, y such that $s(x) = s(y)$. Since $w_h(x \oplus y) \leq n$, the differential branch number is bounded by:

$$\mathcal{B}_d(s) \leq w_h(x \oplus y) + w_h(s(x) \oplus s(y)) \leq n + w_h(0) = n\,.$$

A mapping s is not invertible if and only if there exists a non-trivial linear combination of the output bits that is not balanced [72, p. 350]. This means that there exists a β such that $c(0 \cdot x, \beta \cdot s(x)) \neq 0$, and

$$\mathcal{B}_l(s) \leq w_h(0) + w_h(\beta) \leq n\,.$$

■

Other Choices of Differences

Recall the upper bound for the probability of a differential characteristic, given by (7.2):

$$p \leq \delta^B\,.$$

This bound depends on the specific choice of the difference operation Δ in the following way. A different choice for Δ will change δ in an unpredictable way. However it can be seen that if a box is active for one choice of Δ, it will be active for any choice of Δ. Thus, B stays constant. Note that (7.2) assumes that the key addition is affine. If the key addition is not affine (this depends on the choice for Δ) then the probability of a differential characteristic might be even further reduced.

7.4 Key Schedule

The key schedule expands the cipher key k to the round keys $k^r, r = 1, \ldots R$. It is used to achieve the following goals:

- Provide resistance to attacks in which part of the key is known or guessed by the cryptanalyst, and against attacks that recover first a part of a round key, and then use this knowledge to attack the rest of the cipher.
- Provide resistance to related-key attacks. In a related-key attack, the cryptanalyst compares the encryption of the same plaintexts under a set of keys that differ only in a few bits. Related-key attacks can be successful if keys that differ only in a few bits produce round keys that have a small and predictable difference.
- Provide resistance to attacks where the key can be chosen, e.g., if the cipher is used as the compression function of a hash function.
- Remove any symmetry between the rounds by ensuring that the distance between the round keys of any two rounds is large.

- Remove any symmetry in the round transformation.

The first two objectives benefit from a key schedule with high diffusion, the last three ask for a key schedule that behaves irregularly with respect to the components of the round transformation. More generally, the key schedule can be designed in such a way that when some bits of one round key are known, it is not possible to calculate many bits of any of the other round keys. The differential attack on the full DES [10] exploits the fact that this calculation of round key bits can be done quite easily.

The key schedule should also be efficient for applications that need to change the key frequently. At the same time, when the round keys corresponding to a key k are known, the calculation of the round keys for another key k' should not be more efficient than the original key schedule, so that a cryptanalyst who performs an exhaustive key search gains no obvious advantages. Two kinds of key schedules can be distinguished.

Pseudo-Random

The cipher key is used to seed a pseudo-random noise (PRN) generator. The round keys are the output of the PRN generator. Examples are RC5 [115], CAST (some versions) [4], Blowfish [121] and SHARK [111]. The latter two use the encryption algorithm itself as a PRN generator.

This scheme has several advantages. It is very easy to make the key length variable. The relation between the cipher key and the round keys is quite complex. This removes symmetry and provides resistance against the aforementioned attacks, in the sense that it becomes almost impossible for a cryptanalyst to describe the dependencies between the round keys of different rounds or the round keys derived from different cipher keys. It is, however, equally difficult for the designer to 'prove' the resistance.

The most important disadvantages are that the schemes are slow and that the round keys cannot be generated during the encryption process, which can pose a problem for implementations on processors with a very limited amount of memory (such as smart cards).

Key Evolution

The basic construction takes the cipher key as the first round key. The keys of the next rounds are derived from the previous round key by means of a transformation ψ, called the *key evolution*:

$$\begin{aligned} k^0 &= k \\ k^i &= \psi(k^{i-1}) \,. \end{aligned}$$

A more general construction uses an initial transformation φ, state variables κ^i, and a selection function ϕ. The state variables are produced from the cipher key by means of the transformations φ and ψ. The round keys are produced by selecting bits from the state variables; as follows:

$$\begin{aligned} \kappa^0 &= \varphi(k) \\ \kappa^i &= \psi(\kappa^{i-1}) \\ k^i &= \phi(\kappa^i)\,. \end{aligned}$$

The transformations ψ, φ can take many forms: bit permutations (e.g., the DES [39]), rotations (e.g., IDEA [69], LOKI [17]), addition with a round constant (e.g., THREEWAY [24]), or more general linear (e.g., SQUARE [27]) or nonlinear functions (e.g., SAFER [75], CAST [3] (other versions)). In order to counter related-key attacks, the transformations ψ and φ have to be invertible.

The advantages of this scheme are that the key evolution ψ and the selection ϕ can be made fast and simple, hence the round keys can be generated as the encryption proceeds. Similarly to Section 7.2, coding theory can be used to prove properties of the key schedule. Consider the code $\mathcal{C}_k$, defined as

$$\mathcal{C}_k = \{(\phi(\varphi(k))\|\phi(\psi(\varphi(k)))\|\phi(\psi(\psi(\varphi(k)))\|\dots)\}\,. \tag{7.11}$$

The distance of $\mathcal{C}_k$ gives the minimum difference between the round keys of two different cipher keys.

A disadvantage of the key evolution scheme is that the structure may also be exploited in attacks.

7.5 Trapdoors

A trapdoor cipher contains some hidden structure; knowledge of this structure allows a cryptanalyst to obtain information on the key or to decrypt certain ciphertexts; without this trapdoor information, the block cipher seems to be secure. Researchers have been wary of trapdoors in encryption algorithms ever since the DES [39] was proposed in the seventies [124]. In spite of this, no one has been able to show how a practical block cipher with a trapdoor can be constructed. For most current block ciphers it is relatively easy to provide strong evidence for the non-existence of full trapdoors. A *full* trapdoor is defined to be some secret information which allows a cryptanalyst to obtain knowledge of the key by using a very small number of known plaintexts, no matter what these plaintexts are, or what the key is.

This section presents several methods of constructing block ciphers with a *partial* trapdoor, i.e. a trapdoor that does not necessarily work for all keys, or

that gives a cryptanalyst only partial information on the key [114]. It is demonstrated that for certain block ciphers, trapdoors can be built-in that make the cipher very susceptible to linear cryptanalysis; however, finding these trapdoors can be made very hard, even if the general form of the trapdoor is known. A trapdoor is said to be *detectable* (*undetectable*) if it is computationally feasible (infeasible) to find even if the general form of the trapdoor is known.

Finally, it is demonstrated how such a trapdoor can be used to design a public key encryption scheme based on a conventional block cipher.

7.5.1 Trapdoor $m \times n$ S-boxes

In this section the construction and hiding of trapdoors in S-boxes is discussed.

Construction

The construction starts with an $m \times (n-1)$ S-box $S(x)$. The $n-1$ component functions f_i, $i = 1, \ldots, n$, $i \neq q$ are selected randomly (or following an arbitrary design criterion). Parameter q can take any value between 1 and n. The trapdoor $m \times n$ S-box $T(x)$ is derived from $S(x)$ by adding an extra function in the following way. An n-bit Boolean vector β is chosen with $\beta_q = 1$, then f_q is chosen such that

$$f_q(x) = \bigoplus_{i=1, i \neq q}^{n} \beta_i \cdot f_i(x)\,, \tag{7.12}$$

with probability p_T. Now,

$$\beta \bullet T(x) = 0 \tag{7.13}$$

holds with probability $p_T(\beta)$. This is equivalent to a correlation

$$c_T(\beta) = 2 \cdot p_T(\beta) - 1$$

between the output bits that are selected by β. The trapdoor information is the vector β.

Hiding the Trapdoor

If the S-box is claimed to be randomly selected according to a uniform distribution from all $m \times n$ S-boxes, then it is not difficult to hide a trapdoor in it. Indeed, for large values of m and n, the function $f_q(x)$ is computationally indistinguishable from a randomly selected one. Firstly, it is proven that this construction in fact introduces only one β-vector with a high correlation value, not accompanied by a range of β-vectors with 'slightly smaller' correlation values. The difficulty of finding this trapdoor vector is then discussed.

Introducing no more than one β with high correlation: Suppose that $S(x)$ is an $m \times (n-1)$ S-box with input-output correlations bounded by λ: for all n-bit vectors γ, $c_S(\gamma) \leq \lambda$. Consider now the $m \times n$ S-box $T(x)$ that results from adding $f_q(x)$ to $S(x)$. For all γ with $\gamma_q = 0$ it holds that

$$c_T(\gamma) = c_S(\gamma) \leq \lambda\,,$$

so only the cases where $\gamma_q = 1$ remain. If $p_T = 1$, then $\beta \bullet T(x) = 0$ and

$$\begin{aligned} \gamma \bullet T(x) &= (\gamma \bullet T(x)) \oplus (\beta \bullet T(x)) \\ &= (\gamma \oplus \beta) \bullet T(x)\,. \end{aligned} \tag{7.14}$$

Since $\gamma_q \oplus \beta_q = 0$, for all $\gamma \neq \beta$,

$$c_T(\gamma) = c_S(\gamma \oplus \beta) \leq \lambda\,. \tag{7.15}$$

In practice, (7.14) holds with probability $p_T < 1$ and (7.15) may not hold. In this case, consider the S-box $T'(x)$ that results from (7.12) if $p_T = 1$. All correlations of $T'(x)$ are less than λ. Thus $T(x)$ can be thought of as being constructed by applying $(1 - p_T) \cdot 2^m$ random changes to one component of $T'(x)$. The probability that these random changes to the random S-box will result in a significant change of λ is very small.

Recovering β If a cryptanalyst suspects a relation of the form (7.13), then the $2^n - 1$ non-zero values of β can be exhaustively examined. For each value of β, verifying p_T requires the computation of a Walsh-Hadamard transform on an m-bit Boolean function [8], which requires $\mathcal{O}(m \cdot 2^m)$ operations. If $(m, n) = (8, 32)$ this is feasible and the trapdoor is detectable, but for $(m, n) = (8, 64)$, this requires about 2^{64} Walsh-Hadamard transformations on 8-bit functions, which is currently quite hard. For $(m, n) = (10, 80)$, an exhaustive search is currently not feasible. The speed of search can possibly be increased by lattice methods (such as LLL [71]) or coding theory techniques, but the applicability of these techniques is still an open problem.

The search for the β-vector that has the highest correlation is equivalent to the problem of determining a parity function in the presence of noise. The Parity Assumption [13] suggests that this problem is probably NP-hard. This classification only deals with the general problem; specific instances might be easier to solve. For instance, if p_T is very close to one, then it is possible to use Gaussian elimination to solve the problem.

Define the n Boolean vectors a^j, $j = 1, \ldots, n$, as $a^j_i = f_j(i)$, $i = 0, \ldots 2^m - 1$. Equation (7.12) can then be translated into

$$\bigoplus_{i=1}^{n} \beta_i \cdot a^i = \delta\,. \tag{7.16}$$

If (7.12) holds with probability one, or $p_T(\beta) = 1$, then $\delta = 0$. In this case the a^i's are linearly dependent and the linear relation between the vectors can be recovered in a very efficient way using Gaussian elimination on (7.16). If the probability of (7.12) is smaller than one, then the vectors a^i are independent; $\delta \neq 0$ is unknown to the cryptanalyst, and the Hamming weight of δ is given by

$$w_h(\delta) = 2^m(1 - p_T) .$$

The cryptanalyst can still try to recover β by guessing a value for δ and solving the set of Equations (7.16). Equation (7.16) will only have a solution when the guess for δ is correct. A better strategy for the cryptanalyst may be to use the following equations:

$$\bigoplus_{i=1}^{n} \beta_i \cdot a^i = \bigoplus_{i=1}^{d} \gamma_i \cdot \delta^i . \tag{7.17}$$

The d vectors δ^i are guessed by the cryptanalyst. If the unknown δ can be expressed as a linear combination of the vectors δ^i, then the cryptanalyst can hope to find the trapdoor by solving (7.17) for β and γ. The probability that δ is a linear combination of the d vectors δ^i increases with d.

If the δ^i vectors are linearly independent, then they generate a vector space of size 2^d. To simplify the discussion, only the case of a large *positive* input-output correlation is considered here. In that case the Hamming weight of δ will be low. Also it will be assumed that all the δ^i-vectors have Hamming weight one. The number of vectors in a d-dimensional space with Hamming weight $\leq D$ is given by

$$\sum_{k=1}^{D} \binom{d}{k} .$$

Table 7.4 shows the numerical values for several choices of D and d.

For example, with a 10×40 S-box, there are 2^{10} inputs. For each input the equations may or may not hold, resulting in a number of $2^{2^{10}}$ possible δ-vectors; 2^{202} of them have Hamming weight ≤ 32. If $d = 64$, then the probability p_{lc} that δ is a linear combination of d randomly chosen δ^i vectors is equal to $2^{63}/2^{202}$. The work factor of this algorithm is determined by p_{lc} and by the work necessary to solve (7.17), which is $\mathcal{O}((2^m + n + d)^3)$ (note that the best asymptotic algorithms reduce the exponent from 3 to 2.376 [20]).

It is possible to increase p_{lc} by increasing d. However, if d becomes larger than a certain threshold value, then spurious solutions for δ will start to appear that have a large Hamming weight. These unwanted solutions correspond to β vectors with low correlation values. This effect limits the use of Gaussian elimination. This algorithm will be be more useful than exhaustive search for β if D and n are small, and m is large.

D	$d = 64$	$d = 128$	$d = 256$	$d = 1024$
1	2^6	2^7	2^8	2^{10}
10	2^{37}	2^{44}	2^{58}	2^{78}
20	2^{55}	2^{68}	2^{98}	2^{139}
32	2^{63}	2^{86}	2^{136}	2^{202}
40	2^{64}	2^{92}	2^{156}	2^{240}

Table 7.4: The number of vectors in a d-dimensional space with Hamming weight $\leq D$.

Bent Functions

The construction method can be extended to deal with additional constraints imposed on the functions $f_i(x)$. For example, in some block ciphers (such as the CAST family [2]), it is necessary that the component functions $f_i(x)$ are bent functions. The Maiorana construction for bent functions [35] can then be used to obtain an S-box satisfying Property (7.13): an m-bit bent function $f(x)$ (m is even) is obtained from an $m/2$-bit permutation $\pi(y)$ and an arbitrary $m/2$-bit function $g(z)$ as follows:

$$f(x) = f(y, z) = \pi(y) \odot z \oplus g(z)\,.$$

Here '$\odot$' denotes multiplication in $GF(2^{m/2})$. If two component functions $f_i(x)$ and $f_j(x)$ are derived from the same permutation $\pi(y)$ then,

$$f_i(y, z) \oplus f_j(y, z) = g_i(z) \oplus g_j(z)\,,$$

which can be chosen arbitrarily close to a constant function. Hiding (7.13) in a bent function based S-box can be done as follows. A β with even Hamming weight is chosen randomly. The set of indices where $\beta_i = 1$ is divided arbitrarily into pairs. For each pair of indices a different mapping $x \mapsto (y, z)$ and a different permutation π are selected. The $m/2$-bit functions $g_i(z)$ are defined, and extended to full m-bit functions by adding zero values. It then follows that

$$\beta \bullet T(x) = \bigoplus_{i=1}^{m} \beta_i \cdot g_i(x) = 0$$

with probability p_T.

This construction shows that it is possible to find a set of bent functions that sum to an almost constant function.

7.5.2 Trapdoor Ciphers

This section proposes several constructions for trapdoors in block ciphers starting from the building blocks, i.e., the round functions.

Trapdoor Round Functions

Trapdoors in S-boxes can be extended to trapdoors in the round function of a Feistel cipher [37]. The round functions of variants on CAST [48] and LOKI91 [17] are considered.

tCAST: The CAST ciphers are described in Chapter 6. Using four S-boxes with the same trapdoor β (but with a different value of c_T, denoted by c_{T^i}),

$$\beta \bullet F(x_1, x_2, x_3, x_4) = \bigoplus_{i=1}^{4} \beta \bullet T_i(x_i) \, .$$

Hence some output bits of the round function have a correlation

$$c_F = c_{T_1} c_{T_2} c_{T_3} c_{T_4} \, .$$

As previously mentioned, 8×32 S-boxes can be checked for this type of trapdoor. However, if CAST is extended in a natural way to an 128-bit block cipher by using 8×64-bit S-boxes, then finding this trapdoor becomes very difficult. The technique can be extended to CAST variants where the exor operation is replaced by a modular addition or multiplication.

tLOKI: The expansion in the round function of LOKI91 [17] allows for a subtle trapdoor, not visible in the individual S-boxes, but only in the round function.

The round function of LOKI91 uses the same 12×8 S-box four times, and is defined as:

$$F(x_1, \ldots, x_{32}) = P(S(x_{29}, x_{30}, x_{31}, x_{32}, x_1, \ldots, x_8) \,\|\, S(x_5, x_6, \ldots, x_{16}) \,\|\, S(x_{13}, x_{14}, \ldots, x_{24}) \,\|\, S(x_{21}, x_{22}, \ldots, x_{32})) \, .$$

In this analysis the bit permutation P is not relevant and will be ignored. Since some of the bits are used as input to the S-boxes twice, it is possible to hide a trapdoor in this round function. The trapdoor involves nonlinear functions of the bits that are used twice. Note that the use of nonlinear relations together with a linear relation has already been studied by L.R. Knudsen and M. Robshaw [64]. Let $a^1(x)$, $a^2(x)$, $a^3(x)$, and $a^4(x)$ be four 8-bit (nonlinear)

Boolean functions and $\beta = \beta_1 \| \beta_2 \| \beta_3 \| \beta_4$ a 32-bit Boolean vector. Suppose the following relations hold with probabilities p_1, p_2, p_3, p_4 respectively:

$$\begin{aligned}
\beta_1 \bullet S(x_1, \ldots x_{12}) &= a^1(x_1, x_2, x_3, x_4) \oplus a^2(x_9, x_{10}, x_{11}, x_{12}) \\
\beta_2 \bullet S(x_1, \ldots x_{12}) &= a^2(x_1, x_2, x_3, x_4) \oplus a^3(x_9, x_{10}, x_{11}, x_{12}) \\
\beta_3 \bullet S(x_1, \ldots x_{12}) &= a^3(x_1, x_2, x_3, x_4) \oplus a^4(x_9, x_{10}, x_{11}, x_{12}) \\
\beta_4 \bullet S(x_1, \ldots x_{12}) &= a^4(x_1, x_2, x_3, x_4) \oplus a^1(x_9, x_{10}, x_{11}, x_{12}) \,.
\end{aligned}$$

The trapdoor is based on the fact that the nonlinear functions are all used twice with the same input bits, and thus cancel out. The correlation between the output bits selected by

$$\begin{aligned}
\beta \bullet F(x_1, \ldots, x_{32}) = \beta_1 \bullet S(x_{29}, \ldots, x_8) \;\oplus\; \beta_2 \bullet S(x_5, \ldots, x_{16}) \\
\oplus\; \beta_3 \bullet S(x_{13}, \ldots, x_{24}) \;\oplus\; \beta_4 \bullet S(x_{21}, \ldots, x_{32}) \,,
\end{aligned}$$

is now given by $(2p_1-1)(2p_2-1)(2p_3-1)(2p_4-1)$. For the parameters of LOKI91 this is probably a detectable trapdoor, at least for someone who knows what to look for. Again, larger block sizes and S-boxes would make such trapdoors harder to detect.

Trapdoor Ciphers

The trapdoor round functions defined above can be used to construct a trapdoor cipher. The resulting cipher will have iterative linear relations that approximate the output of every other round. For a cipher with R rounds, about $R/2$ round approximations are required.

For example, consider a version of tCAST with 16 rounds, block size 80 bits, and using four 10×40 S-boxes. If $p_T = 1 - 2^{-5}$ then the round key of the first and the last round can be recovered using a linear attack [80], using approximately 875 known plaintexts. Since the Hamming weight of δ is 32, the Gaussian elimination technique to find the trapdoor will not work faster than exhaustive search.

7.5.3 Extensions

The trapdoors considered here, all use 'type II' linear relations, as defined in [79]: correlations that exist between the output bits of the round function. It is also possible to hide 'type I' linear relations: correlations between input and output bits of the round function. For example, S-boxes can be constructed such that

$$\begin{aligned}
&& \beta \bullet S(x) &= \alpha \bullet x && (A) \\
\text{and} && \alpha \bullet S(x) &= \beta \bullet x && (B) \,,
\end{aligned}$$

with high probability. It is easy to see that these relations can be concatenated in the following way: $AB - BA - AB - \ldots$ The main advantage of this type of relation is that there are more of them than the number of type II relations: 2^{n+m} instead of 2^n. If $(m, n) = (8, 32)$, as in CAST, then there are already 2^{40} cases to verify.

When building the trapdoor into the round function of tLOKI, the fact that in LOKI91 the key is added before the expansion is used. In the DES, the key is added after the expansion; in this case trapdoors can also be introduced. A first approach consists of choosing the functions $a^i(x)$ as linear functions. In this way the absolute value of the correlation between bits is independent of the key. However this imposes a severe restriction on the number of possible trapdoors, which makes them easy to detect. (The DES was checked for these trapdoors, but none was found.) Another option is to hide several key dependent trapdoors. The key schedule could be carefully adapted in such a way that only a small number of key bits have an influence.

In a similar way, differentials can be hidden in block ciphers, in order to make them vulnerable to differential cryptanalysis [10]. However, exploitation of such trapdoors requires chosen, rather than known, plaintexts, which is much less practical.

7.5.4 Public Key Encryption

Besides the obvious use by government agencies for law enforcement purposes, trapdoor block ciphers can also be used for public key cryptography. For this application, a block cipher with variable S-boxes is selected and made widely available (it is a system-wide public parameter). Bob generates a set of S-boxes with a secret trapdoor. These S-boxes form his public key. If Alice wants to send a confidential message to Bob, then she generates a random session key, encrypts her message and a fixed set of plaintexts, and sends the ciphertexts to Bob. The set of plaintexts can be fixed, or can be generated from a short seed using a pseudo-random bit generator. Bob uses the trapdoor and the known plaintexts to recover the session key and decrypts the message.

There seems to be no obvious way to extend this construction to digital signatures.

7.6 Conclusions

The Wide Trail strategy of [26] was developed to design cryptographic algorithms that resist linear and differential cryptanalysis. In this chapter the strategy was extended and construction methods for the different components were developed.

The framework of linear codes over a finite field $GF(2^p)$ enables the efficient construction of diffusion layers with strong diffusion properties. A code $\mathcal{C}_\theta$ can be associated with every mapping θ, where

$$\mathcal{C}_\theta = \{(x \| y) \mid y = \theta(x)\}.$$

If the mapping θ corresponds to a matrix multiplication,

$$\theta(x) = x \cdot A,$$

then the associated code $\mathcal{C}_\theta$ is a linear code. The distance of $\mathcal{C}_\theta$ is equal to the differential branch number of θ. The distance of the dual code of $\mathcal{C}_\theta$ is equal to the linear branch number of θ. If $\mathcal{C}_\theta$ is a Maximum Distance Separable code, then θ has optimal diffusion properties. Reed-Solomon codes are an example of MDS-codes that can be constructed very easily and have the typical dimensions that are required for the diffusion layer of a block cipher. Mappings with optimal diffusion are always complete mappings. The diffusion of incomplete mappings can never be as fast as for MDS-codes. Multi-level diffusion constructions are a means of combining incomplete mappings in such a way that over a number of rounds that depends on the level of diffusion a well-defined amount of diffusion can be guaranteed.

The construction of nonlinear S-boxes has already received a great deal of attention in the cryptographic literature. Algebraic constructions have the disadvantage that they exhibit internal structure which might be exploited in an interpolation attack. This structure can be removed by applying affine mappings to the output, which conserve the nonlinear properties, or by applying nonlinear modifications, which usually weaken the nonlinear properties. It is also possible to generate S-boxes at random and test them against some nonlinearity criteria. However, it remains an open question as to whether S-boxes generated in this way exhibit less structure than (modified) algebraic constructions.

The key schedule originally had no place in the Wide Trail strategy. However, it is necessary to include it in order to provide resistance to related-key attacks, or when considering the case that a block cipher is used as the compression function of a hash algorithm. Two approaches are possible. The first approach uses the key as a seed for a pseudo-random noise generator; the round keys are derived from the output of the PRN generator. This construction is difficult to analyse and hopefully also difficult to cryptanalyse. The second approach is to derive each round key from the key by means of a few affine operations, rotations and permutations. This approach has two advantages: the key setup time is usually much smaller, and the theory that was developed for the diffusion layers can also be applied here, e.g., to find the minimum distance between round keys derived from a different key.

The last section of this chapter discussed a way of hiding trapdoors in random-looking S-boxes. The basic construction introduced a high correlation

between some of the output bits of the S-boxes. If the dimensions of the S-boxes are sufficiently large (e.g., 10×80), then it is impossible to detect the trapdoor. A second method uses the expansion function of the round transformation to hide a trapdoor. The conclusion is that a user cannot trust a random S-box that has been generated by someone else, unless the generation process is clearly explained. If the algorithm as a whole is secret, like Skipjack, then the problem is even worse. The results of this section have been published in [114].

Chapter 8

Block Cipher Proposals

It is your job to devise a code that is so difficult that your opponent cannot break it. At the same time, you try to break your opponent's code, using the minimum number of moves.

This chapter presents two block ciphers, published in [111, 27, 28]. The ciphers are constructed following the Wide Trail design strategy (cf. Chapter 7 and [26]), and in the light of the remarks from Chapter 7.

8.1 SHARK

SHARK [111] is an iterated block cipher. It uses highly nonlinear S-boxes and a Reed-Solomon code to guarantee a good diffusion. The cipher resists linear and differential cryptanalysis after a small number of rounds. SHARK is oriented towards 64-bit architectures.

8.1.1 Structure

SHARK has a block length of 64 bits, but it can easily be extended to larger block lengths. The key length is variable; it is advised to use a key length between 64 and 128 bits. SHARK is not a Feistel cipher, but uses a uniform round transformation, consisting of three operations, selected using the Wide Trail strategy: a nonlinear substitution γ, a linear diffusing layer θ, and an affine key addition σ. The arrangement of the three operations is shown in Figure 8.1.

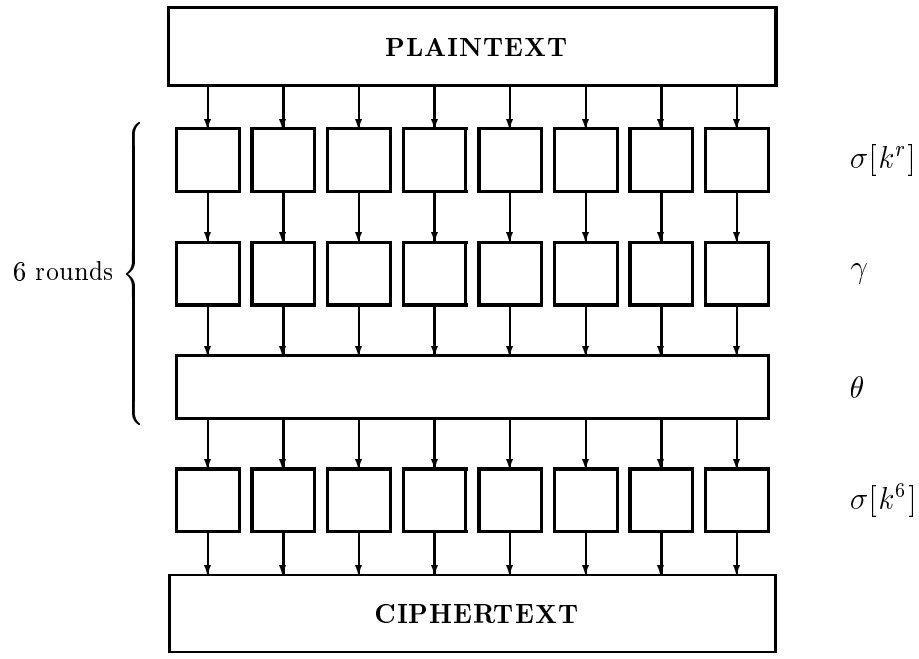


Figure 8.1: The structure of SHARK. The cipher uses a linear key addition $\sigma[k^r]$, a nonlinear substitution γ and a linear diffusion layer θ.

The nonlinear substitution γ

SHARK uses eight instances of the same 8×8 invertible S-box, based on the mapping $s(x) = x^{-1}$ over $GF(2^8)$ and given by:

$$\gamma(x) = (S[x_1], S[x_2], \ldots, S[x_8])\,.$$

The maximum entry in the exor-table of the S-box is four (except from the trivial entry at $(0,0)$), or $\delta = 2^{-6}$ and the maximal correlation $\lambda = 2^{-3}$. This is the only construction from Section 7.3.1 that can be used to produce invertible mappings with the given dimensions. In order to get a more complicated description of the mapping and to distort other internal structure caused by the construction method, the output bits are transformed by an invertible affine transformation in $(GF(2))^8$.

The diffusion layer θ

The diffusion layer θ is a linear mapping from $(GF(2^8))^8$ to $(GF(2^8))^8$. It is based on a $[16, 8, 9]$-Reed-Solomon-code: θ can be written as a matrix multiplication,

$$\theta(x) = x \cdot A, \tag{8.1}$$

where the matrix A is the right hand side of the standard form of the generator matrix of the RS-code. As explained in Section 7.2.3, this construction guarantees that the branch number of θ is 9.

The key addition $\sigma[k^r]$

The basic version of SHARK exors the 64-bit round key with the round input. If c_k denotes the vector of 8-bit components that is defined by k, then the key addition is given by

$$\sigma[k](x) = x \oplus c[k]\,.$$

This method is fast and uniform and there are no weak keys, in the sense that the nonlinear layer and the diffusion layer have the same properties for all keys.

There is also an extended version of SHARK where the round keys have 128 bits. In this version every key k corresponds to an 8 by 8 diagonal matrix $B[k]$ and a vector $c[k]$. The key addition is then defined as:

$$\sigma[k](x) = x \cdot B[k] \oplus c[k]\,.$$

The matrix $B[k]$ has to be invertible. If this is accounted for, there are no weak keys. The advantage of this method is that the round key space is much larger;

this provides resistance against attacks where one round key has to be guessed by the cryptanalyst. The computational overhead, however, is significant. In Section 8.1.2 it is explained how to implement this variant in an efficient way.

Properties

In the next section the following properties of the key addition and the diffusion operation are used to develop an efficient implementation:

1. For every key k there is a key l such that $\sigma[k]^{-1} \equiv \sigma[l]$. This key l is given by $B[l] = B[k]^{-1}$ and $c[l] = c[k]$, and is denoted k^{-1}.

2. The order of the key addition and the diffusion operation can be interchanged:

$$\begin{aligned} \theta \circ \sigma[k](x) &= (x \cdot B[k] \oplus c[k]) \cdot A \\ &= x \cdot A \cdot A^{-1} \cdot B[k] \cdot A \oplus c[k] \cdot A \\ &= \sigma[t_\theta(k)] \circ \theta(x)\,. \end{aligned} \tag{8.2}$$

   The operation t_θ is defined as follows:

$$t_\theta(k) = l \Leftrightarrow B[l] = A^{-1} \cdot B[k] \cdot A \text{ and } c[l] = c[k] \cdot A\,.$$

3. If $B[k]$ is the identity matrix, then $k^{-1} = k$ and $t_\theta(k) \equiv \theta(k)$.

Key Schedule

SHARK uses a key schedule based on a pseudo-random noise generator (cf. Section 7.4). The cipher key is concatenated with itself until it has a length of $7 \cdot 64$ bits, or $7 \cdot 128$ bits for the extended version. This string is encrypted with SHARK in CFB-mode [52], using a fixed key. The first 448 bits of the output form the round keys $c[k^r]$. For the extended version, the next 448 bits are used to form the diagonal elements of the matrices $B[k^r]$. If one of these elements is zero, then it is discarded and all the following values are shifted down one place. An extra encryption of the all-zero string is added at the end to provide the extra diagonal elements. The fixed key used during the key schedule is formed in the following way. The matrices $B[k^r]$ are equal to the identity matrix. The vectors $c[k^r]$ are the first 7 entries of the expanded substitution table T_0, that will be defined in Section 8.1.2.

The Cipher SHARK

The round transformation of SHARK is denoted by $\rho[k]$ and given by:

$$\rho[k] = \theta \circ \gamma \circ \sigma[k]\,.$$

Figure 8.1 shows that SHARK consists of 6 rounds, followed by an extra key addition and an extra diffusion layer, which is the inverse of the round diffusion layers. The encryption function is given by

$$\text{SHARK}[k] = \theta^{-1} \circ \sigma[k^6] \circ \rho[k^5] \circ \rho[k^4] \circ \cdots \circ \rho[k^0]\,. \tag{8.3}$$

The purpose of the extra key addition is to prevent a cryptanalyst from peeling off the last round. The extra diffusion layer is required to make the structure of the decryption similar to the structure of the encryption. Section 8.1.2 explains how the different operations can be combined in table-lookups, in order to optimise the performance.

8.1.2 Implementation

At first, the complexity of the diffusion operation of SHARK seems to be a serious drawback. Instead of a simple permutation or a few exor operations, a matrix multiplication is required to calculate the output of the diffusion layer. This section shows how the diffusion operation can be combined in an efficient way with the S-boxes.

Let the row vector $x = (x_1, \ldots, x_8)$ represent the input of a round, and y the output. Layer θ maps x to $x \cdot A$. The entries of the matrix A are denoted by $a_{i,j}$, a row is denoted by a_i. The combined operation $\theta \circ \gamma$ can then be written as:

$$\begin{aligned} y &= (S(x_1), S(x_2), \ldots, S(x_8)) \cdot A \\ &= S(x_1) \cdot a_1 \oplus S(x_2) \cdot a_2 \oplus \cdots \oplus S(x_8) \cdot a_8\,. \end{aligned} \tag{8.4}$$

Here '$\oplus$' is used to denote addition of elements of $GF(2^8)$ and of vectors of $(GF(2^8))^8$; '$\cdot$' is used to denote multiplication in $GF(2^8)$ and the scalar multiplication of a vector by an element of $GF(2^8)$. We now define the expanded substitution tables $T_i(x)$:

$$\begin{aligned} T_i(x) &= S(x) \cdot a_i \\ &= (S(x) \cdot a_{i,1}, S(x) \cdot a_{i,2}, \ldots, S(x) \cdot a_{i,8})\,. \end{aligned}$$

Equation (8.4) then becomes:

$$y = T_1(x_1) \oplus T_2(x_2) \oplus \cdots \oplus T_8(x_8)\,. \tag{8.5}$$

This operation requires only eight table lookups and seven exor operations (of 64-bit values). The memory requirements increase: the eight T-tables each have 2^8 entries of eight bytes. This sums to 32 kilobytes, which should be compared with a straightforward implementation of the operations, where $S(x)$ and A take only 320 bytes.

The key addition $\sigma[k^r]$ can also be incorporated into the S-boxes. This can be done by defining the key dependent tables

$$U_i^r(x) = T_i(\sigma[k^r](x)) .$$

Since every round has different U-tables, the memory requirements increase by a factor of six. If the key addition is a simple exor, then the U-tables are formed by a simple rearrangement of the rows of the T-tables.

If the key addition includes a key dependent affine transformation and the U-tables are not calculated beforehand, then the round operation becomes very slow.

8.1.3 Inverse Cipher

One of the properties of the Feistel structure is that, for any choice of the round function, the encryption mode and the decryption mode of the cipher differ only in the ordering of the round keys. Block ciphers that use a uniform round transformation lose this general property. However for SHARK, the *structure* of the decryption algorithm is equal to the structure of the encryption algorithm. The used components however, are different. Note that this situation differs from the block cipher IDEA, where encryption and decryption have identical *structure and components*.

The following analysis demonstrates how the components for the decryption mode can be derived from the encryption mode components. For the sake of simplicity, the number of rounds is reduced to two. The encryption operation is then given by:

$$y = \theta^{-1} \circ \sigma[k^2] \circ \theta \circ \gamma \circ \sigma[k^1] \circ \theta \circ \gamma \circ \sigma[k^0](x) . \tag{8.6}$$

In Section 8.1.2 it was explained how $\theta \circ \gamma$ can be combined into one efficient operation. An efficient implementation of SHARK will use a slightly different round operation for the last round. Application of (8.2) to (8.6) results in:

$$y = \sigma[t_\theta^{-1}(k^2)] \circ \gamma \circ \sigma[k^1] \circ \theta \circ \gamma \circ \sigma[k^0](x) . \tag{8.7}$$

In an implementation that follows this formula, the additional inverse diffusion layer does not cause any overhead.

The decryption operation is given by:

$$x = \sigma[(k^0)^{-1}] \circ \gamma^{-1} \circ \theta^{-1} \circ \sigma[(k^1)^{-1}] \circ \gamma^{-1} \circ \theta^{-1} \circ \sigma[(k^2)^{-1}] \circ \theta(y) . \tag{8.8}$$

Application of (8.2) to (8.8) results in:

$$x = \sigma[(k^0)^{-1}] \circ \gamma^{-1} \circ \sigma[t_{\theta^{-1}}((k^1)^{-1})] \circ \theta^{-1} \circ \gamma^{-1} \circ \sigma[t_{\theta^{-1}}((k^2)^{-1})](y) . \tag{8.9}$$

This equation has the same structure as (8.7) where γ and θ are replaced by γ^{-1} and θ^{-1}, and the round keys are different.

8.1.4 Cryptanalysis

Linear and differential cryptanalysis

SHARK was designed according to the principles of the Wide Trail strategy (cf. Chapter 7) in order to make it resistant to linear and differential cryptanalysis. Table 8.1 gives the upper bounds for the probability of an R-round differential characteristic and the input-output correlation (squared) after R rounds, using Formulae (7.2) and (7.3). These values are compared to the corresponding values for the DES. Since the DES is a Feistel cipher, a fair comparison can only be made if the number of rounds in the DES is doubled.

SHARK			DES		
R	p (dc)	c^2 (lc)	R	p (dc)	c^2 (lc)
2	2^{-54}	2^{-54}	4	$2^{-9.6}$	2^{-6}
4	2^{-108}	2^{-108}	8	$2^{-30.5}$	$2^{-19.5}$
6	2^{-162}	2^{-162}	12	$2^{-46.2}$	$2^{-33.5}$
			48	2^{-128}	2^{-148}

Table 8.1: Probabilities for the best differential characteristics and linear approximations as a function of the number of rounds, calculated with Formulae (7.2) and (7.3).

Note that an attack on an R-round scheme does not necessarily require an R-round differential characteristic. It can be assumed that for a differential attack on R rounds of SHARK, a characteristic of at least $R-2$ rounds will be required. The same remark has to be made for a linear attack. Also, the probability of the best differential can be several times higher than the probability of the best characteristic. Equivalently, the correlation between input bits and output bits of the cipher is only approximated by the product of the correlations in each round. When the probability of a differential characteristic or the correlation between a linear combination of input bits and a linear combination of output bits drops below 2^{-63}, it can be considered as irrelevant.

Therefore the values of Table 8.1 can only be used as an indication of the safety margin against linear and differential attacks. For applications that require only 40 bits security, four rounds may suffice. For applications where a conservative security margin is much more important than encryption speed, eight or ten rounds can be used. According to Table 8.1, eight rounds of SHARK give a security level that is comparable to triple-DES (assuming that a characteristic covering $R-2$ rounds is necessary for an attack).

Interpolation attack

The danger of algebraic structures present in the design of a block cipher is demonstrated by the interpolation attack of T. Jakobsen and L.R. Knudsen [54]. They consider a modified version of SHARK, that has S-boxes based on $s(x) = x^{-1}$, but without the affine transformation of the output bits. The relation between round outputs and round inputs is then given by:

$$y_j = \bigoplus_{i=1}^{8} (c_i \oplus x_i)^{-1} a_{i,j} \, .$$

More generally, the relation between the outputs after r rounds and the inputs can be expressed as

$$y_j = \frac{p_1(x_1, \ldots, x_8)}{p_2(x_1, \ldots, x_8)} \, , \tag{8.10}$$

where p_1 and p_2 are polynomials over $GF(2^8)$. It can be shown [54] that the number of unknown coefficients in p_1 and p_2 after R rounds is $2(8^{R-1}+1)^8$. The complexity of p_1 and p_2 is determined by the number of diffusion layers encountered. The coefficients can be determined by an extension of the Lagrange interpolation formula to multivariate polynomials [120]. Since r rounds of SHARK count only $R-1$ diffusion layers, the number of coefficients is actually only $2(8^{R-2}+1)^8$. Given an equal number of known plaintexts it is possible to build and solve the set of equations

$$p_2(x_1, \ldots, x_8) y_j \oplus p_1(x_1, \ldots, x_8) = 0,$$

which are linear in the coefficients of p_1 and p_2. Solving a set of u linear equations requires $\mathcal{O}(u^2)$ memory locations and $\mathcal{O}(u^3)$ operations.

Equation (8.10) can be used to determine the round key of the last round in the following way. The cryptanalyst guesses the byte of the last round key that is added to the output byte y_i from the last round. Given the ciphertext it is then possible to compute one output byte of the last round. Subsequently $2(8^{R-3}+1)^8$ known plaintexts are used to determine p_1 and p_2. Once p_1 and p_2 are determined, a few additional plaintexts are used to verify whether (8.10) holds. If it does not hold then the key byte guess was wrong and the attack has to be repeated using another guess.

The attack can be optimised using a meet-in-the-middle approach. The meet-in-the-middle attack reduces the complexity of the attack by searching for a round key of a round in the middle of the cipher, instead of for the last round. Briefly, R-round SHARK can be split up into two parts:

$$\begin{aligned} y &= g \circ f(x) \\ \text{and} \qquad g^{-1}(y) &= f(x), \end{aligned}$$

where f and g correspond approximately to R_1 and r_2 rounds of SHARK ($R_1 + R_2 = R$). Six-round SHARK involves five effective diffusions, which have to be divided as evenly as possible over f and g. There exist then polynomials p_1, p_2, p_3, p_4 such that

$$\begin{aligned} \frac{p_1(x_1,\ldots,x_8)}{p_2(x_1,\ldots,x_8)} &= \frac{p_3(y_1,\ldots,y_8)}{p_4(y_1,\ldots,y_8)} \\ p_1(x_1,\ldots,x_8)p_4(y_1,\ldots,y_8) &= p_2(x_1,\ldots,x_8)p_3(y_1,\ldots,y_8)\,. \end{aligned}$$

The last equation has $2(8^{R_1-1}+1)^8(8^{R_2-2}+1)^8$ unknown coefficients (the difference in the exponents comes from the fact that the last diffusion layer of f does not vanish). Table 8.2 gives the complexities for versions of SHARK with three to six rounds. In a chosen plaintext attack, some of the bytes of the input can be fixed. This allows the complexity of the attack to be decreased, but if too many bytes are fixed, not enough plaintexts are left. If c bytes are fixed, then the number of unknown coefficients is given by $2(8^{R_1-1}+1)^{8-c}(8^{R_2-2}+1)^8$. The solving of the set of equations dominates the memory requirements and the workload of the attack.

# rounds	type	# texts	memory	workload
3	chosen	2^{11}	2^{22}	2^{33}
3	known	2^{17}	2^{34}	2^{51}
4	chosen	2^{22}	2^{44}	2^{66}
4	known	2^{35}	2^{70}	2^{105}
5	chosen	2^{39}	2^{78}	2^{117}
5	known	2^{52}	2^{104}	2^{156}
6	known	2^{75}	2^{150}	2^{225}

Table 8.2: Complexities for the meet-in-the-middle interpolation attack.

Structure attack

SHARK reduced to three rounds is vulnerable to a dedicated attack that exploits the structure of the cipher. The attack was originally developed for SQUARE in [27], but it is also applicable to SHARK. The attack is independent of the choice of S-boxes, the key scheduling, and the particular choice of θ (as long as it has a differential branch number of 9). It is a chosen plaintext attack, using structures of 256 plaintexts that have seven constant bytes and one byte that takes every value from 0 to 255 exactly once. Such a structure is called a Λ-set.

Definition 8.1 *Let λ be a set of indices. A $\Lambda(\lambda)$-set is a set of 256 vectors*

such that

$$\forall x, y \in \Lambda \begin{cases} x_i \neq y_i & \text{if } i \in \lambda \\ x_i = y_i & \text{else} \end{cases} .$$

Since $\sigma[k]$ and γ operate on each byte separately, application of $\sigma[k]$ or γ to a Λ-set gives another Λ-set with the same λ. Also, it is easy to see that applying θ to $\Lambda(\{i\})$ results in a set of the type $\Lambda(\{1, 2, \ldots, 8\})$.

The attack starts with a $\Lambda(\{i\})$-set. After one round of encryption, the set becomes a $\Lambda(\{1, 2, \ldots, 8\})$-set. In the second round, σ and γ leave the set invariant. The operation θ destroys this property. However, the sum of the values that a byte takes over all texts of the set remains zero, since every byte y_i of the output of θ is a linear combination of the components of the input vector x: $y_i = \bigoplus_{j=1}^{8} x_j a_{i,j}$. The sum over all values in the set is then given by:

$$\bigoplus_{l=0}^{255} y_i = \bigoplus_{l=0}^{255} \bigoplus_{j=1}^{8} a_{i,j} x_j = \bigoplus_{j=1}^{8} a_{i,j} \bigoplus_{l=0}^{255} x_j = \bigoplus_{j=1}^{8} a_{i,j} 0 = 0 . \qquad (8.11)$$

In the third round σ conserves Property (8.11), but γ does not. If the third round is the last round, the θ operation is canceled by the θ^{-1} operation at the end. The cryptanalyst guesses a value for one byte of the key in the last key addition (a byte of $t_\theta^{-1}(k^3)$ is guessed, cf. (8.7)). The last σ and γ operations are inverted for the corresponding ciphertext byte for all texts in the set and a check is made as to whether (8.11) is fulfilled. If it is, the guessed key value is correct with high probability. A second Λ-set can be used to make sure that every byte of the last round key is determined uniquely. The cryptanalyst can reuse the chosen plaintexts to determine all bytes of the key.

On SHARK reduced to three rounds, the attack requires 2^9 chosen plaintexts and a memory of size 2^8. It has a workload of approximately 2^{17} encryptions. There seems to be no obvious way to extend the attack to more than three rounds.

8.1.5 Performance

Since SHARK operates on 64-bit words, it will benefit from a 64-bit architecture. Table 8.3 compares the performance of implementations of SHARK, SAFER and IDEA on a 266 MHz DEC–Alpha and on a 90 MHz Pentium. The Alpha implementations are written in C, the implementations on the Pentium are partially written in assembler. On a Pentium ,SHARK runs at approximately the same speed as SAFER. Experiments with smaller S-boxes show that this degradation of performance is due to the limited on-chip cache size. On a Pentium II, which has a double cache size, SHARK will run at a speed of 5.4 Mbyte/s, assuming

the same clock speed of 90 MHz. With a more realistic clock speed of 166 MHz this scales up to 10 Mbyte/s [15].

	ALPHA	Pentium
SHARK	6.30 Mbyte/s	1.23 Mbyte/s
SAFER	1.03 Mbyte/s	0.725 Mbyte/s
IDEA	1.53 Mbyte/s	1.22 Mbyte/s

Table 8.3: Performance of SHARK, SAFER and IDEA on a 64-bit workstation, and on a Pentium.

8.2 SQUARE

SQUARE [27, 28] is an iterated block cipher. The structure of the cipher is designed to permit efficient implementations on a wide range of processors. The different transformations have been chosen to optimise resistance to differential and linear cryptanalysis. The main difference between SQUARE and SHARK is that the linear transformation of the former is more suitable for implementation on smaller processors. A consequence is that the diffusion in SQUARE is slower; therefore the number of rounds is increased.

8.2.1 Structure

SQUARE has a block length and a key length of 128 bits. However, its modular design approach allows extensions to larger block lengths in a straightforward way. The cipher has a uniform round transformation, composed of four distinct transformations that are selected according to the Wide Trail strategy. Section 8.2.2 shows how these four transformations can be combined into a single set of table-lookups and exor operations.

To make the description of SQUARE more compact, the input of the transformations is represented by a 4×4 matrix of bytes. The element of an input X in row i and column j is specified as $x_{i,j}$. Both indices start from 0. The 32-bit value that is formed by row i of the matrix A is denoted by x_i. Figure 8.2 shows three of the round operations.

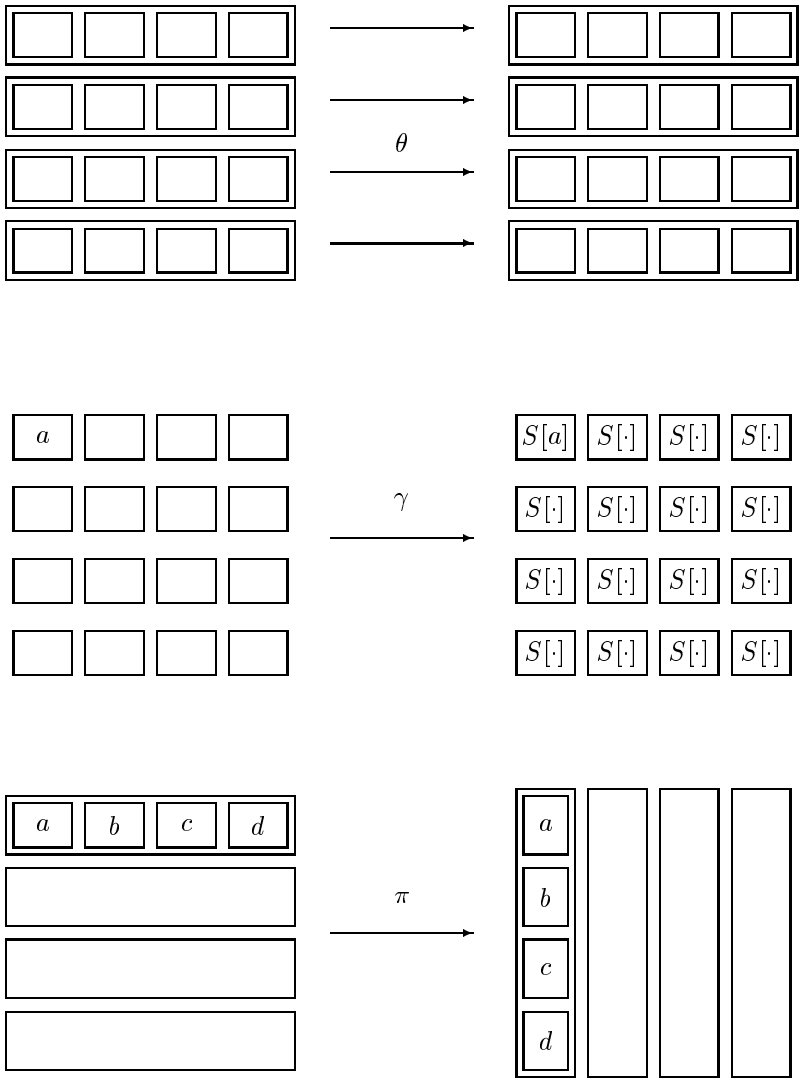


Figure 8.2: Geometrical representation of the basic operations of SQUARE. Operation θ consists of 4 parallel linear diffusion mappings, γ consists of 16 separate substitutions, and π is a transposition.

The Linear Transformation θ

Operation θ is defined as a matrix multiplication (over $GF(2^8)$): $\theta : X \mapsto Y = X \cdot A$, where A is a circulant matrix:

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_3 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_0 \end{bmatrix} .$$

It can be seen that θ operates separately on each of the four rows of its input, since

$$y_{i,j} = a_j x_{i,0} \oplus a_{j-1} x_{i,1} \oplus a_{j-2} x_{i,2} \oplus a_{j-3} x_{i,3},$$

where the indices of a must be taken modulo 4. The linear code associated with θ has generator matrix

$$G = \begin{bmatrix} A & 0 & 0 & 0 \\ 0 & A & 0 & 0 \\ 0 & 0 & A & 0 \\ 0 & 0 & 0 & A \end{bmatrix}.$$

Therefore θ has a linear and differential branch number 5 if and only if all submatrices of A are nonsingular. Table 7.1 shows that for a random choice of coefficients the probability of not having a singular submatrix is $(1 - \frac{1}{256})^{17} \approx 0.93$. This means that most choices for a_i will be good. In a smart card implementation there is only a small amount of memory available and there is no place for large tables. This means that θ and γ cannot be combined into one set of tables, as in Section 8.1.2 and θ has to be implemented as a matrix multiplication. The following choice is probably optimal for a smart card implementation:

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \end{bmatrix} = \begin{bmatrix} \mathtt{2_x} & \mathtt{1_x} & \mathtt{1_x} & \mathtt{3_x} \end{bmatrix}.$$

Multiplication with $\mathtt{1_x}$ costs nothing, multiplication with $\mathtt{2_x}$ can be implemented with one shift and one exor for the reduction and, finally, multiplication with $\mathtt{3_x}$ can be implemented as the sum of the first two. Operation θ^{-1} corresponds to a multiplication with $B = A^{-1}$, also a circulant matrix, where

$$\begin{bmatrix} b_0 & b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} \mathtt{E_x} & \mathtt{9_x} & \mathtt{D_x} & \mathtt{B_x} \end{bmatrix}.$$

The Nonlinear Transformation γ

Transformation γ is a nonlinear byte substitution, identical for all bytes. It uses an invertible 8-bit S-box that is constructed by taking the mapping $s(x) = x^{-1}$ over $GF(2^8)$ and applying a bitwise affine transformation to the output bits (cf. Section 7.3.1). For this choice, $\delta = 2^{-6}$ and $\lambda = 2^{-3}$.

The Byte Permutation π

The effect of π is a transposition of the input: $\pi : X \mapsto X^t$. Clearly π is an involution.

The Key Addition σ

The key addition $\sigma[K^r]$ consists of the bitwise addition of a round key matrix K^r: $\sigma[K^r] : X \mapsto X \oplus K^r$, and is also an involution.

Properties

As in SHARK, the order of the σ and θ can be changed:

$$\begin{aligned} \sigma[K^r] \circ \theta(X) &= X \cdot A \oplus K^r \\ &= (X \oplus K^r \cdot A^{-1}) \cdot A \\ &= \theta \circ \sigma[\theta^{-1}(K^r)] \,. \end{aligned}$$

The same property holds for σ and π.

Since π only transposes the bytes $x_{i,j}$, and γ only operates on the individual bytes, independent of their position (i, j), the order of π and γ can also be changed: $\gamma^{-1} \circ \pi = \pi \circ \gamma^{-1}$.

The Round Key Evolution ψ

The round keys K^r are derived from the cipher key K in the following way. Key K^0 equals the cipher key K. The other round keys are derived iteratively by means of the invertible affine transformation ψ:

$$K^r = \psi(K^{r-1}) \,,$$

where ψ is defined in terms of the rows of the matrix that is formed by the key. The left byte-rotation operation $\mathsf{rotl}(k_i)$ is defined by

$$\mathsf{rotl}[k_{i,0}k_{i,1}k_{i,2}k_{i,3}] = [k_{i,1}k_{i,2}k_{i,3}k_{i,0}] \,,$$

and the right byte-rotation $\mathsf{rotr}(k_i)$ as its inverse. The key evolution is defined by:

$$\psi(K^{r-1}) = K^r \Leftrightarrow \begin{array}{lcl} k_0^r &=& k_0^{r-1} \oplus \mathsf{rotl}(k_3^{r-1}) \oplus C_{r-1} \\ k_1^r &=& k_1^{r-1} \oplus k_0^r \\ k_2^r &=& k_2^{r-1} \oplus k_1^r \\ k_3^r &=& k_3^{r-1} \oplus k_2^r \end{array} \,.$$

The round constants C_t are also defined iteratively: $C_0 = \mathtt{1_x}$ and $C_r = \mathtt{2_x} \cdot C_{r-1}$. This choice removes regularities in the round function.

The 128 bits of the round keys k^r can be divided over eight sets l_i of 16 bits: two bits are in the same set if their positions in the vector k are equal, modulo

eight. The input of each S-box of γ is influenced by one bit from each set of round key bits, and ψ acts separately on each set l_i:

$$\psi(l_i) = l_i \cdot \begin{bmatrix} I & I & I & I \\ 0 & I & I & I \\ 0 & 0 & I & I \\ J & J & J & J \oplus I \end{bmatrix} = l_i \cdot D\,,$$

where I is the 4×4 unity matrix, and

$$J = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The round keys can be described using eight instances of the linear code $\mathcal{C}_\psi$, that has the following generator matrix:

$$G_\psi = [I\ D\ D^2\ D^3\ \ldots\ D^8]\,.$$

Since the distance of this code is 32, the round keys that are derived from two different keys differ in at least 32 bits. Since D is invertible, these differences are spread over the nine round keys. It would be possible to have better diffusion, but only by making the key schedule slower to execute.

The Cipher SQUARE

The building blocks are combined into the round transformation denoted by $\rho[K^r]$:

$$\rho[K^r] = \sigma[K^r] \circ \pi \circ \gamma \circ \theta\,. \tag{8.12}$$

SQUARE is defined as eight rounds, preceded by a key addition $\sigma[K^0]$ and by θ^{-1}:

$$\text{SQUARE}[K] = \rho[K^8] \circ \rho[K^7] \circ \cdots \circ \rho[K^1] \circ \sigma[K^0] \circ \theta^{-1}\,. \tag{8.13}$$

For critical applications the number of rounds can be increased. This can be done in a straightforward way and requires no adaptation of the key schedule.

8.2.2 Implementation

In a similar way to SHARK, the different operations of the SQUARE round transformation can be efficiently combined in table lookups. While SHARK uses 64-bit words, SQUARE uses only 32-bit words. Furthermore, the operations of SQUARE have been chosen such that implementations on platforms with smaller word lengths can also be efficient.

8-bit Processor

On an 8-bit processor SQUARE can be programmed by simply implementing the different component transformations. This is straightforward for π and σ. The transformation γ requires a table of 256 bytes. Operation θ requires multiplication in the field $GF(2^8)$. The matrix A has been chosen to make this very efficient. The key evolution ψ has been chosen to make it easy to calculate the round keys during the encryption. J. Daemen has written an Assembler implementation for Motorola's M68HC05 microprocessor, typical for Smart Cards. The code occupies a total of 547 bytes of ROM, requires 36 bytes of RAM, and one encryption (including the key schedule) takes about 7500 cycles. This corresponds to less than 2 msec using a 4 MHz Clock.

Since θ^{-1} has a higher complexity than θ, an implementation of the inverse cipher uses more memory and the decryption is significantly slower than the encryption. In practical applications, often only the encryption operation is needed on a smart card.

32-bit Processor

Analogously to SHARK, the following succession of steps

$$\theta \circ \sigma[K^r] \circ \pi \circ \gamma = \sigma[K'^r] \circ \theta \circ \pi \circ \gamma \,,$$

with $K'^r = \theta(K^r)$, occur in SQUARE. The operations $\theta \circ \pi \circ \gamma$ can be combined, and $Y = \theta(\pi(\gamma(X)))$ can be expressed for each row y_i as

$$\begin{aligned} y_i &= (S(x_{0,i}), S(x_{1,i}), S(x_{2,i}), S(x_{3,i})) \cdot A \\ &= S(x_{0,i}) \cdot a_0 \oplus S(x_{1,i}) \cdot a_1 \oplus S(x_{2,i}) \cdot a_2 \oplus S(x_{3,i}) \cdot a_3 \,. \end{aligned} \tag{8.14}$$

The expanded substitution tables $T_i(x)$ are defined by

$$T_i(x) = S(x) \cdot a_i \,.$$

Since A is circulant, its rows a_i are rotated versions of row a_0. Thus, the entries of the tables T_i are also rotations of the entries of T_0. Equation (8.14) then becomes:

$$y_i = \bigoplus_j \mathsf{rotr}^j(T_0(x_{ji})) \,.$$

This means that $\theta \circ \pi \circ \gamma$ can be done using 16 table lookups, 12 rotations and 12 exors of 32-bit words. The table T_0 has 256 entries of 32 bits, or one kilobyte in total. Alternatively, the four tables T_i might be used, eliminating the requirement for rotations, but increasing the memory requirement for the tables to 4 kilobytes.

Analogously to SHARK, the application of θ^{-1} will lead to the canceling of θ in one round. Consider the first round, and the preceding σ and θ^{-1}:

$$\begin{aligned}\rho[k^1] \circ \sigma[k^0] \circ \theta^{-1} &= \sigma[k^1] \circ \pi \circ \gamma \circ \theta \circ \sigma[k^0] \circ \theta^{-1} \\ &= \sigma[k^1] \circ \pi \circ \gamma \circ \sigma[\theta(k^0)]\,.\end{aligned}$$

For the first round, $\pi \circ \gamma$ has to be implemented instead of $\theta \circ \pi \circ \gamma$. This would mean that the table S also has to be in memory. However, since $a_1 = \mathtt{1_x}$, the entries of the S-box $S[x]$ can be extracted from T_0, removing the extra storage requirement for S.

8.2.3 Inverse Cipher

It will be shown that the *structure* of the inverse cipher is equal to the structure of the cipher itself. The components and the round keys are different. Consider a one-round version of SQUARE. The decryption operation is given by

$$\theta \circ \sigma[K^0] \circ \rho[K^1]^{-1} = \theta \circ \sigma[K^0] \circ \theta^{-1} \circ \gamma^{-1} \circ \pi \circ \sigma[K^1]\,.$$

Using the transformation's properties, this can be rewritten as:

$$\begin{aligned}\theta \circ \sigma[K^0] \circ \rho[K^1]^{-1} &= \sigma[\theta(K^0)] \circ \pi \circ \gamma^{-1} \circ \sigma[K^1] \\ &= \sigma[\theta(K^0)] \circ \pi \circ \gamma^{-1} \circ \sigma[K^1] \circ \theta^{-1} \circ \theta \\ &= \sigma[\theta(K^0)] \circ \pi \circ \gamma^{-1} \circ \theta^{-1} \circ \sigma[\theta(K^1)] \circ \theta \\ &= \rho'[\theta(K^0)] \circ \sigma[\theta(K^1)] \circ \theta,\end{aligned}$$

where the new round transformation is defined as

$$\rho'[K^r] = \sigma[K^r] \circ \pi \circ \gamma^{-1} \circ \theta^{-1}\,.$$

This derivation can be generalized in a straightforward way to include more than one round. Hence the inverse cipher is equal to the cipher itself with γ replaced by γ^{-1}, θ by θ^{-1}, and with different round key values.

8.2.4 Cryptanalysis

Linear and Differential Cryptanalysis

SQUARE was designed according to the principles of the Wide Trail strategy (cf. Chapter 7). It is an example of a cipher with two-level diffusion. Consider four rounds of SQUARE, without key addition to simplify the discussion:

$$\begin{aligned}\rho \circ \rho \circ \rho \circ \rho &= \pi \circ \gamma \circ \theta \circ \pi \circ \gamma \circ \theta \circ \pi \circ \gamma \circ \theta \circ \pi \circ \gamma \circ \theta \\ &= \pi \circ \gamma \circ \theta \circ \gamma \circ \pi \circ \theta \circ \pi \circ \gamma \circ \theta \circ \gamma \circ \pi \circ \theta \\ &= \mathcal{A}_{\theta,\pi\circ\theta\circ\pi}\,.\end{aligned}$$

The results of Proposition 7.9 can be applied to determine the minimum number of active S-boxes in a differential characteristic or a linear approximation. It has already been shown that θ operates separately on each row of its input and has linear and differential branch number 5. It remains to determine what the branch numbers of $\theta_2 = \pi \circ \theta \circ \pi$ are with respect to the rows of the input. From the definitions of θ and π, it follows that $\theta_2 : X \mapsto A^t \cdot X$. Expressing this in terms of the rows yields:

$$y_i = a_{-i} \cdot x_0 \oplus a_{1-i} \cdot x_1 \oplus a_{2-i} \cdot x_2 \oplus a_{3-i} \cdot x_3 \,.$$

The associated code of θ_2, with the rows x_i, y_i as components, has generator matrix

$$G = [I \; A^t] \,.$$

The choice of A in SQUARE makes the code MDS, and $\mathcal{B}_d^*(\theta_2) = \mathcal{B}_l^*(\theta_2) = 5$. Therefore the minimum number of active S-boxes over four rounds is $5 \times 5 = 25$. Application of (7.2) and (7.3) results in an upper bound of $(2^{-6})^{25} = 2^{-150}$ for the probability of a four-round differential characteristic and $(2^{-3})^{25} = 2^{-75}$ for input-output correlation over four rounds. As mentioned before, these numbers give only an indication of the security level of the cipher.

Structure attack

The structure of SQUARE allows for an efficient dedicated attack, that is an extension of the structure attack on SHARK (cf. Section 8.1.4). The attack is faster than an exhaustive key search for versions of SQUARE with up to six rounds. After describing the basic attack on four rounds, the extension to five and six rounds will be discussed.

Four Rounds Recall the definition of a Λ-set from Section 8.1.4. The set of indices λ becomes a set of index pairs (i, j). Application of γ or $\sigma[k]$, still results in a Λ-set with the same λ. Application of π to a Λ-set with $\lambda = \{(i_1, j_1), \ldots (i_u, j_u)\}$, results in a Λ-set with $\lambda = \{(j_1, i_1), \ldots (j_u, i_u)\}$. Application of θ to a Λ-set with $\lambda = \{(i, j)\}$, results in a Λ-set with $\lambda = \{(i, 0), (i, 1), (i, 2), (i, 3)\}$. A second application of θ results in a λ-set containing all 16 index pairs.

The attack starts with a $\Lambda(\{(i, j)\})$-set. Since the θ of the first round is canceled, the first round does not influence the index set. In the second round the Λ-set is converted to a Λ-set with $\lambda = \{(i, 0), (i, 1), (i, 2), (i, 3)\}$. After the third round, there is still a Λ-set, with λ now containing all index pairs. After θ

of the fourth round, for each byte the sum over all values in the set equals zero:

$$\bigoplus_{l=0}^{255} y_{i,j} = \bigoplus_{l=0}^{255} \bigoplus_{v} a_{j-v} x_{i,v} = \bigoplus_{w} a_w \bigoplus_{l=0}^{255} x_{i,w+j} = \bigoplus_{w} a_w 0 = 0\,.$$

This balance is destroyed by the subsequent application of γ. The cryptanalyst guesses a value for one byte of the last round key. The last σ, π and γ operations are inverted for one ciphertext byte in all texts of the set and the balance of this sum is checked. If the sum is not balanced, the guess for the key byte was wrong. A second Λ-set may be necessary to catch all wrong guesses. The plaintexts can be reused to determine all bytes of the last round key.

Extension by a round at the end Since the single-round diffusion of SQUARE is rather limited, the above technique can still be applied when a round is added at the end. Any byte from the output of θ in the first round can be calculated from the ciphertext by guessing the corresponding byte of the fourth round key, and the four bytes of the fifth round key that are involved.

In this 5-round attack, 2^{40} key values must be checked, and this must be repeated four times in order to determine the fifth round key. Checking a single Λ-set leaves only 1/256 of the wrong key assumptions as possible candidates. Therefore only five sets are required to recover the key.

Extension by a round at the beginning The basic idea is to choose a set of plaintexts that result in a Λ-set at the output of the second round with only one index pair in λ. This requires the assumption of values of four bytes of the round key k^0.

The cryptanalyst starts with a pool of 2^{32} chosen plaintexts, that have a constant value for all bytes in three of the four 'columns'. The fourth column takes all possible values. The four key bytes of the first key addition are guessed and 256 plaintexts are selected that will produce the required Λ-set after the second round, assuming that the first round key guess is correct. The standard four-round attack is performed to recover the last round key. If the attack fails to suggest a single key value, the initial guess for the four bytes of the first round key must be wrong. The cryptanalyst then assumes another value for the first round key and selects another set of 256 texts from the pool.

Complexity of the attacks Combining both extensions results in a six round attack. Although infeasible with current technology, this attack is faster than exhaustive key search. Table 8.4 summarises the attacks.

There seems to be no obvious way in which the attack can be extended to seven rounds or more.

Attack	Plaintexts	Time	Memory
4-round	2^9	2^9	small
5-round type 1	2^{11}	2^{40}	small
5-round type 2	2^{32}	2^{41}	2^{32}
6-round	2^{32}	2^{73}	2^{32}

Table 8.4: Complexities of the attack on SQUARE.

8.2.5 Performance

The reference implementation is written in C and runs at 2.63 Mbyte/s on a 100 MHz Pentium with the Windows95 operating system. An assembler implementation of the algorithm [15] runs at 4.94 Mbyte/s on the same computer.

The M68HC05 Smart Card implementation fits in 547 bytes and takes less than 2 msec. (4 MHz Clock). The high degree of parallelism allows compact hardware implementations in Gbit/s range with current technology.

8.3 Extensions

The modular design of the ciphers makes it easy to extend both block ciphers to larger block lengths. This can be done by increasing the number of parallel S-boxes and/or by increasing the size of the S-boxes.

Both ciphers use a uniform round transformation. It is also possible to design a Feistel cipher using the same building blocks. This Feistel cipher would have twice the block length of the original designs, and the round transformation would be

$$\begin{aligned} s^{r+1} &= t^r \\ t^{r+1} &= s^r \oplus \rho(t^r, k^r)\,. \end{aligned}$$

An advantage of this approach is that the inverse of cipher does not require the inverse of ρ. This allows more freedom in the choice of S-boxes, because they no longer have to be invertible. Also, the diffusion layer and the key addition do not have to be interchangeable. This means that it becomes possible to use a nonlinear diffusion layer θ. In order to be able to apply ρ with a single set of table lookups (cf. Section 8.1.2), the mapping θ should still be *additively separable*.

Definition 8.2 *A mapping θ is additively separable if*

$$f(x, y) \equiv f(x, 0) \oplus f(0, y)\,.$$

Considering the expanded tables $T[x]$ as the 'real' S-boxes, such a cipher would be very similar to the CAST algorithms. Both ciphers use S-boxes with small input size and large output size. The important difference is that the round transformation of the presented designs has a guaranteed diffusion and nonlinearity. If ρ is chosen to be invertible, then the round function is balanced and is not vulnerable to the attack that is presented in Chapter 6.

8.4 Conclusions

Two new block ciphers have been presented. Both ciphers have a uniform round structure. SHARK is a 64-bit block cipher oriented towards 64-bit architectures. The key length can vary between 64 and 128 bits. Since it has optimal diffusion, the cipher is proposed with only six rounds. On a 266 MHz DEC Alpha, a C implementation of SHARK runs at 6.30 Mbyte/s. While the performance on PC's is not impressive, SHARK will be very fast on tomorrow's computers. The assembler implementation runs at 1.23 Mbyte/s on a 90 MHz Pentium, but it will run at 10 Mbyte/s on a 166 MHz Pentium II. There exists one attack on three rounds, requiring 2^9 chosen plaintexts, a memory of size 2^8 and an effort of about 2^{17} encryptions. A second attack breaks four rounds of a modified version of SHARK, but it is not applicable to the actual algorithm. The design of SHARK has been published in [111].

SQUARE is a 128-bit block cipher with a 128-bit key. It can be efficiently implemented on a wide range of processors, including smart cards. The cipher uses a two-level diffusion structure and is proposed with eight rounds. On a 100 MHz Pentium SQUARE runs at 4.94 Mbyte/s. The same attack that breaks three rounds of SHARK, breaks six rounds of SQUARE. It requires 2^{32} chosen plaintexts, a memory of size 2^{32} and an effort of about 2^{73} encryptions. The design of SQUARE has been published in [27, 28].

Chapter 9

Conclusions and Open Problems

In this thesis a number of block ciphers have been studied. Block ciphers are necessarily algorithms with a sophisticated structure. The complexity of the structure makes the design and the analysis very difficult. The first part of this thesis groups our contributions to the analysis of block ciphers. The second part groups our contributions to the design of new block ciphers.

9.1 Cryptanalysis of Block Ciphers

During the last decade a number of simple statistical techniques have been developed that can be used in the study of cryptographic algorithms. Differential cryptanalysis [10] studies the propagation of differences through the various stages of an algorithm and linear cryptanalysis [77] studies the correlation between linear combinations of input bits and output bits. Both techniques are described in Chapter 3 together with the most common variants and extensions. A theoretical derivation leads to the observation that differential attacks in principle can also be applied when the signal-to-noise ratio, as defined by E. Biham and A. Shamir, is smaller than one, contrary to what was believed earlier. This is illustrated in Chapter 5 with an attack on a reduced version of IDEA.

In order to make the analysis of block ciphers more tractable, it is often necessary to make a number of simplifying assumptions. The first basic assumption is that the different rounds of the cipher operate independently of one another and that they can be decoupled. Moreover, it is often assumed that the separate building blocks of the round transformation operate independently and can be studied separately. A third simplification is made by averaging the key depen-

dent behaviour of the transformations over all possible keys. In this way a kind of 'average behavior' for the block cipher is studied, that is independent of the key. These approximations are useful, but they also have limitations. Chapter 4 and 5 discuss our extensions of the known analysis techniques that have been obtained by removing some of the simplifications.

Chapter 4 discusses the introduction of probability theory techniques. The use of maximum likelihood estimators and Bayes' rule instead of the customary counters allows the extension of differential cryptanalysis to modes where the output of the block cipher is only partially visible, e.g., the m-bit CFB mode with small m. Other optimisations have been made in order to analyse block ciphers that are used as the compression function of a hash algorithm. The optimisations are based on the fact that in this situation the cryptanalyst has much more control over the inputs of the cipher than in the encryption modes.

Chapter 5 introduces techniques that exploit key dependent relations. Some algorithms (such as IDEA and MAA) rely to a great extent on nonlinear key addition in order to provide the security of the algorithm. In this case there are often 'weak keys', for which the nonlinear addition can be approximated very easily. For these weak keys, the algorithm becomes highly vulnerable to certain attacks. Differentials with a probability that is too low 'on average' turn out to be useful because for a large number of the keys their probability is significantly higher than the average. Another presented attack strategy combines attacks that each work for a class of 'weak keys'. By performing the attacks in parallel it is possible to recover any key belonging to any of the classes.

In Chapter 6 a new attack has been presented. While the attack was originally developed to break algorithms of the CAST family, it is also applicable to reduced versions of LOKI and Luby-Rackoff constructions.

The main conclusion that can be drawn is that the analysis of block ciphers is still in its infancy. There are a few basic analysis techniques that can be applied with some success to most designs, but these techniques have to be extended and adapted for almost every new design that is investigated. The complexity and the nature of existing block cipher analyses could result in an amount of frustration: on a first view it is not easy to distinguish a strong block cipher from a weak one. There are no known techniques for a designer to prove the security of his block cipher. On the other hand, demonstrating weaknesses in bad block ciphers designs may actually require more work than the design itself. Most block ciphers lie in the twilight zone between probably secure and provably not.

Regrettable as it is, this situation must not lead to a careless attitude where "since nothing is proven, everything is allowed." The cryptanalysis of MacGuffin shortly after its publication, the devastating attack on Akelarre and the fact that several important design flaws in early members of the CAST family of algorithms have been pointed out, show that there are situations where an

incautious design can be corrected. Also, the extensions and modifications of the basic techniques that are made during the analysis of practical designs, lead to a better understanding of the inherent trade-offs in block cipher design.

9.2 Design of Block Ciphers

The second part of this thesis elaborates on the Wide Trail design strategy of J. Daemen [26]. Chapter 7 extends the strategy and discusses construction methods for the building blocks of the round transformation of block ciphers. Here the most important contribution consists of the application of the theory of linear codes to construct mappings with good diffusion properties. Linear codes define associated mappings; the differential and linear branch number of the mappings are equal to the distance of the associated code and the distance of the dual of the associated code, respectively. Optimal diffusion layers can be built from Maximum Distance Separable Codes. It is shown how incomplete mappings can be used in the round transformation and still provide good diffusion by means of a multi-level diffusion construction. For the nonlinear layer, S-boxes can be derived from the inverse mapping in the finite field $GF(2^8)$ [93], or they can be randomly generated and tested. Coding theory can also be used to construct a key scheduling with provable properties. Alternatively, the key scheduling can be based on a pseudo-random noise generator that is seeded with the key.

Finally, in Chapter 8 two new block ciphers have been proposed that were designed using the wide trail strategy. SHARK is a 64-bit cipher with optimal diffusion, oriented towards 64-bit architectures. SQUARE is a 128-bit cipher that uses two-level diffusion and that can be implemented efficiently on a wide range of processors, from smart cards to 64-bit processors. Both ciphers use expanded tables that combine the diffusion layer with the nonlinear layer to reduce the extra work that has to be done for the more complex diffusion layer.

Probably the most important result from this part of the thesis is the observation that it is possible to construct mappings with good cryptographic properties by using results from other subfields of mathematics. An advantage is that this allows to construct block ciphers with some provable properties, although it is still not possible (yet) to prove the security of the resulting ciphers.

9.3 Open Problems

The following problems were encountered during the research that was performed for this thesis, and are still unsolved.

Truncated differentials are an interesting extension of differential cryptanalysis. Unlike for ordinary differentials, the probability of truncated differentials is to a large extent independent of the nonlinear elements of a round transformation. It would be interesting to develop a heuristic rule to determine the resistance of a given cipher against truncated differential attacks. Also it remains an open question whether there exists an equivalent concept for linear cryptanalysis.

Key dependent relations can be a very powerful tool for a cryptanalyst, but at the moment there exists no efficient way to find the best relations and the weakest keys. Key dependent relations could be very useful to analyse a cipher like Blowfish, with key dependent S-boxes.

The CAST block cipher family has given birth to new members, with an as yet unknown security level.

A problem with optimal diffusion mappings is that they require many operations to execute. Using the theory of linear codes it is probably possible to investigate the trade-off between fast diffusion and fast executing.

It remains an open question as to whether it is possible to find S-boxes with maximal distance to all linear functions, and with minimal differential uniformity, without at the same time introducing mathematical structure. And if it is theoretically possible, is it practically feasible ?

In Chapter 7 it was indicated how trapdoor block ciphers can be used to build a public key encryption scheme, but this matter requires further investigation before a practical design can be proposed.

The ciphers SHARK and SQUARE await further cryptanalysis.

Bibliography

I have put this book [126] next to Schneier's on my bookshelf. I recommend that if you are serious about cryptography, you should do the same.
Bob Bruen

[1] C.M. Adams and S.E. Tavares, "The structured design of cryptographically good S-boxes," *Journal of Cryptology*, Vol. 3, No. 1, 1990, pp. 27–42.

[2] C.M. Adams and S.E. Tavares, "Designing S-boxes for ciphers resistant to differential cryptanalysis," *Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography*, W. Wolfowicz, Ed., Fondazione Ugo Bordoni, 1993, pp. 181–190.

[3] C.M. Adams, "Simple and effective key scheduling for symmetric ciphers," *Proceedings of SAC'94, Workshop on Selected Areas in Cryptography*, pp. 129–133.

[4] C.M. Adams, "Constructing symmetric ciphers using the CAST design procedure," *Designs, Codes, and Cryptography*, to appear.

[5] G. Alvarez, D. de la Guiaía, F. Montoya and A. Peinado, "Akelarre: a new block cipher algorithm," *Proceedings of SAC'96, Third Annual Workshop on Selected Areas in Cryptography*, Queen's University, Kingston, Ontario, 1996.

[6] K. Aoki and K. Ohta, "Differential-linear cryptanalysis of FEAL-8," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences,* Vol. E79-A, No. 1, January 1996.

[7] K. Aoki, K. Kobayashi and S. Moriai, "Best differential characteristic search of FEAL," *Fast Software Encryption, LNCS 1267*, E. Biham, Ed., Springer-Verlag, 1997, pp. 41–53.

[8] K.G. Beauchamp, *Walsh Functions and Their Applications,* Academic Press, New York, 1975.

[9] A. Beguelin, J. J. Dongarra, R. Manchek, K. Moore, R. Wade, J. Plank and V. Sunderam, *HeNCE: a user's guide,* Version 1.2, December 1992.

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

[11] E. Biham, "New types of cryptanalytic attacks using related keys," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 398–409.

[12] M. Blaze and B. Schneier, "MacGuffin: an unbalanced Feistel network block cipher," *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 97–110.

[13] A. Blum, M. Furst, M. Kearns and R.J. Lipton, "Cryptographic primitives based on hard learning problems," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 278–291.

[14] J. Borst, L.R. Knudsen and V. Rijmen, "Two attacks on reduced IDEA," *Advances in Cryptology, Proceedings Eurocrypt'97, LNCS 1233*, W. Fumy, Ed., Springer-Verlag, 1997, pp. 1–13.

[15] A. Bosselaers, proprietary software, K.U.Leuven.

[16] B.O. Brachtl, D. Coppersmith, M.M. Hyden, S.M. Matyas, C.H. Meyer, J. Oseas, S. Pilpel and M. Schilling, *"Data authentication using modification detection codes based on a public one way encryption function,"* U.S. Patent Number 4,908,861, March 13, 1990.

[17] L. Brown, M. Kwan, J. Pieprzyk and J. Seberry, "Improving resistance against differential cryptanalysis and the redesign of LOKI," *Advances in Cryptology, Proceedings Asiacrypt'91, LNCS 739*, H. Imai, R.L. Rivest, and T. Matsumoto, Eds., Springer-Verlag, 1993, pp. 36–50.

[18] F. Chabaud and S. Vaudenay, "Links between differential and linear cryptanalysis," *Advances in Cryptology, Proceedings Eurocrypt'94, LNCS 950*, A. De Santis, Ed., Springer-Verlag, 1995, pp. 356–365.

[19] C. Charnes, L. O'Connor, J. Pieprzyk, R. Safavi-Naini and Y. Zheng, "Comments on Soviet encryption algorithm," *Advances in Cryptology, Proceedings Eurocrypt'94, LNCS 950*, A. De Santis, Ed., Springer-Verlag, 1995, pp. 433–438.

[20] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, 1987, pp. 1–6.

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

[22] T.W. Cusick and M.C. Wood, "The REDOC-II cryptosystem," *Advances in Cryptology, Proceedings Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 545–563.

[23] J. Daemen, R. Govaerts and J. Vandewalle, "Weak keys for IDEA," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 224–231.

[24] J. Daemen, R. Govaerts and J. Vandewalle, "A new approach towards block cipher design," *Fast Software Encryption, LNCS 809*, R. Anderson, Ed., Springer-Verlag, 1994, pp. 18–32.

[25] J. Daemen, R. Govaerts and J. Vandewalle, "Cryptanalysis of 2,5 rounds of IDEA," *Technical Report ESAT-COSIC Report 94-1,* Department of Electrical Engineering, Katholieke Universiteit Leuven, March 1994.

[26] J. Daemen, "Cipher and hash function design strategies based on linear and differential cryptanalysis," *Doctoral Dissertation*, March 1995, K.U.Leuven.

[27] J. Daemen, L.R. Knudsen and V. Rijmen, "The block cipher SQUARE," *Fast Software Encryption, LNCS 1267*, E. Biham, Ed., Springer-Verlag, 1997, pp. 149–165.

[28] J. Daemen, L.R. Knudsen and V. Rijmen, "The SQUARE Encryption algorithm," *Dr. Dobb's Journal*, to appear.

[29] D. Davies and S. Murphy, "Pairs and triplets of DES S-boxes," *Journal of Cryptology*, Vol. 8, No. 1, 1995, pp. 1–26.

[30] D. Davies and W. Price, *Security for Computer Networks*, 2nd ed., Wiley, 1989.

[31] D. Davies, "A message authenticator algorithm suitable for a mainframe computer," *Advances in Cryptology, Proceedings Crypto'84, LNCS 196*, G.R. Blakley and D. Chaum, Eds., Springer-Verlag, 1985, pp. 393–400.

[32] D. Davies, D.O. Clayden, "The message authenticator algorithm (MAA) and its implementation," *NPL Report DITC 109/88,* Feb. 1988.

[33] M.H. Dawson and S.E. Tavares, "An expanded set of S-box design criteria based on information theory," *Advances in Cryptology, Proceedings Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 352–367.

[34] W. Diffie and M. Hellman, "Exhaustive cryptanalysis of the NBS data encryption standard," *Computer*, pp. 74–78, 1977.

[35] J.F. Dillon, "Elementary Hadamard difference sets," *Proceedings of the Sixth Southeastern Conference on Combinatorics, Graph Theory and Computing, Boca Raton, Florida, Congressum Numerantium No. XIV, Utilitas Math., Winnipeg, Manitoba*, 1975, pp. 237–249.

[36] H. Feistel, "Cryptography and computer privacy," *Scientific American*, Vol. 228, No. 5, May 1973, pp. 15–23.

[37] H. Feistel, W.A. Notz and J.L. Smith, "Some cryptographic techniques for machine-to-machine data communications," *Proc. IEEE*, Vol. 63, No. 11, November 1975, pp. 1545–1554.

[38] W. Feller, *An Introduction to Probability Theory and Its Applications,* Vol. 1, Wiley & Sons, 1968.

[39] *Data Encryption Standard,* Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[40] FIPS 81, *DES modes of operation,* National Bureau of Standards, 1980.

[41] FIPS 180-1, *Secure hash standard,* NIST, US Department of Commerce, Washington D.C., April 1995.

[42] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, *PVM 3 User's guide and reference manual,* May 1993.

[43] H. Gilbert and G. Chassé, "A statistical attack of the FEAL-8 cryptosystem," *Advances in Cryptology, Proceedings Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 22–32.

[44] H. Gilbert and P. Chauvaud, "A chosen plaintext attack of the 16-round Khufu cryptosystem," *Advances in Cryptology, Proceedings Crypto'94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 359–368.

[45] C. Harpes, G.G. Kramer and J.L. Massey, "A Generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma," *Advances in Cryptology, Proceedings Eurocrypt'95, LNCS 921*, L.C. Guillou and J.-J. Quisquater, Eds., Springer-Verlag, 1995, pp. 24–38.

[46] C. Harpes, "Cryptanalysis of iterated block ciphers," *ETH Series in Information Processing,* Vol. 7, J.L. Massey, Ed., Hartung-Gorre Verlag Konstanz, 1996.

[47] 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,* Information Systems Lab., Dept. of Electrical Eng., Stanford Univ., 1976.

[48] H.M. Heys and S.E. Tavares, "On the security of the CAST encryption algorithm", *Canadian Conference on Electrical and Computer Engineering*, pp. 332–335, Sept. 1994, Halifax, Canada.

[49] ISO 8731, *Banking – approved algorithms for message authentication, Part 1, DEA, Part 2, Message Authentication Algorithm (MAA),* 1987.

[50] ISO 8732, *Banking – key managament (wholesale),* 1988.

[51] ISO/IEC 9797, *Information technology - Data cryptographic techniques - Data integrity mechanisms using a cryptographic check function employing a block cipher algorithm,* 1993.

[52] ISO/IEC 10116, *Information technology - Security techniques - Modes of operation of an n-bit block cipher algorithm,* 1991.

[53] T. Jakobsen, "Correlation attacks on block ciphers," *Master's thesis*, Technical University of Denmark, January 1996.

[54] T. Jakobsen and L.R. Knudsen, "The interpolation attack on block ciphers," *Fast Software Encryption, LNCS 1267*, E. Biham, Ed., Springer-Verlag, 1997, pp. 28–40.

[55] J. Kelsey, B. Schneier and D. Wagner, "Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES," *Advances in Cryptology, Proceedings Crypto'96, LNCS 1109*, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 237–252.

[56] K. Kim, T. Matsumoto, H. Imai, "A recursive construction method of S-boxes satisfying strict avalanche criterion," *Advances in Cryptology, Proceedings Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 564–575.

[57] L.R. Knudsen, "Iterative characteristics of DES and s^2–DES," *Advances in Cryptology, Proceedings Crypto'92, LNCS 740*, E.F. Brickell, Ed., Springer-Verlag, 1993, pp. 497–511.

[58] L.R. Knudsen, "New potentially 'weak' keys for DES and LOKI," *Advances in Cryptology, Proceedings Eurocrypt'94, LNCS 950*, A. De Santis, Ed., Springer-Verlag, 1995, pp. 419–424.

[59] L.R. Knudsen, "Block ciphers – analysis, design and applications," *PhD. Thesis, DAIMI PB 485*, Aarhus University, 1994.

[60] L.R. Knudsen, "Truncated and higher order differentials," *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 196–211.

[61] L.R. Knudsen, "A key-schedule weakness in SAFER-K64," *Advances in Cryptology, Proceedings Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 274–286.

[62] L.R. Knudsen and T.A. Berson, "Truncated differentials of SAFER," *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 15–26.

[63] L.R. Knudsen and B. Preneel, "Hash functions based on block ciphers and quaternary codes," *Advances in Cryptology, Proceedings Asiacrypt'96, LNCS 1163*, K. Kim and T. Matsumoto, Eds., Springer-Verlag, 1996, pp. 77–90.

[64] L.R. Knudsen and M.J.B. Robshaw, "Non-linear approximations in linear cryptanalysis," *Advances in Cryptology, Proceedings Eurocrypt'96, LNCS 1070*, U. Maurer, Ed., Springer-Verlag, 1996, pp. 224–236.

[65] L.R. Knudsen and W. Meier, "Improved differential attack on RC5," *Advances in Cryptology, Proceedings Crypto'96, LNCS 1109*, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 216–228.

[66] N. Koblitz, *A Course in Number Theory and Cryptography,* Springer-Verlag, New York, 1987.

[67] X. Lai, "On the Design and Security of Block Ciphers," *PhD thesis*, ETH, Zürich, Switzerland, 1992.

[68] X. Lai and J.L. Massey, "A proposal for a new block encryption standard," *Advances in Cryptology, Proceedings Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 389–404.

[69] X. Lai, J.L. Massey and S. Murphy, "Markov ciphers and differential cryptanalysis," *Advances in Cryptology, Proceedings Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 17–38.

[70] S.K. Langford and M.E. Hellman, "Differential-linear cryptanalysis," *Advances in Cryptology, Proceedings Crypto'94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 26–39.

[71] A.K. Lenstra, H.W. Lenstra, Jr. and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Annalen,* No. 261, pp. 513–534, 1982.

[72] R. Lidl and H. Niederreiter, *Finite Fields*, Encyclopdia of Mathematics and its Applications 20, Addison-Wesley, Reading, Massachusetts, 1983.

[73] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM J. Computing,* Vol. 17, No. 2, pp. 373–386.

[74] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes,*, North-Holland, Amsterdam, 1977.

[75] J.L. Massey, "SAFER K-64: a byte-oriented block-ciphering algorithm," *Fast Software Encryption, LNCS 809*, R. Anderson, Ed., Springer-Verlag, 1994, pp. 1–17.

[76] J.L. Massey, "SAFER K-64: One year later," *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 212–241.

[77] M. Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 386–397.

[78] M. Matsui, "Cryptanalysis of DES cipher (I)," December 1993, preprint.

[79] M. Matsui, "The first experimental cryptanalysis of the Data Encryption Standard," *Advances in Cryptology, Proceedings Crypto'94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 1–11.

[80] M. Matsui, "On correlation between the order of S-box and the strength of DES," *Advances in Cryptology, Proceedings Eurocrypt'94, LNCS 950*, A. De Santis, Ed., Springer-Verlag, 1995, pp. 366–375.

[81] M. Matsui, "New block encryption algorithm MISTY," *Fast Software Encryption, LNCS 1267*, E. Biham, Ed., Springer-Verlag, 1997, pp. 54–68.

[82] S.M. Matyas, C.H. Meyer and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," *IBM Techn. Disclosure Bull.,* Vol. 27, No. 10A, 1985, pp. 5658–5659.

[83] W. Meier, "On the security of the IDEA block cipher," *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 371–385.

[84] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, October 1996.

[85] R.C. Merkle, *Secrecy, authentication, and public key systems,* UMI Research Press, Ann Arbor, Michigan, 1979.

[86] R.C. Merkle, "A certified digital signature," *Advances in Cryptology, Proceedings Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 218–238.

[87] R.C. Merkle, "Fast software encryption functions," *Advances in Cryptology, Proceedings Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 476–501.

[88] S. Miyaguchi, "The FEAL cipher family," *Advances in Cryptology, Proceedings Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 627–638.

[89] S. Moriai, K. Aoki and K. Ohta, "The best linear expression search of FEAL," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences,* Vol. E79-A, No. 1, January 1996.

[90] S. Murphy, "The cryptanalysis of FEAL-4 with 20 chosen plaintexts," *Journal of Cryptology*, Vol. 2, No. 3, 1990, pp. 145–154.

[91] N.J. Nilssen, *Problem-solving methods in artificial intelligence,* McGraw-Hill, 1971.

[92] K. Nyberg, "Perfect nonlinear S-boxes," *Advances in Cryptology, Proceedings Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 378–386.

[93] K. Nyberg, "Differentially uniform mappings for cryptography," *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 55–64.

[94] K. Nyberg, "Linear approximations of block ciphers," *Advances in Cryptology, Proceedings Eurocrypt'94, LNCS 950*, A. De Santis, Ed., Springer-Verlag, 1995, pp. 439–444.

[95] K. Nyberg, "S-boxes and round functions with controlled linearity," *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 111–130.

[96] K. Nyberg, "Generalized Feistel networks," *Advances in Cryptology, Proceedings Asiacrypt'96, LNCS 1163*, K. Kim and T. Matsumoto, Eds., Springer-Verlag, 1996, pp. 91–104.

[97] K. Nyberg and L.R. Knudsen, "Provable security against a differential attack," *Journal of Cryptology*, Vol. 8, No. 1, 1995, pp. 27–38.

[98] L. O'Connor, "On the distribution of characteristics in bijective mappings," *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 360–370.

[99] K. Ohta and M. Matsui, "Differential attack on message authentication codes," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 200–211.

[100] K. Ohta and K. Aoki, "Linear cryptanalysis of the Fast Data Encipherment Algorithm," *Advances in Cryptology, Proceedings Crypto'94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 12–16..

[101] W.W. Peterson and E.J. Weldon, *Error-Correcting Codes,* The MIT Press, Cambridge, 1972.

[102] B. Preneel,"Analysis and design of cryptographic hash functions," *Doctoral Dissertation*, January 1993, K.U.Leuven.

[103] B. Preneel, R. Govaerts and J. Vandewalle, "Hash functions based on block ciphers: a synthetic approach," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 368–378.

[104] B. Preneel, "Differential cryptanalysis of hash functions based on block ciphers," *Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993, pp. 183-188.

[105] B. Preneel, M. Nuttin, V. Rijmen and J. Buelens, "Cryptanalysis of the CFB mode of the DES with a reduced number of rounds," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 212–223.

[106] B. Preneel, V. Rijmen and P.C. van Oorschot, "Security analysis of the Message Authenticator Algorithm," *European Transactions on Telecommunications, special focus on Security*, to appear.

[107] B. Preneel and P.C. van Oorschot, "MDx-MAC and building fast MACs from hash functions," *Advances in Cryptology, Proceedings Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 1–14.

[108] B. Preneel and P.C. van Oorschot, "On the security of two MAC algorithms," *Advances in Cryptology, Proceedings Eurocrypt'96, LNCS 1070*, U. Maurer, Ed., Springer-Verlag, 1996, pp. 19–32.

[109] V. Rijmen and B. Preneel, "Improved characteristics for differential cryptanalysis of hash functions based on block ciphers," *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 242–248.

[110] V. Rijmen and B. Preneel, "Cryptanalysis of MacGuffin," *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 353–358.

[111] V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, E. De Win, "The cipher SHARK," *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 99–111.

[112] V. Rijmen and B. Preneel, "On weaknesses of non-surjective round functions," *Proceedings of the Workshop on Selected Areas in Cryptography – SAC'95*, Ottawa, May 18–19, 1995, pp. 100–106.

[113] V. Rijmen, B. Preneel and E. De Win, "On weaknesses of non-surjective round functions," *Designs, Codes, and Cryptography*, to appear.

[114] V. Rijmen and B. Preneel, "A family of trapdoor ciphers," *Fast Software Encryption, LNCS 1267*, E. Biham, Ed., Springer-Verlag, 1997, pp. 139–148.

[115] R.L. Rivest, "The RC5 encryption algorithm," *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 86–96.

[116] O.S. Rothaus, "On bent functions," *Journal of Combinatorial Theory (A),* Vol. 20, 1976, pp. 300–305.

[117] S. Rudeanu, *Boolean Functions and Equations,* North-Holland Publishing Company, 1974.

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

[119] K. Sakurai and S. Furuya, "Improving linear cryptanalysis of LOKI91 by probabilistic counting method," *Fast Software Encryption, LNCS 1267*, E. Biham, Ed., Springer-Verlag, 1997, pp. 114–133.

[120] T. Sauer and Y. Xu, "On multivariate Lagrange interpolation," Math. Comp. 64, 1995, pp. 1147–1170.

[121] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," *Fast Software Encryption, LNCS 809*, R. Anderson, Ed., Springer-Verlag, 1994, pp. 191–204.

[122] C.E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, No. 30, 1949, pp. 50–64.

[123] A. Shimizu and S. Miyaguchi, "Fast data encipherment algorithm FEAL," *Advances in Cryptology, Proceedings Eurocrypt'87, LNCS 304*, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 267–278.

[124] M.E. Smid and D.K. Branstad, "The Data Encryption Standard. Past and future," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 43–64.

[125] O. Staffelbach and W. Meier, "Cryptographic significance of the carry for ciphers based on integer addition," *Advances in Cryptology, Proceedings Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 601–615.

[126] D.R. Stinson, *Cryptography, theory and practice*, CRC Press, 1995.

[127] T. Tokita, T. Sorimachi and M. Matsui, "Linear cryptanalysis of LOKI and s^2DES," *Advances in Cryptology, Proceedings Asiacrypt'94, LNCS 917*, J. Pieprzyk and R. Safavi-Naini, Eds., Springer-Verlag, 1995, pp. 293–303.

[128] P.C. van Oorschot and M. Wiener, "A known-plaintext attack on two-key triple encryption," *Advances in Cryptology, Proceedings Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 318–325.

[129] B. Van Rompay and J. Verelst, "Analysis of Blowfish," *Master's Thesis* (in Dutch), Katholieke Universiteit Leuven, Dept. ESAT, 1996.

[130] S.A. Vanstone and P.C. van Oorschot, *An Introduction to Error Correcting Codes with Applications,* Kluwer, Boston, 1989.

[131] S. Vaudenay, "On the weak keys of Blowfish," *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 27–32.

[132] A.F. Webster and S.E. Tavares, "On the design of S-boxes," *Advances in Cryptology, Proceedings Crypto'85, LNCS 218*, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 523–534.

[133] M. Wiener, "Efficient DES key search," presentation at Rump Session of Crypto (August, 1993), Santa Barbara, CA. Available as TR-244, School of Computer Science, Carleton University, Ottawa, Canada, May 1994.

Appendix A

Block Cipher Survey

In recent years there has been a considerable number of new block cipher proposals. This appendix attempts to provide a survey of the most important proposals, and the best published attacks on them. (Speaking about Sisyphean tasks, ...)

The following abbreviations are used:

l block length

k key length

R number of rounds

F or U the type of the block cipher: F stands for Feistel network, U stands for uniform transformation

For theoretical attacks the work effort to obtain the required plaintext-ciphertext pairs is not counted.

The block ciphers are listed in alphabetical order.

Blowfish : $l = 64$, $k \leq 448$, $R = 16$, F [121].
Blowfish uses key dependent S-boxes. If the S-boxes are known, there is a differential attack [131] that works for a fraction 2^{-17} of the keys. It requires 2^{51} chosen plaintexts, a memory of 2^{32} and an effort of about 2^{57} encryptions. The attack breaks eight rounds for any key with 2^{48} chosen plaintexts, a memory of size 2^{32} and an effort of about 2^{45} encryptions. Section 3.6 describes an attack on four rounds that recovers the full key.

CAST : $l = 64$, $k = 64$ to 128, $R = 8$ to 16, F [1, 2, 3, 4].
CAST is actually a design procedure. Ciphers designed according to this

procedure inherit its name. Chapter 6 describes an attack on eight rounds [112, 113]. The attack requires 2^{62} chosen plaintexts. For CAST [2, 3] with 16-bit round keys the attack requires a memory of size 2^{16} and an effort of about 2^{75} encryptions. For CAST with 32-bit round keys or 37-bit round keys [4] this becomes a memory of size 2^{32} or size 2^{37} and an effort of about 2^{91} or 2^{96} encryptions respectively. For versions reduced to six rounds, the attack becomes very practical.

DES : $l = 64$, $k = 56$, $R = 16$, F [39].
The best theoretical attack known is the linear attack [77], requiring 2^{43} known plaintexts, a memory of size 2^{13} and an effort of about 2^{19} encryptions. Due to the short key, an exhaustive key search is feasible [133].

FEAL : $l = 64$, $k = 64$, $R = 8, 16, 24$ or 32, F [123].
The best attack known on eight rounds is a differential-linear attack [6]. It requires 12 (twelve) chosen plaintexts, 2^{18} bytes of memory and has a workload of 35 days computer time. The versions with 16, 24 or 32 rounds are vulnerable to a differential attack that requires $2^{30}, 2^{46}$ or 2^{67} chosen plaintexts and has a workload of $2^{30}, 2^{46}$ or 2^{67} encryptions [10]. The 16-round and 24-round versions are also vulnerable to a linear attack, requiring 2^{19} or 2^{42} known plaintexts [89]. The differential attacks also apply to FEAL-X and FEAL-NX [10].

GOST : $l = 64$, $k = 256$, $R = 32$, F [19]. The S-boxes of GOST are not specified.
The only published attack is a related-key chosen plaintext attack [55]. The probability of the differential characteristic depends on the S-boxes. Over a large set of randomly selected S-boxes this probability varies between 2^{-43} and 2^{-80} for a characteristic that may be used in an attack on twelve rounds.

IDEA : $l = 64$, $k = 128$, $R = 8(8.5)$, F (generalised). The output transformation is sometimes counted as an extra half-round [69].
The best attack known is a truncated differential attack on three rounds including the output transformation [14] (cf. Section 5.2). About 1% of the keys can be recovered using 2^{40} chosen plaintexts, a memory of 2^{48} and an effort of about 2^{51} encryptions. To find 31% of the keys, 2^{48} chosen plaintexts are required, the same amount of memory and an effort of 2^{59} encryptions.

Khafre : $l = 64$, $k = 64$ or 128, $R = 16, 24, 32, \ldots$, F [87].
The best attack known is a chosen plaintext attack on 24 rounds [10]. It requires 2^{53} chosen plaintexts, a memory of 2^8 and one hour of computer time.

Khufu : $l = 64$, $k \leq 512$, $R = 16, 24, 32, \ldots$, F [87].
The best attack is a chosen plaintext attack on 16 rounds [44]. It requires 2^{43} chosen plaintexts, a memory of 2^{25} and an effort of about 2^{43} operations.

LOKI91 : $l = 64$, $k = 64$, $R = 16$, F [17].
There is a chosen plaintext attack on LOKI91 that breaks 13 rounds [59]. It requires 2^{58} chosen plaintexts. The memory requirements are negligible and the effort can be estimated at about 2^{48} encryptions. By using 2^{33} chosen plaintexts, it is possible to speed up an exhaustive key search by a factor of four: only 2^{61} encryptions need be done on average [59]. The best known plaintext attack breaks 12 rounds [119]. It requires 2^{63} known plaintexts, a memory of 2^{21} and an effort of about 2^{63} encryptions.

MISTY : $l = 64$, $k = 128$, $R = 8, 12$, F (generalized)[81]. There are actually two MISTY algorithms: MISTY1 is an eight-round Feistel cipher with a new kind of key addition, MISTY2 has a more generalized Feistel structure and has 12 rounds.
To date no attack has been published.

RC5 : l, k and R are variable. The "nominal" values for the parameters are $l = 64$, $k = 128$, $R = 12$, U [115]. Every round of RC5 is composed of two Feistel rounds.
The best attack on this version is a chosen plaintext attack [65]. It requires 2^{54} chosen plaintexts and a small amount of memory and work. For some weak keys, these numbers are lowered. If l is increased to 128, then an attack on 24 rounds would require 2^{123} chosen plaintexts.

REDOC-II : $l = 70$, $k = 70$, $R = 10$, U [22]
The best attack is a chosen plaintext attack on four rounds [10]. It requires 2^{66} chosen plaintexts and a memory of size 2^{15}.

SAFER : $l = 64$, $k = 64$ or 128, $R = 6$, 8 or 10, U [75, 76]. Currently there are four versions of SAFER published: SAFER K-64 has $k = 64$ and $R = 6$, SAFER SK-64 has $k = 64$ and $R = 8$, SAFER K-128 and SAFER SK128 have $k = 128$ and $R = 10$. The key scheduling of the K versions has a weakness [61].
The best attack is a truncated differential attack on five rounds of SAFER K-64 [62]. It requires 2^{45} chosen plaintexts, a memory of 2^{32} and has a workload of 2^{46} encryptions. The attack also recovers 32 key bits of SAFER K-128, reduced to five rounds.

Shark : $l = 64$, $k = 64$ to 128, $R = 6$, U [111] (cf. Section 8.1).
The best attack is a structure attack on three rounds (cf. Section 8.1.4).

It requires 2^9 chosen plaintexts, a memory of size 2^8 and has a workload of 2^{17} encryptions.

Square : $l = 128$, $k = 128$, $R = 8$, U [27, 28] (cf. Section 8.2).
The best attack is a structure attack on six rounds (cf. Section 8.2.4, [27]). It requires 2^{32} chosen plaintexts, a memory of size 2^{32} and has a workload of 2^{73} encryptions.

3DES : $l = 64$, $k = 112$ or 168, $R = 48$, F. 3DES consists of three DES encryptions in cascade. The three DES operations can use three different keys, or the first and the last key can be equal [50].
The best attack on three-key 3DES is the meet-in-the-middle attack: it uses two known plaintexts, a memory of 2^{56} and has a workload of 2^{112} encryptions. The best chosen plaintext attack on two-key 3DES requires 2^{56} chosen plaintexts, a memory of size 2^{56} and has a workload of 2^{56} encryptions [85]. The best known plaintext attack involves a trade-off [128]. When given 2^n known plaintexts, it requires a memory of 2^{56} and a work effort of about 2^{120-n} encryptions.

3-Way : $l = 96$, $k = 96$, $R = 11$, U [24].
To date no attack has been published.

Nederlandse Samenvatting

Dit proefschrift behandelt de analyse en het ontwerp van een bepaalde klasse van cryptografische algoritmen: iteratieve blokcijfers. De blokcijfers worden geanalyseerd in verschillende modes: de standaard ECB mode waarin elk blok apart vercijferd wordt, de 'Cipher Feed Back' mode [40] en een mode waarin ze gebruikt worden als hutsfunctie.

1 Inleidende Begrippen

In de huidige maatschappij wint *cryptologie* elke dag aan belang. Waar het vercijferen en ontcijferen van boodschappen vroeger enkel een militaire aangelegenheid was, is in 1977 met de publicatie van de Amerikaanse encryptie-standaard, de DES [39], het startschot gegeven voor uitgebreid academisch onderzoek op dit terrein. Vandaag de dag verdienen mensen hun brood met het verkopen van cryptografische producten voor niet-militaire toepassingen.

Het belang van cryptologie volgt uit het feit dat deze wetenschap bestudeert hoe *informatie* beschermd kan worden. Informatie is hèt handelsproduct van deze tijd. Moderne digitale technieken maken het mogelijk om informatie snel en goedkoop op te slaan en uit te wisselen, zodat beslissingen steeds kunnen gebaseerd worden op recente informatie. De keerzijde van de medaille is dat informatie snel veroudert en dat het overzichtelijk catalogeren van de bergen informatie een ware Sisyphus-arbeid is. Hoewel informatie in ruwe vorm zeer goedkoop is, kan verwerkte informatie dus waardevol zijn, en moet zij beschermd worden.

Informatie kan beschermd worden tijdens het transport of wanneer ze opgeslagen wordt, tegen ongeoorloofde modificatie en tegen 'diefstal' (copiëren). Traditioneel wordt bescherming geboden door *encryptie*: de informatie wordt gecodeerd met een algoritme dat afhankelijk is van een kleine hoeveelheid geheime informatie, de *sleutel*, op zo'n manier dat het onmogelijk is om de informatie weer te decoderen als de sleutel niet gekend is.

De ontwikkeling van nieuwe technieken zoals hutsfuncties en publieke-sleu-

telalgoritmes heeft geleid tot de ontwikkeling van nieuwe toepassingen zoals digitale handtekeningen en elektronisch geld.

1.1 Cryptografie en Cryptanalyse

De veiligheid van een algoritme kan gedefinieerd worden op drie verschillende manieren [102]. De eerste definitie komt uit de informatietheorie: een algoritme is *onvoorwaardelijk veilig* als het niet kan gebroken worden door een tegenstander met onbeperkte rekenkracht. De tweede definitie komt uit de complexiteitstheorie: een algoritme is veilig als men kan bewijzen dat het aantal bewerkingen dat nodig is om het te breken exponentieel toeneemt als functie van het aantal bewerkingen dat men nodig heeft om te coderen. De studie van de aanvallen op cryptografische algoritmes wordt *cryptanalyse* genoemd; in de *cryptografie* bestudeert men het ontwerp van nieuwe algoritmes en toepassingen.

De derde aanpak is gebaseerd op de praktijk. De veiligheid van een algoritme wordt bepaald door zo nauwkeurig mogelijk te schatten hoeveel tijd en rekenkracht een aanvaller nodig zal hebben om het algoritme te breken. De schatting wordt gebaseerd op de resultaten van gekende analysetechnieken en een onderzoek door ervaren cryptanalysten. Deze aanpak heeft het belangrijke voordeel dat er op relatief eenvoudige wijze praktisch bruikbare algoritmes mee ontworpen kunnen worden. De betrouwbaarheid van de aanpak is echter sterk afhankelijk van de kwaliteit van de cryptanalyse die gedaan wordt.

1.2 Iteratieve Blokcijfers

In [122] beschrijft C.E. Shannon als eerste encryptie-algoritmes die opgebouwd worden door verschillende transformaties te concateneren: de zogenaamde *product-algoritmes*. Feistel is de eerste die een algoritme beschrijft dat gevormd wordt door een herhaalde toepassing van *dezelfde* transformatie [37].

Definitie A.1 *Een* iteratief blokcijfer *is een algoritme dat als invoer een klaartekstblok met een lengte van l bits heeft en als uitvoer een cijfertekstblok met een lengte l'. De transformatie van klaartekst naar cijfertekst gebeurt onder invloed van een sleutel en door middel van een herhaalde toepassing van een inverteerbare transformatie ρ, de* rondetransformatie.

De sterkte van een blokcijfer wordt vooral bepaald door de eigenschappen van de rondetransformatie. De twee meest gebruikte types rondetransformaties zijn het Feistel Netwerk [37] en de Uniforme transformatie [36].

Bij een Feistel netwerk wordt de invoer van ρ gesplitst in twee helften. Eén helft wordt gekopieerd naar de uitvoer en gebruikt als invoer voor een niet-lineaire functie, de zogenaamde F-functie, bij de andere helft wordt de uitvoer van de F-functie opgeteld. Het voordeel van deze constructie is dat ze altijd

inverteerbaar is, onafhankelijk van de keuze voor de F-functie. De rondetransformatie van de DES heeft deze structuur.

De Uniforme transformatie wordt soms ook een Substitutie-permutatie-netwerk genoemd. Bij de uniforme transformatie wordt de volledige invoer eerst door een niet-lineaire substitutietabel geleid en dan vervolgens gewijzigd door een transformatie met goede diffusie-eigenschappen. De rondetransformaties van SAFER [75], SHARK [111] en SQUARE [27] hebben deze structuur.

Blokcijfers kunnen gebruikt worden in verschillende modes [40] en toepassingen. In deze thesis worden blokcijfers geanalyseerd in de basismode ('Electronic Code Book') en de 'Ciphertext Feed Back' mode. Daarnaast wordt ook het gebruik van een blokcijfer in een hutsmode bestudeerd. Er wordt ook een manier voorgesteld om blokcijfers te gebruiken in een asymmetrisch encryptieschema.

1.3 Aanvallen

Bij de analyse van een cryptografisch algoritme wordt er altijd van uitgegaan dat aanvallers beschikken over een volledige beschrijving van het algoritme en alle details, behalve de gebruikte sleutel. Wanneer een cryptanalyst uit onderschepte cijferteksten en statistische informatie over de klaarteksten de sleutel kan terugvinden of de klaarteksten, dan spreekt men van een enkel-cijfertekst aanval. De meeste blokcijfers zijn bestand tegen een enkel-cijfertekst aanval.

Wanneer men veronderstelt dat de aanvaller bij enkele cijferteksten de bijhorende klaartekst kent, dan spreekt men van een gekende-klaartekst aanval. Bij een gekozen-klaartekst aanval mag de aanvaller de cijferteksten opvragen van klaarteksten naar zijn keuze. In een verwante-sleutel aanval beschikt de aanvaller over de cijferteksten die bekomen werden door dezelfde set klaarteksten te vercijferen onder verschillende sleutels. Hoewel deze aanvallen steeds minder en minder uitvoerbaar worden in de praktijk, worden ze toch bestudeerd omdat men op die manier een soort veiligheidsmarge kan inbouwen: algoritmes die bestand zijn tegen een gekozen-klaartekst aanval geven vermoedelijk een betere bescherming dan andere algoritmes, zelfs als de enige praktisch uitvoerbare aanval een enkel-cijfertekst aanval is.

2 Cryptografische Basistechnieken

Wanneer de aanvallen op blokcijfers ingedeeld worden volgens de wiskundige technieken die zij gebruiken, kunnen er twee grote klassen onderscheiden worden. Beide klassen van aanvallen kunnen bestudeerd worden gebruik makend van wiskundige technieken die ontwikkeld werden voor de studie van Booleaanse functies: Walsh transformatie, Hamming afstand, correlaties, ...

2.1 Differentiële Cryptanalyse

De eerste klasse valt onder de noemer *differentiële cryptanalyse* [10]. Bij deze aanvallen wordt de encryptie bestudeerd van een set klaarteksten die onderling een welbepaald verschil vertonen. Zelfs indien de sleutel niet gekend is, is het mogelijk om vertrekkende van twee klaarteksten die een bepaald verschil vertonen (een *'paar'*), de verschillen van de tussenresultaten te voorspellen met een bepaalde kans. Een *differentiële* karakteristiek is gedefinieerd als het tupel dat gevormd wordt door de voorspelde verschillen in de tussenresultaten. De kans van een differentiële karakteristiek is de kans dat de verschillen in de tussenresultaten van een willekeurig paar encrypties correct voorspeld worden.

De voorspelde verschillen in de tussenresultaten en de geobserveerde waarden van de klaarteksten en cijferteksten suggereren een aantal mogelijke waarden voor de sleutel. Als de kans van de karakteristiek kleiner is dan 1, is men niet zeker dat de voorspelde tussenresultaten correct zijn en moet de aanval herhaald worden voor een aantal paren. Als de kans van de karakteristiek voldoende hoog is, en er per paar niet teveel foute waarden voor de sleutel gesuggereerd worden, dan zal na verwerking van een voldoende groot aantal paren de vaakst gesuggereerde waarde voor de sleutel de correcte waarde zijn.

De differentiële aanval kan op verschillende manieren geoptimiseerd worden en aangepast aan de specifieke structuur van het geanalyseerde algoritme. Een eerste observatie is dat tussenresultaten eigenlijk alleen correct voorspeld moeten worden als ze ook in de aanval gebruikt worden. Bepalend voor het succes van een aanval is dan niet de kans van één differentiële karakteristiek, maar wel de som van de kansen van alle mogelijke karakteristieken die dezelfde waarden voorspellen voor de tussenresultaten die gebruikt worden in de aanval. Een *differentiële bundel* [69] wordt gedefinieerd als het tupel dat gevormd wordt door het gedeelte van de (voorspelde) tussenresultaten dat gebruikt wordt, en kan gezien worden als een verzameling differentiële karakteristieken die verschillen in de waarden van de niet voorspelde tussenresultaten. De kans van een differentiële bundel is gelijk aan de som van alle karakteristieken die er deel van uitmaken.

Bij *afgeknotte differentiële bundels* [60] wordt voor voorspelde tussenresultaten nog meer vrijheid gelaten; er wordt bijvoorbeeld enkel voorspeld of bepaalde bytes verschillen van nul of niet. Deze techniek is vaak bruikbaar om algoritmes te analyseren die werken op bytes in plaats van op individuele bits, bijvoorbeeld SAFER [75]. Bij *hogere orde differenties* worden in plaats van verschillen van twee waarden verschillen van verschillen gebruikt.

2.2 Lineaire Cryptanalyse

Lineaire aanvallen [77] exploiteren *correlaties* tussen lineaire combinaties van uitgangsbits van de rondetransformatie en lineaire functies van hun ingangsbits. De verwachte waarde van de correlatie tussen bits over verschillende ronden heen wordt benaderd door de verwachte waarden voor de correlaties over de afzonderlijke ronden te vermenigvuldigen. Deze benadering komt overeen met het gebruik van een differentiële karakteristiek in differentiële cryptanalyse; en er bestaat ook hier een uitbreiding die vergeleken kan worden met een differentiële bundel: de *lineaire omhulling*, die rechtstreeks de correlatie over de verschillende ronden heen gebruikt.

In een *differentieel-lineaire* aanval [70] wordt een differentiële karakteristiek gebruikt om een lineaire aanval te optimiseren.

2.3 Vereenvoudigingen

Om de toepasbaarheid en de complexiteit van differentiële aanvallen te schatten worden vaak een aantal vereenvoudigende veronderstellingen gemaakt: de kans van een differentiële bundel wordt benaderd door de kans van een karakteristiek, er wordt aangenomen dat de ronden onafhankelijk opereren en er wordt een soort gemiddelde van de kansen over alle mogelijke waarden van de sleutels gemaakt. Dezelfde benaderingen worden doorgevoerd voor lineaire aanvallen. Deze vereenvoudigingen laten vaak toe om op efficiënte wijze aanvallen te bedenken. Dit werd geïllustreerd op twee algoritmes: MacGuffin [12] en een gereduceerde versie van Blowfish [121]. MacGuffin is niet beter bestand tegen differentiële cryptanalyse dan de DES, en is ook gebroken met een lineaire aanval. Blowfish gereduceerd tot vier ronden werd gebroken met een tweede orde differentiële aanval.

3 Verbeterde Differentiële Cryptanalyse

Door gebruik te maken van een aantal technieken uit de theorie van de kansberekening is het mogelijk om de differentiële aanval uit te breiden en toe te passen op blokcijfers in de m-bit CFB mode, ook voor kleine waarden van m. Daarnaast wordt er in dit hoofdstuk een verbeterde aanval gepresenteerd voor hutsfuncties die gebaseerd zijn op een blokcijfer.

3.1 Cryptanalyse van de DES in de CFB Mode

Wanneer een blokcijfer in de m-bit CFB mode gebruikt wordt, met m klein, dan werkt de gewone differentiële aanval niet meer. m bepaalt hoeveel bits van de output van het blokcijfer zichtbaar zijn in de output; in de gewone differentiële

aanval vormt dit aantal een strikte bovengrens voor het aantal sleutelbits dat kan bepaald worden. Als m klein is, kan er maar een klein aantal klassen van 'equivalente' sleutels onderscheiden worden (2^m) en de meeste paren zullen dan alle waarden voor de sleutel suggereren. De differentiële aanval wordt nu uitgebreid door het eenvoudige onderscheid 'gesuggereerde sleutel — niet gesuggereerde sleutel' te vervangen door een waarschijnlijkheidsverdeling: aan elke waarde voor de sleutel wordt een a posteriori waarschijnlijkheid toegekend die met behulp van de regel van Bayes berekend wordt uit de a priori waarschijnlijkheid en de waarden van de klaarteksten en cijferteksten voor het verwerkte paar. Deze techniek maakt beter gebruik van de beschikbare informatie dan de gewone aanval. Met 2^{39} gekozen klaarteksten kunnen drie sleutelbits bepaald worden voor de DES in 8-bit CFB mode, gereduceerd tot 8 ronden. Voor grotere waarden van m kunnen meer sleutelbits teruggevonden worden en kan het aantal nodige klaarteksten gereduceerd worden.

3.2 Maximum Likelihood

Door het gebruik van maximum likelihood schatters kunnen zowel de lineaire als de differentiële aanval verbeterd worden. De techniek gebruikt een probabilistische relatie tussen klaartekst en cijfertekst, waarvan de kans op eenvoudige wijze afhangt van een deel van de sleutel. De cryptanalyst verzamelt klaarteksten en cijferteksten en kan zo de kans van de probabilistische relatie schatten. De maximum likelihood schatter zal dan de meest waarschijnlijke waarde voor de sleutel opleveren. Om de differentiële aanval op een blokcijfer in de CFB-mode te verbeteren, wordt de differentiële karakteristiek als relatie gebruikt. Op die manier kunnen voor de DES in 8-bit CFB-mode, gereduceerd tot 8 ronden, 10 sleutelbits teruggevonden worden in plaats van slechts drie.

De techniek werd ook toegepast op de lineaire aanval op de DES, maar experimenten toonden aan dat hierdoor geen verbetering optreedt. De techniek is wel nuttig bij blokcijfers waar de gewone lineaire aanval geen onderscheid kan maken tussen een hele klasse van sleutels, zoals bijvoorbeeld het blokcijfer Akelarre [5].

3.3 Differentiële Cryptanalyse van Hutsfuncties

Hutsfuncties die gebaseerd zijn op blokcijfers kunnen onderworpen worden aan een differentiële aanval die veel gelijkenis vertoont met de differentiële aanval op het onderliggende blokcijfer. Er zijn echter een paar belangrijke verschillen die maken dat een gewone differentiële aanval sub-optimaal presteert. Er werd een nieuwe aanval ontwikkeld en karakteristieken gezocht die optimaal afgestemd zijn op de nieuwe aanval. Met deze aanval kan een botsing gevonden worden voor een hutsfunctie die gebaseerd is op de DES, gereduceerd tot 12 ronden,

met een werkfactor van $2^{21.4}$ encrypties. De klassieke aanval, die gebaseerd is op de verjaardagsparadox, heeft een werkfactor van $2^{24.6}$ encrypties.

4 Sleutelafhankelijke Analyse

Zoals reeds aangehaald gebruikt men voor de schatting van de performantie en complexiteit van een aanval op een cryptografisch algoritme vaak een soort gemiddelde waarde over alle mogelijke sleutels. Hierbij wordt er vanuit gegaan dat de verschillende ronden onafhankelijk van elkaar opereren. Deze benaderingen geven niet altijd correcte resultaten. Er kan bijvoorbeeld een klasse van sleutels bestaan waarvoor de kans van sommige differentiële karakteristieken en/of sommige bitcorrelaties een veel hogere waarde hebben dan de gemiddelde waarde. Deze sleutels worden dan 'zwakke sleutels' genoemd. Er wordt gedemonstreerd dat het authenticatie-algoritme MAA [32] verschillende klassen van zwakke sleutels heeft. Een boodschap die geauthenticeerd werd met behulp van een zwakke sleutel is relatief gemakkelijk te vervalsen en het is ook gemakkelijker om zo'n zwakke sleutel terug te vinden.

Een andere manier om blokcijfers aan te vallen waarbij de kans van differentiële karakteristieken sterk afhankelijk is van de sleutel, is om verschillende karakteristieken in parallel te gebruiken. De performantie van de aanval wordt dan bepaald door het maximum van de kansen van de gebruikte karakteristieken. Dezelfde techniek kan gebruikt worden in een lineaire aanval en wordt geïllustreerd met een aanval op een gereduceerde versie van IDEA [69]. Met deze aanval is het mogelijk om 3 ronden van IDEA te breken.

Er wordt ook aan aanval gepresenteerd die werkt als de kans van een afgeknotte differentiële bundel groter *of kleiner* is dan de gemiddelde waarde. Deze aanval geeft een nieuw inzicht in de werking van differentiële cryptanalyse; totnogtoe werd steeds aangenomen dat alleen differentiële karakteristieken of bundels met een grote kans gebruikt konden worden. De nieuwe techniek wordt hier gebruikt om 3.5 ronden van IDEA te breken.

5 Niet-Surjektieve Aanval

Een belangrijk ontwerpcriterium voor nieuwe blokcijfers is vaak de snelheid die gehaald kan worden op een moderne processor. Daarom worden meestal basisbewerkingen gebruikt die snel uitgevoerd kunnen worden door een processor en waarvan men hoopt dat ze bijdragen tot de veiligheid van het algoritme. Een voorbeeld hiervan is het gebruik van sterk niet-lineaire substitutietabellen. Dit laat toe om het aantal ronden van een algoritme laag te houden en toch sterke weerstand tegen differentiële en lineaire cryptanalyse op te bouwen. Echter,

het gevaar van deze aanpak is dat het algoritme kwetsbaar wordt voor andere aanvallen.

Er wordt een aanval ontwikkeld die werkt op Feistel cijfers met een beperkt aantal ronden en een niet-surjektieve F-functie, of meer algemeen, een niet-gebalanceerde F-functie. De aanval bewijst dat Feistel cijfers met een niet-surjektieve F-functie minstens 8 à 10 ronden moeten hebben om veilig te zijn.

Deze aanval wordt toegepast op verschillende algoritmes die ontwikkeld werden met de CAST ontwerpstrategie [2, 3, 48] en ook op LOKI91 [17]. LOKI91 gereduceerd tot 7 ronden kan gebroken worden met deze aanval. De aanval toont aan dat de CAST algoritmes met 8 ronden slechts marginale veiligheid bieden. Bovendien werd er een duidelijke zwakheid geïdentificeerd in het sleutelschema van een enkele vroege voorstellen voor CAST algoritmes.

6 Ontwerpstrategie & Bouwblokken

De eerste theoretische beschouwingen over het ontwerp van encryptie-algoritmes kunnen gevonden worden in het werk van C.E. Shannon [122], waar voorgesteld wordt om eenvoudige substituties af te wisselen met transformaties die een sterke 'vermenging' veroorzaken. Het resultaat van de combinatie is dat *"iedere significante statistische relatie tussen ingang en uitgang van het encryptie-algoritme zeer ingewikkeld is en sterk afhankelijk van alle betrokken parameters—de redundantie is verspreid (diffused) en verborgen (confused) door de vermengende transformatie."*

Vervolgens zijn er verschillende strategieën voorgesteld door H. Feistel [36], X. Lai, J.L. Massey en S. Murphy [68, 69], L.R. Knudsen en K. Nyberg [96, 97] en anderen. In [26] ontwikkelde J. Daemen de *strategie van het brede spoor.*

In de strategie van het brede spoor wordt de rondetransformatie opgebouwd uit een aantal transformaties:

1. een niet-lineaire substitutie die opereert op blokken van p bits,
2. een lineaire diffusielaag die de verschillende p-bit blokken vermengt, en
3. een affiene sleuteltoevoeging.

Een vierde belangrijke bouwblok voor een goed blokcijfer is het sleutelschema. Aan de laatste twee bouwblokken werd in de oorspronkelijke formulering van de ontwerpstrategie minder aandacht besteed.

De verschillende componenten worden onafhankelijk geëvalueerd ten opzichte van een ontwerpcriterium en geselecteerd. Hierdoor is het mogelijk om algoritmes te construeren die met een zeer grote waarschijnlijkheid bestand zijn tegen lineaire en differentiële cryptanalyse. Dit gebeurt door de niet-lineaire

bouwblokken zo te kiezen dat de maximale kans van een differentiële karakteristiek over één blok klein is en door de lineaire bouwblokken zo te kiezen dat de uitvoer van de niet-lineaire bouwblokken van de vorige ronde optimaal gespreid wordt over de invoer van de niet-lineaire bouwblokken in de volgende ronde. Hetzelfde effect wordt beoogd voor de correlaties tussen ingangsbits en uitgangsbits van de rondetransformatie. Hoewel deze strategie geen bewijsbaar veilige constructies oplevert, lijken de goede diffusie-eigenschappen en de sterke niet-lineariteit toch sterke algoritmes te waarborgen.

6.1 Nieuwe Elementen in de Ontwerpstrategie

De belangrijkste aanvulling op de strategie van het brede spoor wordt gevormd door de introductie van lineaire codes. De toepassing van codetheorie laat toe om op eenvoudige wijze transformaties met optimale diffusie-eigenschappen te construeren. Hiervoor wordt gebruik gemaakt van 'Maximum Distance Separable codes' (MDS-codes). Bovendien kunnen een aantal eigenschappen van de transformaties op elegante wijze bewezen worden.

Transformaties met optimale diffusie hebben een bepaalde minimale complexiteit; voor sommige toepassingen is deze complexiteit te groot. Het is dan mogelijk om transformaties te gebruiken die in plaats van al na 1 ronde pas een goede diffusie garanderen na 3 of meer ronden. Dit leidt uiteindelijk tot constructies waarbij de diffusietransformatie niet meer gelijk gekozen worden voor alle ronden zodat de verschillende rondetransformaties niet meer aan mekaar gelijk zijn.

Er zijn in de cryptografische literatuur al verschillende methodes beschreven om goede niet-lineaire substitutietabellen expliciet te construeren [93]. Voor toepassingen waar de inherente mathematische structuur van deze constructies ongewenst is, werd bestudeerd wat het effect is van kleine willekeurige wijzigingen in de tabellen en wat de verwachte performantie is van willekeurig geconstrueerde tabellen.

De theorie van de lineaire codes kan ook gebruikt worden om sleutelschema's te construeren met bewijsbare eigenschappen.

6.2 Valluikcijfers

Valluikcijfers zijn cijfers met een verborgen structuur, *een valluik*. Voor gebruikers die de verborgen structuur niet kennen lijken het veilige algoritmes, maar mensen die het valluik kennen, kunnen het cijfer eenvoudig breken. Er wordt een manier voorgesteld om valluiken te verbergen in blokcijfers. Dit kan zo gebeuren dat het met de beschikbare rekenkracht onmogelijk is om het valluik te detecteren, zelfs als men weet wat de algemene structuur van het valluik is.

'Willekeurige' substitutietabellen kunnen op eenvoudige wijze zo aangepast worden dat het verband tussen bepaalde ingangsbits en uitgangsbits bijna lineair wordt. Op die manier zal een lineaire aanval zeer eenvoudig worden voor iemand die het verband kent. Door de tabellen aangepast te dimensioneren (bv. 10 ingangsbits en 40 uitgangsbits), is het voor iemand anders niet doenbaar om het verband te vinden.

Gelijkaardige valluiken kunnen geconstrueerd worden door de verschillende tabellen van een rondetransformatie in functie van mekaar te bepalen, zodat de afzonderlijke tabellen veilig genoeg lijken, maar hun combinatie weer een verborgen relatie bevat.

Deze valluikcijfers zouden kunnen gebruikt worden in een encryptieschema met publieke sleutels. De techniek toont ook aan dat een men niet blind mag vertrouwen op andermans ontwerpen, zeker niet als er gebruik gemaakt wordt van 'willekeurige' of geheime tabellen.

7 Nieuwe Blokcijfers

Twee blokcijfers werden ontworpen met behulp van de strategie van het brede spoor.

7.1 SHARK

SHARK is geen Feistel cijfer, maar heeft een uniforme rondetransformatie. Het vercijfert klaartekstblokken van 64 bits, de sleutel kan elke lengte tussen 64 en 128 bits hebben. SHARK gebruikt een lineaire transformatie met optimale diffusie en sterk niet-lineaire substitutietabellen. Het is al na zes ronden bestand tegen differentiële en lineaire cryptanalyse.

Voor een efficiënte implementatie kan de diffusietransformatie opgenomen worden in de substitutietabellen. Vooral als het onderliggend platform 64-bit operaties ondersteunt leidt dit tot een hoge encryptiesnelheid (6.3 Mbyte/s op een 266 MHz DEC-Alpha).

De beste bekende aanval op SHARK breekt 3 ronden met 2^9 gekozen klaarteksten. De aanval is echter niet uitbreidbaar naar meer ronden.

7.2 SQUARE

Ook SQUARE heeft een uniforme rondetransformatie. Het vercijfert klaartekstblokken van 128 bits, onder invloed van een sleutel van 128 bits. De lineaire transformatie van SQUARE heeft geen optimale diffusie, maar is wel zo gekozen dat de diffusie over 4 ronden zeer sterk is De keuze van de lineaire transfor-

matie laat toe om SQUARE efficiënt te implementeren op een heel scala van processoren, van goedkope smart cards tot performante werkstations.

SQUARE telt 8 ronden en is bestand tegen differentiële en lineaire cryptanalyse. Er bestaat wel een aanval die 6 ronden breekt met 2^{32} gekozen klaarteksten en een werkfactor van 2^{73} encrypties, maar deze aanval kan niet uitgebreid worden naar 7 of meer ronden [27].

Op een 100 MHz Pentium haalt een assembler implementatie van SQUARE een encryptiesnelheid van 4.94 Mbyte/s. De referentie-implementatie in C haalt 2.63 Mbyte/s.

8 Besluit en Open Problemen

Blokcijfers zijn per definitie algoritmen met een ingewikkelde structuur, die het ontwerp en de analyse ervan moeilijk maken. Het eerste deel van dit proefschrift bevat onze bijdragen aan de analyse van blokcijfers. Het tweede deel bevat onze aanvullingen op een ontwerpstrategie voor cryptografische algoritmen en twee nieuwe ontwerpen.

De bijdragen aan de cryptanalyse van blokcijfers bestaan vooral uit de introductie van enkele gevorderde statistische technieken. De technieken zijn vooral bruikbaar in situaties waar de gebruikelijke aannames niet gelden, bijvoorbeeld wanneer het blokcijfer niet gebruikt wordt in de standaard ECB-mode of wanneer de karakteristieken van het cijfer sterk afhangen van de gebruikte sleutel. Er wordt ook een nieuwe aanval geïntroduceerd.

De complexiteit van blokcijfers en hun analyses kan gemakkelijk aanleiding geven tot een soort fatalisme: er bestaan (nog) geen bewijsbaar veilige blokcijfers en het vinden van de zwakheden in een blokcijfer vraagt een hele inspanning. De meeste blokcijfers bevinden zich ergens tussen de polen 'waarschijnlijk veilig' en 'aantoonbaar onveilig'. Dit mag echter niet leiden tot een zorgeloze houding tijdens het ontwerp. De snelle cryptanalyse van MacGuffin en het feit dat verschillende belangrijke ontwerpfouten zijn aangetoond in de eerste CAST-algoritmes tonen dit aan.

Het ontwerp van blokcijfers gebeurt meestal op een ad hoc manier; ook hier zou veel gewonnen kunnen worden door een beter aanwenden van het beschikbare arsenaal aan wiskundige technieken. Een eerste stap werd gezet door een verband te leggen tussen sommige bouwblokken van een rondetransformatie en lineaire codes.

Enkele ideeën voor verder onderzoek zijn de volgende:

- De weerstand van een algoritme tegen een aanval met afgeknotte differentiële bundels wordt meestal niet bepaald door de niet-lineaire bouwblokken, maar wel door de algemene structuur van de rondetransformatie.

Het zou interessant zijn om te kunnen beschikken over een heuristiek om de weerstand van een blokcijfer te bepalen. Ook is nog niet bekend of de techniek kan uitgebreid worden naar lineaire cryptanalyse.

- Een verdere studie van sleutelafhankelijke relaties in blokcijfers.
- Studie van de veiligheid van de nieuwe cijfers die ontwikkeld zijn met de CAST ontwerpstrategie.
- Een verdere studie van codetheorie om betere compromissen te vinden tussen goede diffusie en snelle implementaties.
- Onderzoeken of het mogelijk is om goede niet-lineaire substitutietabellen te construeren zonder tegelijkertijd wiskundige structuren in te bouwen die gevaarlijk kunnen zijn.
- Een praktisch ontwerp voor een asymmetrisch encryptieschema dat gebruik maakt van blokcijfers met een valluik.
- Cryptanalyse van SHARK en SQUARE.