STATISTICAL CRYPTANALYSIS OF BLOCK CIPHERS

THÈSE N^{O} 3179 (2005)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut de systèmes de communication

SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Pascal JUNOD

ingénieur informaticien dilpômé EPF
de nationalité suisse et originaire de Sainte-Croix (VD)

acceptée sur proposition du jury:

Prof. S. Vaudenay, directeur de thèse
Prof. J. Massey, rapporteur
Prof. W. Meier, rapporteur
Prof. S. Morgenthaler, rapporteur
Prof. J. Stern, rapporteur

Lausanne, EPFL
2005

to Mimi and Chloé

Acknowledgments

First of all, I would like to warmly thank my supervisor, Prof. Serge Vaudenay, for having given to me such a wonderful opportunity to perform research in a friendly environment, and for having been the perfect supervisor that every PhD would dream of. I am also very grateful to the president of the jury, Prof. Emre Telatar, and to the reviewers Prof. em. James L. Massey, Prof. Jacques Stern, Prof. Willi Meier, and Prof. Stephan Morgenthaler for having accepted to be part of the jury and for having invested such a lot of time for reviewing this thesis.

I would like to express my gratitude to all my (former and current) colleagues at LASEC for their support and for their friendship: Gildas Avoine, Thomas Baignères, Nenad Buncic, Brice Canvel, Martine Corval, Matthieu Finiasz, Yi Lu, Jean Monnerat, Philippe Oechslin, and John Pliam. Without them, the EPFL (and the crypto) would not be so fun!

Without their support, trust and encouragement, the last part of this thesis, FOX, would certainly not be born: I owe to MediaCrypt AG, especially to Ralf Kastmann and Richard Straub many, many, many hours of interesting work. I thank them warmly!

The worldwide cryptologic community is especially friendly; I would like to thank all the researchers throughout the world for having shared with me ideas, discussions about cryptology, or simply nice moments during conferences: Alex Biryukov, Emmanuel Bresson, Nicolas Courtois, Pierre-Alain Fouque, Mounir Idrassi, Robert Johnson, Liam Keliher, Sébastien Kunz-Jacques, Simon Künzli, Marco Maccheti, Gwenaëlle Martinet, Frédéric Muller, Phong Nguyen, Gilles Piret, Thomas Pornin, Emmanuel Prouff, Jean-Jacques Quisquater, Frédéric Raynal, Renato Renner, Michael Scott, François-Xavier Standaert, and David Wagner. Especially, I would like to thank Prof. Ueli Maurer for having opened my eyes on cryptology during his outstanding lectures at ETH Zurich.

Since acknowldegments are definitely the most delicate part of a thesis, I would like to thank all people I might have forgotten. Last but not least, I would like to express my gratitude to my family, my close friends and to all the geeks of Kayak-Club Lausanne for sharing with me all (or almost all) the time where I do not think about crypto. Finally, I would like to thank my wife Myriam for her love and for her infinite patience. This thesis is dedicated to her and to our daughter Chloé.

Abstract

Since the development of cryptology in the industrial and academic worlds in the seventies, public knowledge and expertise have grown in a tremendous way, notably because of the increasing, nowadays almost ubiquitous, presence of electronic communication means in our lives. Block ciphers are inevitable building blocks of the security of various electronic systems. Recently, many advances have been published in the field of public-key cryptography, being in the understanding of involved security models or in the mathematical security proofs applied to precise cryptosystems. Unfortunately, this is still not the case in the world of symmetric-key cryptography and the current state of knowledge is far from reaching such a goal. However, block and stream ciphers tend to counterbalance this lack of "provable security" by other advantages, like high data throughput and ease of implementation.

In the first part of this thesis, we would like to add a (small) stone to the wall of provable security of block ciphers with the (theoretical and experimental) statistical analysis of the mechanisms behind Matsui's linear cryptanalysis as well as more abstract models of attacks. For this purpose, we consider the underlying problem as a statistical hypothesis testing problem and we make a heavy use of the Neyman-Pearson paradigm. Then, we generalize the concept of linear distinguisher and we discuss the power of such a generalization. Furthermore, we introduce the concept of sequential distinguisher, based on sequential sampling, and of aggregate distinguishers, which allows to build sub-optimal but efficient distinguishers. Finally, we propose new attacks against reduced-round version of the block cipher IDEA.

In the second part, we propose the design of a new family of block ciphers named FOX. First, we study the efficiency of optimal diffusive components when implemented on low-cost architectures, and we present several new constructions of MDS matrices; then, we precisely describe FOX and we discuss its security regarding linear and differential cryptanalysis, integral attacks, and algebraic attacks. Finally, various implementation issues are considered.

Résumé

Depuis le développement de la cryptologie dans les mondes industriel et académique, les connaissances et l'expertise publique ont crû de manière soutenue, notamment en raison de l'omniprésence des moyens de communication électronique dans la vie de tous les jours. Les algorithmes de chiffrement par blocs sont ainsi des briques de base incontournables de la sécurité de nombreux systèmes. Récemment, de nombreuses avancées ont été publiées dans le domaine de la cryptographie à clef publique, que ce soit dans la compréhension des modèles de sécurité en jeu, ou dans les preuves mathématiques de sécurité appliquées à des systèmes bien précis. Malheureusement, le monde de la cryptographie symétrique reste clairement en retrait, bien que la situation évolue lentement. Les algorithmes de chiffrement par blocs, ou par flot, tendent ainsi à compenser leur manque de "sécurité prouvée" par d'autres atouts, tels qu'un débit de chiffrement élevé et une certaine facilité d'implantation.

Dans la première moitié de cette thèse, nous tentons d'ajouter une (modeste) brique à l'édifice de la sécurité prouvée des algorithmes de chiffrement par blocs en analysant (théoriquement et expérimentalement) les mécanismes statistiques sous-jacents à la cryptanalyse linéaire de Matsui ainsi qu'à des modèles d'attaques plus abstraits. Pour atteindre ce but, nous interprétons le problème comme un test d'hypothèses statistiques en utilisant notamment le paradigme de Neyman-Pearson. Nous généralisons ensuite le concept de distingueur linéaire et nous en discutons la puissance. Nous introduisons également le concept de distingueur séquentiel, basé sur l'échantillonage séquentiel, ainsi que celui de distingueur à aggrégats, qui permet de construire des distingueurs certes sub-optimaux, mais néanmoins efficaces. Finalement, nous proposons une série de nouvelles attaques contre des versions réduites de l'algorithme IDEA.

Dans la seconde moitié, nous proposons une nouvelle famille d'algorithmes de chiffrement par blocs, baptisée FOX. Pour cela, nous étudions sur des architectures à bas coût l'efficacité de composants de diffusion optimaux, et nous proposons plusieurs nouvelles constructions de matrices MDS. Enfin, nous décrivons précisément FOX, nous étudions sa sécurité vis-à-vis des attaques linéaires, différentielles, intégrales et algébriques, et nous discutons finalement les aspects liés à son implantation.

Contents

Chapter 1

Introduction

Since the appearance of cryptology in the industrial and academic worlds in the seventies, public knowledge and expertise in this fascinating scientific domain have grown in a tremendous way, notably because of the increasing, nowadays almost ubiquitous, presence of electronic communications means in our lives.

Interestingly, we note that the scientific development of cryptology has followed different paths: for instance, "provable secure" public-key cryptosystems, i.e. cryptographic algorithms for which the security can be mathematically proved as difficult as a well-known (supposedly hard to solve) computational problem, exist since a few years, and the involved models of security have been seriously studied and are nowadays well understood. Unfortunately, this is still not the case in the world of symmetric-key cryptography, where algorithms are widely deployed in many applications (being in the civilian life or for military purposes), and the current state of knowledge is far from reaching such a goal. Block and stream ciphers tend to counterbalance this lack of "provable security" by other advantages, like high data throughput and ease of implementation, and it is not astonishing that practical solutions involve both public-key and symmetric worlds, exploiting their advantages in parallel.

During the last 30 years, the academic research on the security of block ciphers has evolved from an *empirical* way to solve the problem of designing a secure algorithm to an *heuristic* one, where a list of well-established and well-understood security properties that a block cipher must fulfill in order to be secure is available. But ultimately (and unfortunately), the experience and the feelings of the designers often matter more for which concerns the (good or bad) quality of the result, and more importantly, the perceived security of a block cipher is still heavily dependent of the talent, the intuition, and the time at disposal of the people attempting at breaking it!

Finally, we note that the gap between the present state of knowledge

and real, practical provable security for block ciphers tends to fill only very slowly, mainly because of the rather slow pace of the research.

Thesis Outline and Contributions

The goals of the thesis are twofold: on the one hand, we aim at adding a (small) stone to the wall of provable security of block ciphers with the statistical analysis of the mechanisms behind Matsui's linear cryptanalysis. On the other hand, we present the design of a new family of block ciphers named FOX developed on behalf of the company MediaCrypt AG [218]. This thesis is organized as follows:

- The chapter §2 is devoted to the description of the current (publicly available) knowledge about block ciphers. After having reviewed some terminology in §2.1, we describe in §2.2 the three block ciphers having the most significant practical impact at the time of writing, namely the former American standard, DES, and the current one, AES, as well as IDEA, and we discuss their security. In the same section, we briefly address the classical modes of operation of block ciphers. Then, in §2.3, we focus on the known attacks developed to break block ciphers; we successively discuss black-box, statistical, algebraic as well as more "exotic" attacks. Finally, we describe in §2.4 three models of security, namely Shannon's perfect security, (time- or memory-) bounded adversaries, and ad-hoc proofs of security.

- Chapter §3 begins by recalling the statistical tests framework due to Neyman and Pearson in §3.1, since we make heavy use of it in the subsequent sections. In §3.2, we present a new analysis of Matsui's linear cryptanalysis against DES, we discuss its optimality, we show that it can be improved using an optimal key-ranking procedure, and we finally present some experimental results issued from the implementation of both the original and the improved versions. Then, in §3.3, we study the statistical modelization of generic *distinguishers*, and we derive the precise description of optimal linear and differential distinguishers and we analyze their performances; furthermore, we introduce and discuss the concepts of generalized linear distinguishers, of aggregate distinguishers, and of sequential distinguishers. Finally, in §3.4, we present some new attacks on reduced-round versions of IDEA.

- While the chapter §3 was undoubtedly oriented towards the cryptanalysis of block ciphers, the chapter §4 is more (but not completely) dedicated to the design of block ciphers. In §4.1, we review different strategies commonly used to design block ciphers and we discuss their advantages as well as their drawbacks. In §4.2, we focus on the design

of fast diffusive components, and we present several new constructions. Last but not least, the FOX family of block ciphers is described in §4.3, as well as first attempts to break it, a discussion of its security, and implementation issues.

Our contributions can be summarized as follows:

- A new statistical modelization of three variants of Matsui's linear cryptanalysis has been proposed and their theoretical success probability has been computed.
- The concept of "optimal key-ranking procedure" has been formally defined, allowing the optimization of Matsui's attack against DES.
- Matsui's attack against DES, as well as our improved version, have been implemented on conventional PCs and run 21 times.
- The shape of optimal computationally unbounded distinguishers between two binary random sources has been proposed, as well as tight bounds on their advantage. These results have been applied to conventional linear and differential distinguishers.
- The concept of "generalized linear distinguisher" has been proposed, as well as the computation of their power.
- The concept of "aggregate distinguishers" has been proposed and their benefits have been discussed.
- The concept of "sequential distinguishers", based on sequential sampling, have been proposed and they have been shown, given a fixed advantage, to be optimal with respects to the number of required samples.
- New linear-like attacks against reduced-round versions of the block cipher IDEA have been demonstrated.
- A study of fast diffusive components for block ciphers has been done, and several new constructions have been proposed.
- The design of a new family of block ciphers named FOX has been proposed, as well as first attempts to break it, a discussion of its resistance towards several attacks, and a discussion of implementation issues.

Most of the results described in this thesis have been published: a paper [144] describing the statistical modelization of Matsui's linear cryptanalysis, as well as experimental results derived from its implementation has been presented at Selected Areas in Cryptography in 2001. The description of optimal linear and differential distinguishers, obtained by considering the

task of a distinguisher as a statistical hypothesis test, the application of Chernoff bounds derived in §3.3.2, and the discussion of sequential distinguishers have been published in a paper [145] presented at EUROCRYPT 2003. Optimal key-ranking procedures and their application to the linear cryptanalysis of DES have been published in a paper [149] co-written with Serge Vaudenay and presented at FAST SOFTWARE ENCRYPTION in 2003. The FOX family of block ciphers, which is a joint work with Serge Vaudenay, has been the subject of a paper [150] presented at SELECTED AREAS IN CRYPTOGRAPHY in 2004 and has been the subject of two European patent applications [146, 148]; the complete specifications of the ciphers have been published as an EPFL technical report [147] and an updated and corrected version is available in another technical report [151]. The work about fast diffusive components, which is a joint work with Serge Vaudenay as well, is the subject of another paper [152] presented at SELECTED AREAS IN CRYPTOGRAPHY in 2004, too. The notion of generalized linear distinguishers, obtained in close collaboration with Thomas Baignères and Serge Vaudenay, has been published in [14] and presented at ASIACRYPT 2004. Finally, the new attacks against reduced round version of IDEA are described in a paper which follows, at the time of writing, the submission process.

Chapter 2

A Brief Overview of Block Ciphers

This chapter aims at offering a wide, although inevitably not exhaustive, overview of the world of block ciphers. It is organized as follows: §2.1 defines the terminology as well as basic concepts necessary to the good understanding of this thesis; then, §2.2 presents and describes in details three block ciphers which are probably the most important and the most widely disseminated ones nowadays, namely DES, IDEA, and AES. The part §2.3 focuses on the presentation of the current state-of-the-art in the domain of *cryptanalysis* of block ciphers, and §2.4 treats different kinds of *security proofs* applied to block ciphers.

2.1 Terminology

2.1.1 Basic Definitions

Roughly expressed, a *symmetric-key block cipher* is a cryptographic system whose principal aim is to guarantee the *confidentiality* of data. Mao [198] gives the following definition of a *cryptographic system*.

Definition 2.1.1 (Cryptographic system). *A* cryptographic system *consists of the following:*

- *a* plaintext message space $\mathcal{P}$ *which is a set of strings over some alphabet;*
- *a* ciphertext message space $\mathcal{C}$ *which is set of possible ciphertext messages;*
- *an* encryption key space $\mathcal{K}$ *which is the set of possible encryption keys, and a* decryption key space $\mathcal{K}'$ *which is the set of possible decryption keys;*
- *an efficient* key generation algorithm*:* $\gamma : \mathbb{N} \rightarrow \mathcal{K} \times \mathcal{K}'$*;*

- *an efficient* encryption algorithm*:* $\varepsilon : \mathcal{P} \times \mathcal{K} \rightarrow \mathcal{C}$*;*
- *an efficient* decryption algorithm*:* $\varepsilon' : \mathcal{C} \times \mathcal{K}' \rightarrow \mathcal{P}$*.*

For a security parameter 1^ℓ*, the key generation algorithm outputs a key pair* $(k, k') \in \mathcal{K} \times \mathcal{K}'$ *of length* ℓ*. For* $k \in \mathcal{K}$ *and* $p \in \mathcal{P}$*, we denote by*

$$c = \varepsilon_k(p)$$

the encryption operation *and by*

$$p = \varepsilon'_{k'}(c)$$

the decryption operation*. It is furthermore necessary that for all* $m \in \mathcal{M}$ *and all* $k \in \mathcal{K}$*, there exists* $k' \in \mathcal{K}'$ *such that*

$$\varepsilon'_{k'}\left(\varepsilon_k(p)\right) = p$$

A block cipher can be seen in a simple way as a deterministic, memoryless, invertible function mapping an n-bit plaintext block $p \in \{0,1\}^n$ to an n-bit ciphertext block $c \in \{0,1\}^n$; furthermore, this function is parametered by a single ℓ-bit secret key $k \in \{0,1\}^\ell$; in other words, and using the terminology of Def. 2.1.1, $\varepsilon = \varepsilon'$, $\mathcal{K} = \mathcal{K}'$, and $k = k'$. The notion of *symmetry* in block ciphers comes hence from the fact that the same key is used for *both* encryption and decryption operations; the opposed notion is the *asymmetric* or *public-key cryptography* [91,92] which uses different, related keys for both operations.

In order that a ciphertext decrypts to a *unique* plaintext for a given fixed key, it is necessary that the encryption function is a bijection; this restrict the number of block ciphers to the $(2^n)!$ permutations on n-bit values. As this value is extremely large[1] for common values of n (64 or 128 bits), the size of the key further restricts the number of reachable permutations. Usual key lengths (up to 256 bits) imply that this number is actually an infinitesimally small fraction of all possible permutations. Informally, the goal is to make it practically impossible to retrieve the plaintext from the ciphertext without any knowledge on the secret key.

The concept of block cipher is summarized in a formal way in Def. 2.1.2, taken out of [221]. As the a block cipher is memoryless, we will see it as a *function* and therefore use a different notation.

Definition 2.1.2 (Symmetric-Key Block Cipher). *An* n*-bit* symmetric-key block cipher *is a function* $\mathsf{e} : \{0,1\}^n \times \{0,1\}^\ell \rightarrow \{0,1\}^n$ *such that for each key* $k \in \{0,1\}^\ell$*, the* encryption function $\mathsf{e}(p,k)$*, written* $\mathsf{e}_k(p)$*, is an invertible mapping from* $\{0,1\}^n$ *to* $\{0,1\}^n$*. The inverse mapping is the* decryption function*, denoted* $\mathsf{d}_k(c)$*, where* $c = \mathsf{e}_k(p)$ *denotes the ciphertext* c *resulting from the encryption of plaintext* p *under key* k*.*

[1] Using Stirling's formula, one can show that $\log(2^n!) \approx 2^n \cdot (n - 1.44)$.

Note that it is possible to imagine *probabilistic* block ciphers which take some randomness in addition to the key as input in order to select a mapping in a non-deterministic way; thus, each time a plaintext block p is encrypted under the key k, the output is a *set* of eligible ciphertext blocks and the function chooses one ciphertext block c out of this set. Since the encryption function is essentially one-to-many, the requirement for invertibility implies data expansion, which is a disadvantage of randomized encryption; furthermore, gathering "good" randomness is not a trivial problem in the real world. However, depending on the strength of the security model under consideration, some randomness may be required. In practice, this property is often shifted to the use of randomized *modes of operations* (see §2.2.4). We will not address probabilistic block ciphers in more details in this thesis.

Virtually all block ciphers are *product ciphers*, i.e. they combine at least two or more transformations in a manner intending that the resulting cipher is more secure than the individual components. The underlying idea is to build a complex encryption function by combining several simple operations which offer complementary, but individually insufficient security properties. A very important class of product ciphers is the category of *iterated* block ciphers (see Fig. 2.1). The key idea is to iterate the same round function f several times on the plaintext block p. More precisely, an *iterated block cipher* is a block cipher involving the sequential repetition of an internal function f called a *round function*. Parameters include the number of rounds r, the block bit size n and the bit size ℓ of the input key k from which r *subkeys* $k^{(i)}$ (also called *round keys*) are derived. For invertibility purposes, the round function f must be a bijection on the round input for each value $k^{(i)}$.

Various schemes are used to build modern iterated block ciphers, like substitution-permutation networks (SPNs), Feistel schemes and variants, and many others; we will discuss them later on. The round keys $k^{(i)}$ are derived from the key k by an algorithm named *key-schedule algorithm*. Iterated block ciphers have several advantages: it is possible to implement them in an efficient way, because one can reuse the same code or circuit in each round. Furthermore, it is easier to analyze them in a security point of view since several theoretical results concerning iterated block ciphers are available.

A more "high-level" way to build a new block cipher consists in combining directly block ciphers. The key point is that the keys used by the individual block ciphers should be statistically independent; however, the distinction is not always clear. A *cascade cipher* is usually defined [221] as being the concatenation of $s \geq 2$ block ciphers (called *stages*), each with (statistically) independent keys: the plaintext is the input of the first stage, the output of stage i is the input of stage $i+1$ and the output of stage s is defined to be the ciphertext. *Multiple encryption* is similar to cascade ciphers, but the keys may be dependent and the stage block ciphers may

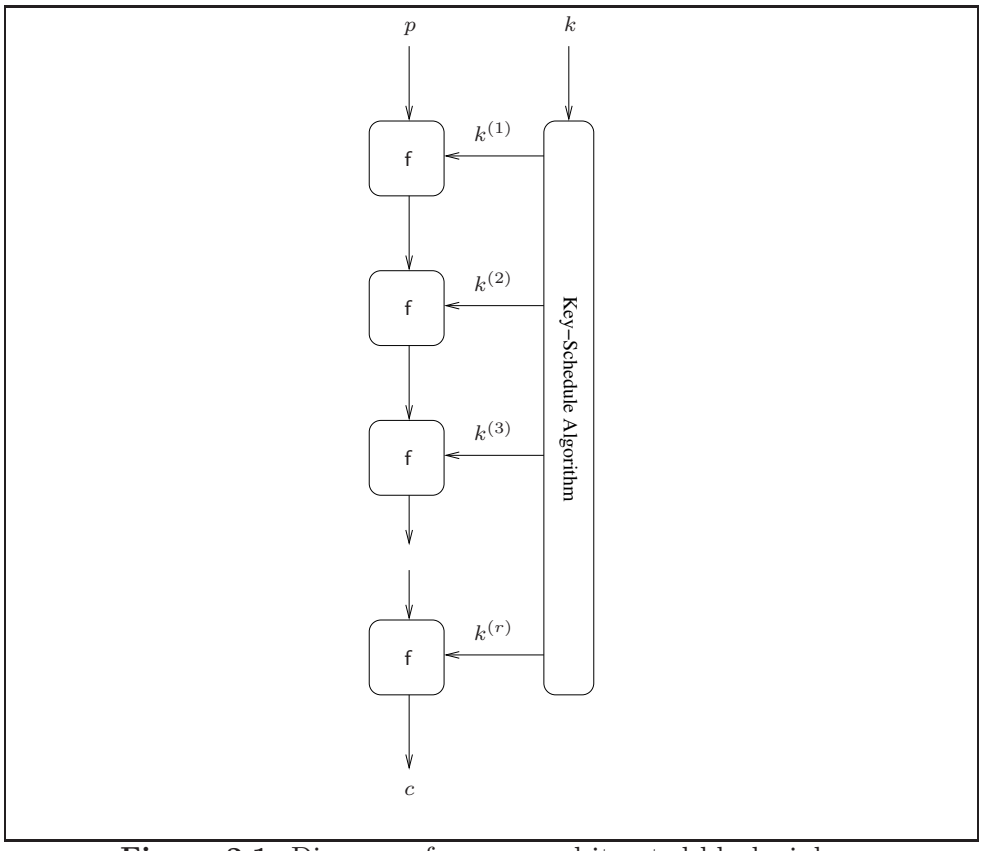

Figure 2.1: Diagram of an r-round iterated block cipher.

be either a block cipher e or its corresponding decryption function d. Most common constructions of multiple encryption are the *double encryption* and *triple encryption* (see Def. 2.1.3 and Def. 2.1.4).

Definition 2.1.3 (Double Encryption). *For a block cipher* e_k*,* double encryption *is defined as* $\mathsf{e}(x) = \mathsf{e}_{k_2}(\mathsf{e}_{k_1}(x))$*, where* k_1 *and* k_2 *are statistically independent.*

Definition 2.1.4 (Triple Encryption). *For a block cipher* e_k*,* triple encryption *is defined as*

$$\mathsf{e}(p) = \mathsf{e}_{k_3}^{(3)}\left(\mathsf{e}_{k_2}^{(2)}\left(\mathsf{e}_{k_1}^{(1)}(p)\right)\right)$$

where $\mathsf{e}^{(i)}$ *denotes either* $\mathsf{e}_k(.)$ *or* $\mathsf{d}_k(.)$*. The case*

$$\mathsf{e}(p) = \mathsf{e}_{k_3}^{(3)}\left(\mathsf{d}_{k_2}^{(2)}\left(\mathsf{e}_{k_1}^{(1)}(p)\right)\right)$$

is called EDE triple-encryption*; the sub-case* $k_1 = k_3$ *is called* two-key triple encryption.

2.1.2 Good and Bad Block Ciphers

The rigorous evaluation of block ciphers is an extremely difficult and time-consuming task since several criteria have influence on the good (or bad) quality of a block cipher:

- *Security level*: A very important criterion in the evaluation of a block cipher is obviously its *estimated* security level. Unfortunately, the current state of the science does not allow (up to now) to prove in a mathematical, rigorous way whether a given (practical) block cipher is secure or not; although the concept of *perfect security* [295] has been formalized several decades ago, perfect ciphers (like the *one-time pad* [321], for instance) are very impractical for a real use, as they require at least as many key bits as the message length. This fact explains why evaluations projects, like the AES [3], NESSIE [247], or CRYPTREC [72] efforts, are necessary. Thus, at this time, some subjectivity will inevitably be present in the estimation of the security of a block cipher. Does a given algorithm come with a certain "proof of security"? Does it has withstood expert cryptanalysis from several people over a substantial time period (i.e. years)? Responses to this kind of questions help to take a decision regarding the security criterion, but do not give any formal guarantee about the security towards (yet) unknown attacks.

- *Throughput*: Block ciphers are often used to encrypt large amounts of data; this makes throughput an important evaluation criterion as well. Throughput is related to the complexity of the cryptographic mapping and the degree to which the algorithm is tailored to a particular platform or implementation context. One often differentiates hardware and software cases, the speed of the algorithm setup, the key setup, a key change and the encryption and decryption operations.

- *Flexibility*: Usually, an expected important property of a block cipher is that it offers a large *flexibility* at different points of view. For instance, a flexible algorithm may offer several possible block and key sizes, allowing to tailor an instance of the block cipher to precise external requirements. Another flexibility form concerns implementation issues. If the block cipher under consideration can be implemented on various platforms, i.e. on fast 32-bit, 64-bit microprocessors, on hardware (either as an ASIC or on a FPGA), on low-cost 8-bit architectures (like a smartcard) while keeping an acceptable throughput, then one can consider it as flexible. Finally, a block cipher can be used as a building block in various (but unusual) cryptographic constructions (like a hash function, an authentication code, or a stream cipher); if it offers an acceptable security level in all of these situations, then one can consider that it is a flexible block cipher.

2.2 Examples of Block Ciphers

It is astonishing how many different designs of block ciphers have been proposed in the academic literature; Fig. 2.2 aims at listing the most important ones. Few of them have a real impact in the practical life, but most of them suggest interesting questions and open problems about their security. In the next parts, we describe precisely three block ciphers which are probably the most frequently encountered ones in practice, namely DES, IDEA, and AES.

2.2.1 Data Encryption Standaard (DES)

The "*Data Encryption Standard*" (DES), also known as the "*Data Encryption Algorithm*" (DEA) by ANSI or as DEA-1 by ISO, respectively, has been a *de facto* worldwide symmetric encryption standard for more than two decades (the latest version[2] of the NIST standard is [246]). In this section, we first recall some historical perspectives and we sketch its definition, since we will use it in a later part.

[2]At the time of writing, the NIST has announced that the standard will shortly be withdrawn.

Algorithm	Reference(s)	Algorithm	Reference(s)
AES	[245]	LOKI	[49, 50]
Akelarre	[4]	LOKI 97	[51]
Anubis	[17]	Lucifer	[301]
BEAR	[6]	MacGuffin	[41]
BKSQ	[80]	Magenta	[138]
Blowfish	[284]	Mars	[53]
Camellia	[7]	Misty1	[205]
CAST128	[1]	Nimbus	[196]
CAST256	[2]	Noekeon	[78]
CS Cipher	[305]	NUSH	[184]
Crypton	[186, 187]	PES	[181]
DEAL	[163]	Q	[217]
DES	[242]	RC2	[278]
DESX	[160]	RC5	[277]
DFC	[114]	RC6	[279]
DFCv2	[118]	Redoc II	[75]
E2	[8]	Rijndael	[79, 80]
FEAL	[226, 296]	SAFER K-64	[199]
FEALNX	[227]	SAFER+	[200]
FOX	§4.3	SAFER++	[201]
Frog	[111]	SC2000	[298]
GOST	[116]	Serpent	[23]
Grand Cru	[44]	SHACAL	[123]
Hasty Pudding	[288]	Shark	[275]
Hierocrypt L1	[64]	Skipjack	[244]
Hierocrypt3	[63]	Square	[82]
ICE	[177]	TEA	[327]
IDEA	[182]	Twofish	[286]
Kasumi	[99]	Triple-DES	[93]
Khafre	[223]	UES	[124]
Khazad	[18]	VINO	[89]
Khufru	[223]	XMX	[232]
LION	[6]		

Figure 2.2: List of Block Ciphers

DES has been designed[3] by a group of permanent researchers working in the seventies for IBM Corp: Coppersmith, Konheim, Adler, Notz, Smith, Feistel, Tritter, Tuckerman, Meyer, Grossman, McNeil, Tuchmann, and Oseas.

Historical Perspectives

Interestingly, the history[4] of DES is closely related to the history of "modern" cryptology. In the early 70s, non-military research about cryptographic algorithms was nearly inexistent and very few people understood the science of cryptology. In 1972, the former US "*National Bureau of Standards*" (NBS), known nowadays as the "*National Institute of Standards and Technology*" (NIST), initiated a program with the goal of protecting computer and communications data; part of this program was the development of a *single* standard cryptographic algorithm, such that it could be tested and certified, and different equipments using it could interoperate easily.

In 1973, the NBS issued a public request for proposals; the propositions demonstrated that there was a lot of public interest in this field, but very little expertise, since none of the submissions came only close to meeting the requirements defined by the NBS. A second request was issued one year later, and the NBS eventually received a promising candidate: an algorithm based on another one developed by IBM, called Lucifer. The NBS requested help of the "*National Security Agency*" (NSA), and comments by the general public.

Many people were afraid that the NSA had modified the algorithm to install a trapdoor, complained about the reduced key size (from 128-bit to 56-bit) and the inner working of the algorithm, since the rationales behind the design of the algorithm were kept secret. In 1976, the NBS initiated two workshops, one dedicated to the mathematics of the algorithm and the possibility of a trapdoor, the other being devoted to the possibility of increasing key's length. For instance, Brickel *et al.* [47] conclude in a paper aiming at discovering the design criteria of DES S-boxes, that

> "*All the structure of the S-boxes that we have described appears to be the result of design principles. The question that remains is whether it is a complete list of the design principles used in creating the S-boxes. This question could be answered in the negative if further structure was discovered in the S-boxes that did not occur in the boxes created using these design principles.*"

[3] The names of people having worked on the design of DES have been disclosed by Coppersmith during an invited lecture at Crypto'2000 conference. See [60] as well.

[4] This part is mostly inspired by the book of Schneier [283, Chapter 12], and we refer the reader to it for more information about the history of DES.

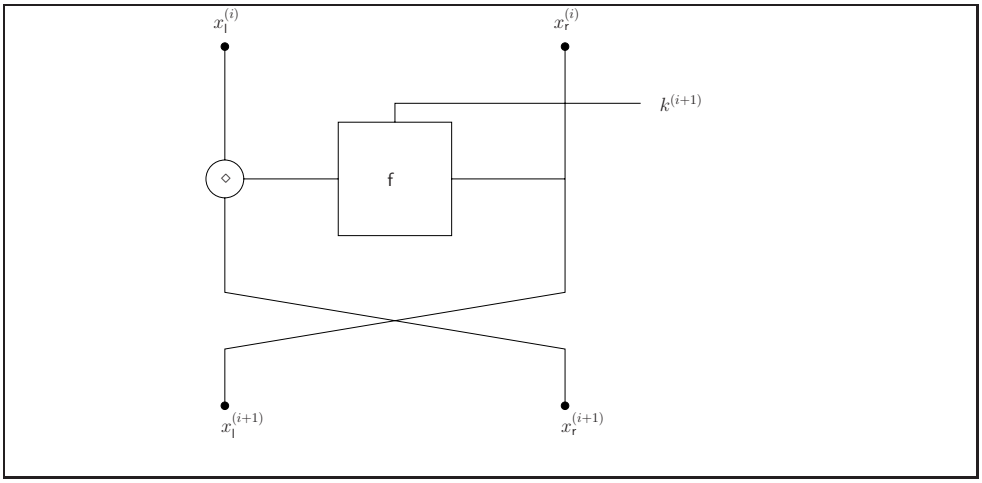

Figure 2.3: A Feistel Round Γ mapping $x_{\mathsf{l}}^{(i)}||x_{\mathsf{r}}^{(i)}$ to $x_{\mathsf{l}}^{(i+1)}||x_{\mathsf{r}}^{(i+1)}$.

Despite of criticism, DES was adopted as a federal standard on November 23, 1976 and authorized for use on all unclassified government communications. The official and initial description of the standard can be found in [242]. As the terms of the DES standards stipulate that it should be reviewed every five years, it was re-certified in 1983, 1987, 1993, and in 1999. However, in 1997, as DES was showing some signs of old age and as it can no more be considered as a secure algorithm (mainly because of its small key size, see §2.3.2), the NIST has decided to launch a process[5] in order to find a successor, known as the *Advanced Encryption Standard* (AES).

Description of DES

DES is a *Feistel cipher* encrypting a 64-bit block with help of a 56-bit key. It consists in applying an initial permutation to the plaintext, then applying 16 consecutive Feistel rounds (the final swap being omitted), and finally applying the inverse of the initial permutation. The Feistel cipher structure [102] is guaranteed to be reversible (or, in other words, one can use the same process, up to the subkeys order, to encrypt and to decrypt data). Furthermore, one can notice that the function f does not need to be a bijection. The concept of Feistel cipher is formally described in Def. 2.2.1 and Fig. 2.3 illustrates a Feistel round.

Definition 2.2.1 (Feistel Cipher). *A* Feistel cipher Γ *is an iterated cipher mapping a plaintext of* $n = 2t$ *bits, denoted* $x_{\mathsf{l}}^{(0)}||x_{\mathsf{r}}^{(0)}$*, with* t*-bit blocks* $x_{\mathsf{l}}^{(0)}$*, and* $x_{\mathsf{r}}^{(0)}$ *to a ciphertext* $x_{\mathsf{l}}^{(r)}||x_{\mathsf{r}}^{(r)}$ *through an* r*-round process, where* $r \geq 1$.

[5] See §2.2.3 for more details about AES.

For $0 \leq i < r$*, round* i *maps* $x_{\mathsf{l}}^{(i)}||x_{\mathsf{r}}^{(i)} \mapsto x_{\mathsf{l}}^{(i+1)}||x_{\mathsf{r}}^{(i+1)}$ *according to*

$$\Gamma : \left\{ \begin{array}{rcl} x_{\mathsf{l}}^{(i+1)} & = & x_{\mathsf{r}}^{(i)} \\ x_{\mathsf{r}}^{(i+1)} & = & x_{\mathsf{l}}^{(i)} \diamond \mathsf{f}(x_{\mathsf{r}}^{(i)}, k^{(i+1)}) \end{array} \right.$$

where each subkey $k^{(i)}, 1 \leq i \leq \ell$ *is derived from the key* k *and* $\diamond$ *is a group law on* $\{0,1\}^t$*.*

The overall structure of DES is illustrated in Fig. 2.4, the right part of the figure being the *key-schedule algorithm*, while the left part is the Feistel scheme.

A DES key is often expressed as a 64-bit string, where the least significant (i.e. leftmost) bit of each byte is ignored and used as parity check bit to ensure that the key is error-free; the process of selecting these bits is performed by the transformation PC1 (Fig. 2.6), which eliminates the superfluous bits and permutes the remaining ones. After this operation, a different 48-bit subkey is generated for each of the 16 rounds of DES in the following manner: first the 56-bit key is divided into two 28-bit registers. Then, the halves are rotated to the left by two positions for all rounds but the rounds number 1, 2, 9, and 16, which are rotated to the left by a single position. After being rotated, 48 out of the 56 bits are selected and permuted by a compression function PC2 (Fig. 2.7). Because of the rotation, a different subset of key bits is used in each subkey. Actually, each bit is used in approximately 14 of the 16 rounds, but not all bits are used exactly the same number of times.

The initial transformation IP (Fig. 2.5) and its inverse IP^{-1} are straightforward bit permutations on 64-bit strings; actually, they does not possess any cryptographic meaning as they are key-independent GF(2)-linear operations.

The round function of DES is illustrated in Fig. 2.8 (note that the flow of the figure goes from the right to the left). It takes a 32-bit input x and outputs a 32-bit value y. First, a transformation called *expansion-permutation* EP expands the 32-bit input to 48 bits; this operation duplicates and permutes certain bits. Then, the 48-bit round subkey is combined with the output of EP with an exclusive-or operation (this operation being denoted K in Fig. 2.8). The result feeds then the substitution part of the cipher.

The substitution stage is composed of eight different non-linear transformations (see Fig. 2.9) mapping 6-bit values to 4-bit ones which are usually called "*S-boxes*". Hence, the 48-bit are split into eight 6-bits blocks. Each separate block is operated on by a separate S-box. An S-box is defined as a table of four rows and sixteens columns. The leftmost and the rightmost bit of the 6-bit input are combined to compute an index selecting the row, and the four inner bits specify the index of the column. For instance, the input `100100` to S-box S_5 gives `10` as row index and `0010` as column index, which results in the output `0001`.

Finally, the end of DES's round function consists of a straight bit permutation P which maps each output bit of the substitution stage to an output position, which means that no bit is used twice and no bit is ignored.

Design Criteria of DES

As outlined in the previous section, the non-published design criteria of DES have been the subject of much debates during the 80s and 90s. According to [67], Brown noticed in her PhD thesis [48] that

> "It has been stated [173] that there were 12 (possibly 13) criteria used, which resulted in about 1000 suitable S-boxes, of which the implementors chose 8."

Brickel *et al.* [47] mentioned that the only source of for specific design principles seem to be responses from the NSA to a study of the DES made by Lexar Corp. In these comments, the NSA labeled the following points as "design criteria" for the S-boxes: (1) no S-box is a linear or affine function of the input, (2) changing one input bit to an S-box results in changing at least two output bits, and (3) $\mathsf{S}(x)$ and $\mathsf{S}(x \oplus \mathtt{001100})$ must differ in at least two bits; the following points were labeled by the NSA as "caused by design criteria": (1) $\mathsf{S}(x) \neq \mathsf{S}(\mathtt{11}ab\mathtt{00})$ for any choice of a and b, and (2) the S-boxes were chosen to minimize the difference between the number of 1's and 0's in any S-box output when any single input is held constant.

After the invention of differential cryptanalysis by Biham and Shamir (see §2.3.3), Coppersmith revealed [60] the criteria used in the S-box design: (1) each S-box should have six bits of input and four bits of output, (2) no output bit of an S-box should be too close to a linear function of the input bits, (3) each row of an S-box should contain all possible outputs, (4) if two inputs to an S-box differ in exactly one bit, their outputs must differ by at least two bits, (5) if two inputs to an S-box differ exactly in the middle two bits, their outputs must differ by at least two bits, (6) if two inputs to an S-box differ in their first two bits and agree on their last two, the two outputs must differ, and finally, (7) for any nonzero 6-bit difference between inputs, no more than eight of the thirty-two pairs of inputs exhibiting that difference may result in the same output difference.

In an invited talk at the Crypto'2000 conference, Coppersmith mentioned some other design criteria as well: (1) when the two outer bits are fixed, the rest is a permutation on four bits, (2) $\Delta_{\text{in}} = \mathtt{0x00}xy\mathtt{00} \implies \Delta_{\text{out}} \neq 0$, where Δ_{in} and Δ_{out} denote the input and output differences (relatively to $\oplus$) and finally, (3) the implementation of an S-box should use at most 47 logical gates.

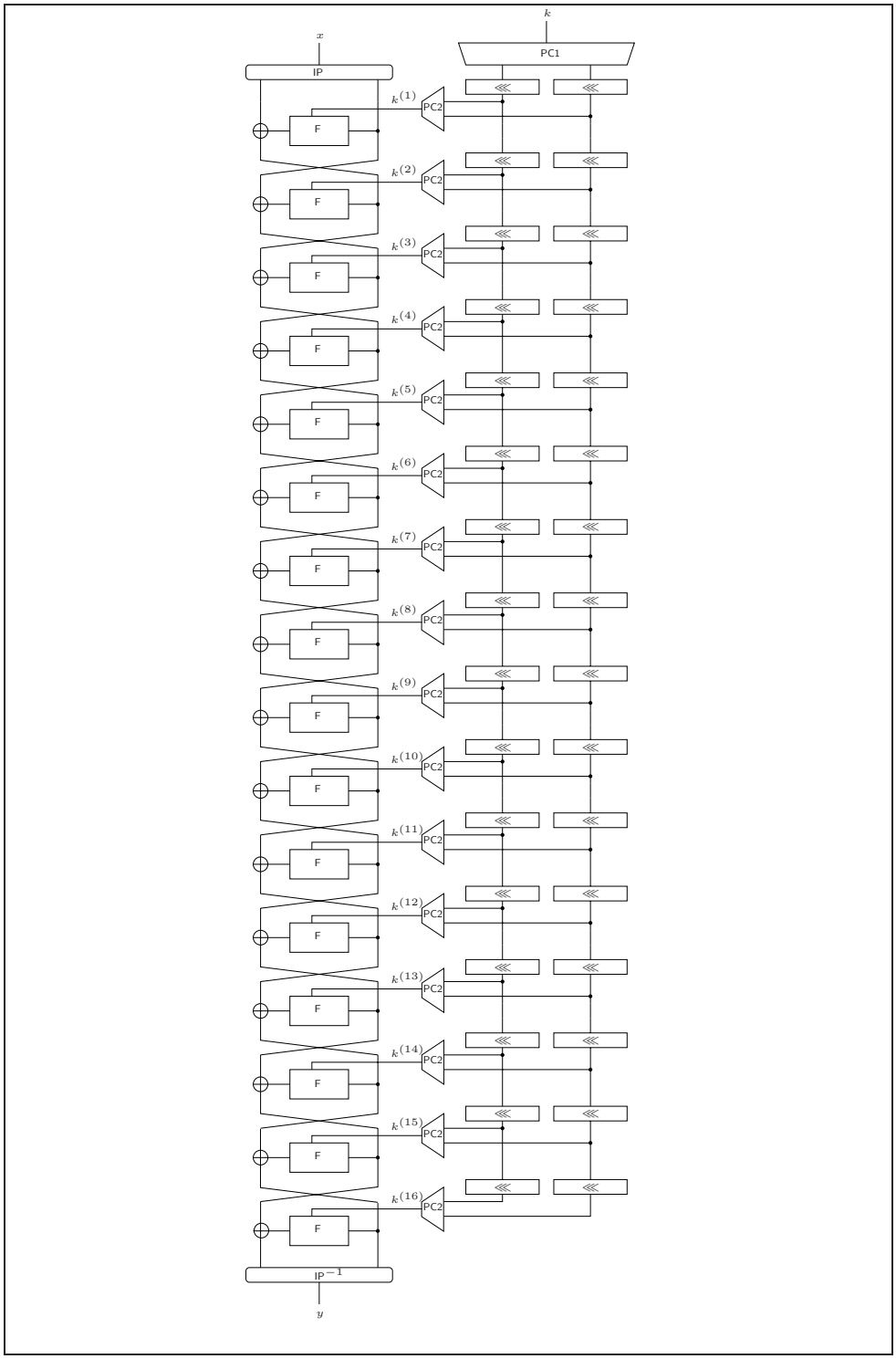

Figure 2.4: DES High-Level Scheme and Key-Schedule Algorithm

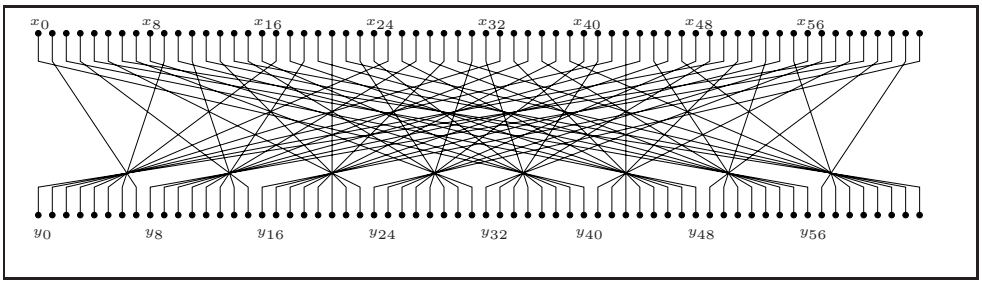

Figure 2.5: IP transformation

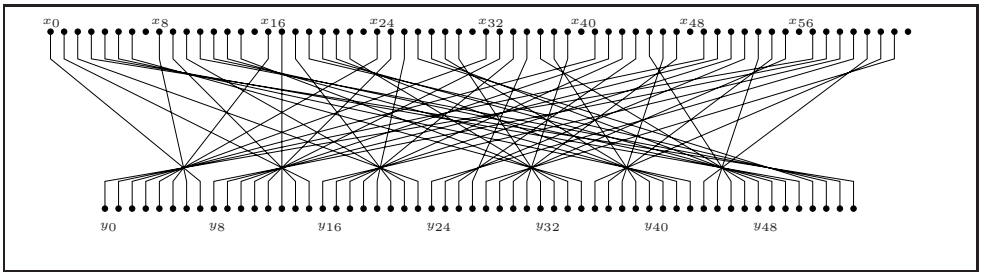

Figure 2.6: PC1 transformation

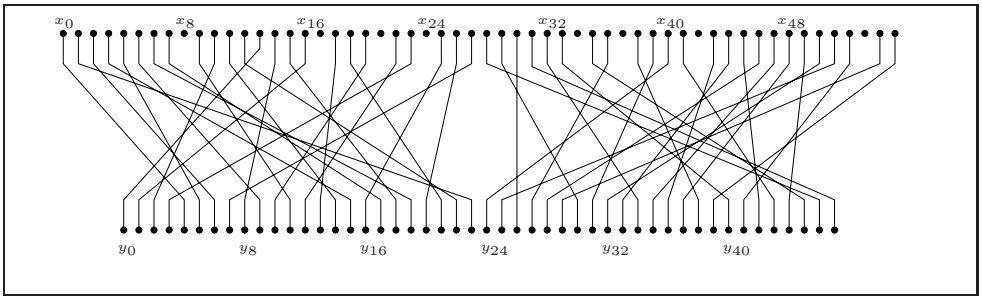

Figure 2.7: PC2 transformation

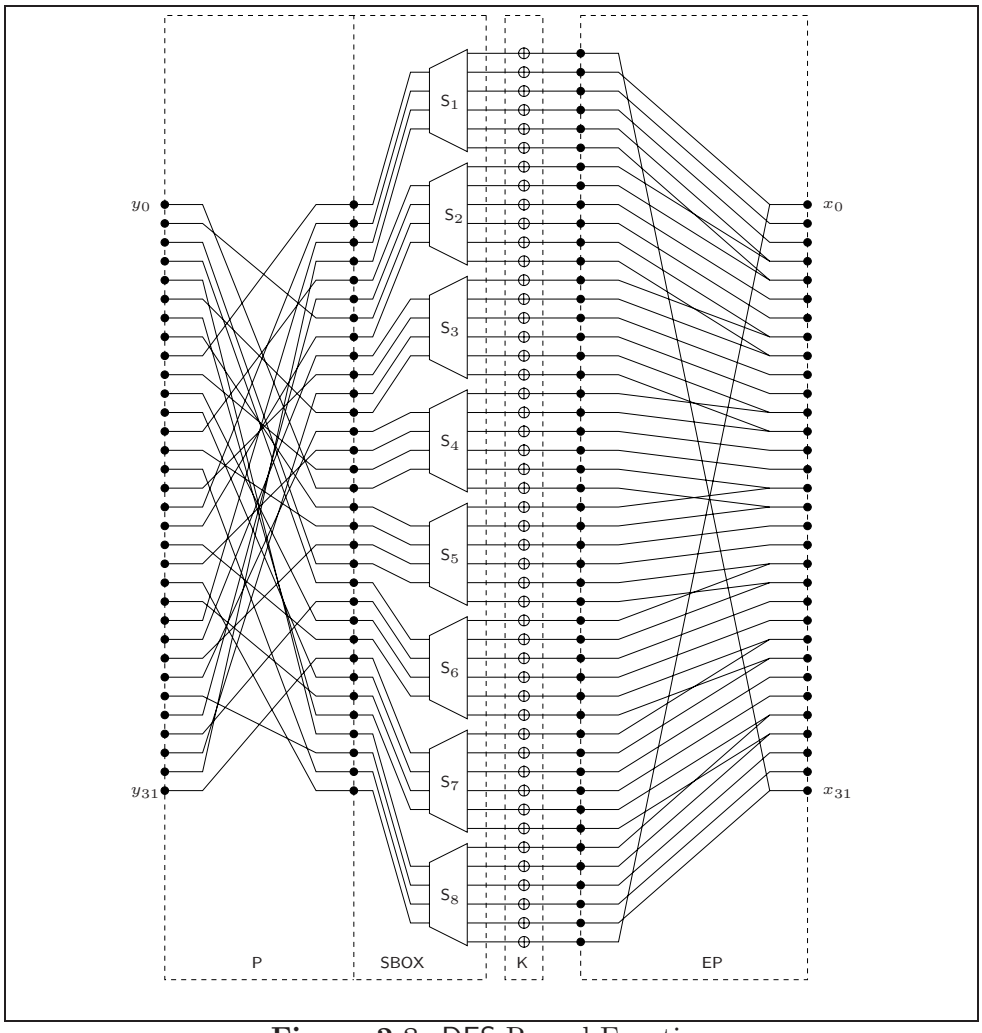

Figure 2.8: DES Round Function

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

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

S_3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
1	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
2	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
3	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C

S_4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
1	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
2	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
3	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E

S_5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
1	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
2	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	E
3	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3

S_6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
1	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
2	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
3	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D

S_7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
1	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
2	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
3	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C

S_8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	D	2	8	4	6	F	B	1	A	9	3	E	5	0	C	7
1	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
2	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
3	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

Figure 2.9: DES S-boxes

Cryptanalysis of DES

More than any other block cipher, and since it was one of the first commercially developed block cipher with open specifications, DES has motivated a large amount of cryptanalytical efforts: differential cryptanalysis, linear cryptanalysis, and Davies' attack are generic attacks invented for the purpose of breaking DES (see §2.3.3 for a description of these attacks).

The main weakness of DES is its short key length which allows nowadays to break it practically using an exhaustive key search; for instance, a dedicated machine has been built in 1998 by the Electronic Frontier Foundation to demonstrate the vulnerability of such a small key length. More information is given in §2.3.2. Hellman's time-memory tradeoff (see [129] and §2.3.2) can break DES using about 2^{38} cells of memory and 2^{38} operations after a single precomputation of 2^{56} operations.

Biham and Shamir's differential cryptanalysis [31–33] breaks DES faster than an exhaustive search if 2^{47} chosen plaintexts are available[6]. Another attack breaking DES faster than exhaustive search is an improvement of Davies' attack [83], by Biham and Biryukov [24, 25]; this known-plaintext attack requires 2^{50} known plaintext-ciphertext pairs.

In 1993, Matsui demonstrated [202, 203] that linear cryptanalysis can break DES as well, provided 2^{43} known plaintext-ciphertext pairs are available. Based on the same principles, Shimoyama and Kaneko [297] replaced linear approximations by probabilistic quadratic relations to slightly reduce the data complexity. Matsui's attack was transformed later by Knudsen and Matthiassen [166] in a chosen-plaintext attack, hence sightly reducing the required amount of data as well. We describe and discuss experimental results on the linear cryptanalysis of DES in §3.2.5, (see [144] as well) and we will demonstrate how to optimally use all the information furnished by Matsui's probabilistic linear approximations (see §3.2.4, page 80 and [149]).

2.2.2 IDEA

IDEA was one of the first proposed alternatives to DES. It was designed by Lai and Massey [179, 182] and is actually a "tweak" of PES [181] needed to counter differential cryptanalysis. IDEA has withstood a large amount of cryptanalytical effort (its popularity is due to the fact that it was chosen as the block cipher in the first versions of the software *Pretty Good Privacy (PGP)* [109] by Zimmerman) and all kinds of cryptanalytical attacks surprisingly well until now. Its strength is certainly due to an elegant and simple design approach which consists in mixing three algebraically incompatible group operations, namely the addition of vectors over $\mathrm{GF}(2)^{16}$, denoted "$\oplus$", the addition of integers over $\mathbb{Z}_{2^{16}}$, denoted "$\boxplus$", and the multiplication in $\mathrm{GF}\left(2^{16}+1\right)^*$, denoted "$\odot$".

[6] Note that this attack still works if the data are coming from up to 2^{33} different keys.

Description of IDEA

IDEA encrypts 64-bit data blocks under a 128-bit key; it consists of eight identical rounds and a final half-round (a key addition layer similar to those in a full round). Fig. 2.10 illustrates the computational flow of one round. Round r transforms a 64-bit input represented as a vector of four 16-bit words to an output vector of the same size:

$$(x_1^{(r)}, x_2^{(r)}, x_3^{(r)}, x_4^{(r)}) \mapsto (y_1^{(r)}, y_2^{(r)}, y_3^{(r)}, y_4^{(r)})$$

This process is parametered by six 16-bit subkeys denoted $k_i^{(r)}$, with $1 \leq i \leq 6$, which are derived from the master 128-bit key by means of the key-schedule algorithm. One evaluates the three IDEA algebraic operations as follows: $\oplus$ is a simple exclusive-or operation, $\boxplus$ is the addition modulo 2^{16} and $\odot$ is the common multiplication modulo $2^{16}+1$ (where 0 is considered as the number 2^{16}). First, two intermediate values $\alpha^{(r)}$ and $\beta^{(r)}$ are computed as follows:

$$\begin{aligned} \alpha^{(r)} &= \left(x_1^{(r)} \odot k_1^{(r)}\right) \oplus \left(x_3^{(r)} \boxplus k_3^{(r)}\right) \\ \beta^{(r)} &= \left(x_2^{(r)} \boxplus k_2^{(r)}\right) \oplus \left(x_4^{(r)} \odot k_4^{(r)}\right) \end{aligned}$$

These two values form the input of the *multiplication-addition box (MA-box)* which provides two outputs $\gamma^{(r)}$ and $\delta^{(r)}$:

$$\begin{aligned} \delta^{(r)} &= \left(\left(\alpha^{(r)} \odot k_5^{(r)}\right) \boxplus \beta^{(r)}\right) \odot k_6^{(r)} \\ \gamma^{(r)} &= \left(\alpha^{(r)} \odot k_5^{(r)}\right) \boxplus \delta^{(r)} \end{aligned}$$

Finally, the output of the round r is given by

$$\begin{aligned} y_1^{(r)} &= \left(x_1^{(r)} \odot k_1^{(r)}\right) \oplus \delta^{(r)} \quad , \qquad y_2^{(r)} = \left(x_2^{(r)} \boxplus k_2^{(r)}\right) \oplus \gamma^{(r)} \\ y_3^{(r)} &= \left(x_3^{(r)} \boxplus k_3^{(r)}\right) \oplus \delta^{(r)} \quad , \qquad y_4^{(r)} = \left(x_4^{(r)} \odot k_4^{(r)}\right) \oplus \gamma^{(r)} \end{aligned}$$

A *half-round* is defined to be the key-addition layer; we denote its output $(c_1^{(r)}, c_2^{(r)}, c_3^{(r)}, c_4^{(r)})$.

The key-schedule of IDEA allows to derive fifty-two 16-bit subkeys out of the 128-bit key k. Its description is straightforward; first, order the subkeys as

$$k_1^{(1)}, \ldots, k_6^{(1)}, k_1^{(2)}, \ldots, k_6^{(2)}, \ldots, k_1^{(9)}, \ldots, k_4^{(9)}$$

partition k into eight 16-bit blocks, and assign these blocks directly to the first eight subkeys. Then, do the following until all remaining subkeys are assigned: rotate k left 25 bits, partition the result, and assign these blocks to the next eight subkeys. In Fig. 2.11, we give explicitely the value of the subkeys (where $k_{[0\ldots15]}$ means the bits 0 to 15 (inclusive) of k, $k_{[117\ldots4]}$ means the bits 117-127 and 0-4 of k, and where the leftmost bit of k is numbered with 0).

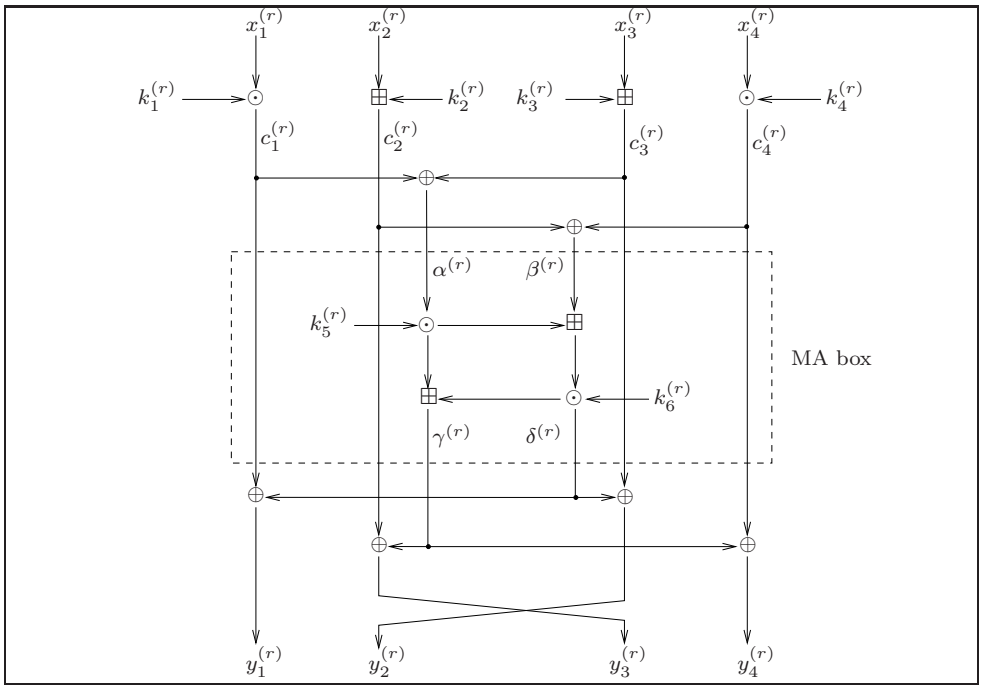

Figure 2.10: Round r of IDEA

Round r	$k_1^{(r)}$	$k_2^{(r)}$	$k_3^{(r)}$	$k_4^{(r)}$	$k_5^{(r)}$	$k_6^{(r)}$
1	$k_{[0\ldots15]}$	$k_{[16\ldots31]}$	$k_{[32\ldots47]}$	$k_{[48\ldots63]}$	$k_{[64\ldots79]}$	$k_{[80\ldots95]}$
2	$k_{[96\ldots111]}$	$k_{[112\ldots127]}$	$k_{[25\ldots40]}$	$k_{[41\ldots56]}$	$k_{[57\ldots72]}$	$k_{[73\ldots88]}$
3	$k_{[89\ldots104]}$	$k_{[105\ldots120]}$	$k_{[121\ldots8]}$	$k_{[9\ldots24]}$	$k_{[50\ldots65]}$	$k_{[66\ldots81]}$
4	$k_{[82\ldots97]}$	$k_{[98\ldots113]}$	$k_{[114\ldots1]}$	$k_{[2\ldots17]}$	$k_{[18\ldots33]}$	$k_{[34\ldots49]}$
5	$k_{[75\ldots90]}$	$k_{[91\ldots106]}$	$k_{[107\ldots122]}$	$k_{[123\ldots10]}$	$k_{[11\ldots26]}$	$k_{[27\ldots42]}$
6	$k_{[43\ldots58]}$	$k_{[59\ldots74]}$	$k_{[100\ldots115]}$	$k_{[116\ldots3]}$	$k_{[4\ldots19]}$	$k_{[20\ldots35]}$
7	$k_{[36\ldots51]}$	$k_{[52\ldots67]}$	$k_{[68\ldots83]}$	$k_{[84\ldots99]}$	$k_{[125\ldots12]}$	$k_{[13\ldots28]}$
8	$k_{[29\ldots44]}$	$k_{[45\ldots60]}$	$k_{[61\ldots76]}$	$k_{[77\ldots92]}$	$k_{[93\ldots108]}$	$k_{[109\ldots124]}$
8.5	$k_{[22\ldots37]}$	$k_{[38\ldots53]}$	$k_{[54\ldots69]}$	$k_{[70\ldots85]}$		

Figure 2.11: Complete Key-Schedule of IDEA

Cryptanalysis of IDEA

The cryptanalysis process of IDEA has been a rather lengthy process (see Fig. 2.12 for a recapitulation). To the best of our knowledge, Meier [219] was the first one to publish an attack based on differential cryptanalysis against up to 2.5 rounds running faster than an exhaustive search. Then, Borst *et al.* [45] presented a differential-linear attack against 3 rounds and a truncated differential attack on 3.5 rounds; Biham *et al.* [27] managed to break 4.5 rounds using impossible differentials. Motivated by a paper of Nakahara *et al.* [238] explaining how to break 2.5 rounds using an integral attack, Demirci [84] was able to break up to 4 rounds; one year later, these results were extended [85] using meet-in-the middle techniques to break up to 5 rounds slightly faster than an exhaustive search. Very recently, Nakahara *et al.* [239] devised known-plaintext attacks against reduced-round versions of IDEA using ideas of Demirci as well as an (unpublished) observation of Biryukov. Other papers [35, 77, 128] present attacks against the full version of IDEA, but these attacks work only for a negligible fraction of the keys. Additionally, we will present in §3.4 some efficient linear-like and square-like attacks on reduced-round versions of IDEA.

2.2.3 Advanced Encryption Standard (AES)

As mentioned above, due to the improvement of sciences and technology, DES is no longer appropriate for securing electronic communications. The NIST decided to launch in 1997 a new standardization process, known as *Advanced Encryption Standard*. This process, on a competitive basis, was completely open: anyone (i.e. non-American citizen and companies as well) was invited to submit a candidate algorithm and to send public comments on the other proposals. Fifteen candidates were accepted from all over the world in 1998: CAST256, Crypton, DEAL, DFC, E2, Frog, Hasty Pudding, LOKI 97, Magenta, Mars, RC6, Rijndael, SAFER+, Serpent, and Twofish. Based on extensive public comments, this set of candidates was reduced to 5 finalists (Rijndael, Mars, Serpent, Twofish and RC6) in 1999. In October 2000, Rijndael was selected to become the AES. The algorithm has been designed by two Belgian cryptographers, Daemen and Rijmen. A book [81] explaining the algorithm, its features and the rationales behind its design has been published.

Description of AES

AES processes 128-bit data blocks under a 128-, 192- or 256-bit key. Its design consists in writing the 128-bit plaintext as a 4×4 square matrix of bytes (this principle was first proposed in Square). The encryption process is made

Rounds	Data	Time	Attack type	Ref.	Note
2	2^{10} CP	2^{42}	differential	[219]	Memory: 2^{32}
2	23 CP	2^{64}	square-like	[84]	
2.5	2^{10} CP	2^{106}	differential	[219]	Memory: 2^{96}
2.5	2^{10} CP	2^{32}	differential	[77]	For one key out of 2^{77}
2.5	2^{18} CP	2^{58}	square	[238]	
2.5	2^{32} CP	2^{59}	square	[238]	
2.5	2^{48} CP	2^{79}	square	[238]	
2.5	2 CP	2^{37}	square	[238]	Under 2^{16} rel. keys
2.5	55 CP	2^{81}	square-like	[84]	
2.5	90 KP	2^{90}	linear-like	[239]	
3	2^{29} CP	2^{44}	differential-linear	[45]	
3	71 CP	2^{71}	square-like	[84]	
3	2^{33} CP	2^{64}	collision	[85]	Memory: 2^{64}
3.5	2^{56} CP	2^{67}	truncated diff.	[45]	
3.5	$2^{38.5}$ CP	2^{53}	impossible diff.	[27]	Memory: 2^{48}
3.5	2^{34} CP	2^{82}	square-like	[84]	
3.5	2^{24} CP	2^{73}	collision	[85]	
3.5	103 CP	2^{103}	square-like	[84]	
3.5	112 KP	2^{112}	linear-like	[239]	
4	2^{37} CP	2^{70}	impossible diff.	[27]	Memory: 2^{48}
4	2^{34} CP	2^{114}	square-like	[84]	
4	2^{24} CP	2^{89}	collision	[85]	Memory: 2^{64}
4	121 KP	2^{114}	linear-like	[239]	
4.5	2^{64} CP	2^{112}	impossible diff.	[27]	
4.5	2^{24} CP	2^{121}	collision	[85]	Memory: 2^{64}
5	2^{24} CP	2^{126}	collision	[85]	Memory: 2^{64}

Figure 2.12: Attacks against IDEA

of 10, 12 or 14 rounds, for keys of 128-, 192- and 256-bit, respectively. It begins with the addition (through a XOR operation) of a round subkey to the input data. Then, a fixed number of rounds of a substitution-permutation network (SPN) is applied.

Each round consists in four operations: first, a bijection, SubBytes, is applied to the matrix, consisting basically of 16 byte-wise substitutions defined by the same substitution table. SubBytes is a bijective mapping offering optimal characteristics regarding non-linearity (and thus an optimal resistance towards linear and differential cryptanalysis, see §2.3.3). It consists in the inversion operation in GF $\left(2^8\right)$ where elements of the field are polynomials of degree at most 7 on GF (2) modulo the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$; it is followed by an affine transformation over GF $(2)^8$. Second, a circular shift called ShiftRows of all rows of the matrix is applied: row number $i, 1 \leq i \leq 4$ is rotated to the right by i positions. Third, a linear transformation, defined by a 4×4 matrix over GF $\left(2^8\right)$, called MixColumns, is applied on each column (the last round omitting the MixColumns and ShiftRows steps); more precisely, this operation considers a column as the coefficients of a polynomial over GF $\left(2^8\right) / (x^8 + x^4 + x^3 + x + 1)$ and the columns are multiplied by the polynomial $\texttt{0x03} \cdot x^3 + \texttt{0x01} \cdot x^2 + \texttt{0x01} \cdot x + \texttt{0x02}$ modulo $x^4 + 1$. MixColumns possesses optimal diffusion properties: if ℓ input bytes are modified, at least $5 - \ell$ output bytes will be modified. Finally, a 128-bit round subkey is XORed.

The decryption process consists simply in applying the respective inverse operations in the reverse order, while the key-schedule algorithm description may be found in [81].

Cryptanalysis of AES

At the time of writing this thesis, there is no (undisputed) attack against the full version of AES breaking it faster than an exhaustive key search.

The designers of AES claim [81] that no 4-round differential characteristic hold with probability greater than 2^{-150}, and no 4-round linear characteristic exist with a bias greater than 2^{-75}, as an analysis of the propagation of activity patterns leads to the conclusion that any linear or differential characteristic activate at least 25 S-boxes. Note however that such numbers have to be taken with a grain of salt, since there must exist at least a differential or a linear hull which hold with a probability of 2^{-128} or a bias equal to 2^{-64}, respectively; this demonstrates that the cumulative effect of characteristics is huge in AES. However, more recently, Keliher *et al.* [155–157] obtained (with help of about 200'000 hours of computations) an upper bound on the maximum average linear hull probability (and on the maximum expected differential probability) of 2^{-92} for 9 rounds or more. Using more theoretical arguments, Park *et al.* obtained [258] an upper bound of 2^{-112} and of 2^{-105} on the maximum differential probability, and on the maximum linear

hull probability for 4 rounds of AES, respectively.

Biham and Keller [30] have proposed an impossible differential cryptanalysis on 5 rounds which was later extended to 6 rounds in [59] and to 7 rounds, with help of a weakness of the key-schedule algorithm, by Phan [264].

Several types of integral attacks [103, 115, 194] have been applied with success against reduced-round versions of AES, the best having been proposed by Gilbert and Minier [115]. Furthermore, as AES has a simple algebraic structure, several papers have demonstrated puzzling properties, which may eventually be exploited in the future in an attack [11,16,70,104,108,235].

2.2.4 Modes of Operation

A block cipher handles messages as data blocks. Usually, the size of data to be encrypted is larger than the block size n of the cipher under consideration. Thus, these data have to be divided into a sequence of message blocks having the same length n. Different *modes of operation*, based on an underlying block cipher, have been designed. These modes of operation usually provide a set of desirable properties to the ciphertext blocks, such as adding randomness to a block cipher, padding plaintexts to an arbitrary length, control of error propagation, or even the transformation of a block cipher in a stream cipher.

Modes of operation are often discussed according to following criteria:

- *Error expansion*: two types of error may occur during the transmission of a message processed by a mode of operation. Either one can encounter *slip errors* which means that either bits are deleted or inserted in an arbitrary manner, or *bit-flipping errors* where some bits are flipped. Depending on the mode in usage, these errors may cause no error expansion, i.e. the errors are limited to the same block, finite error expansion, i.e. the error propagates to a limited amount of block, or even an infinite error propagation.

- *Plaintext redundancy*: this criterion is related to whether the mode under consideration allows the plaintext probability distribution to propagate to the ciphertext, either in a partial or in a full way.

- *Random access*: certain modes of operation allow to read or to modify a block at random position without the need to read (or modify) other blocks.

- *Parallel processing*: certain modes of operation have the property to be processed in a parallel way; this means that every block can be computed without the need to know other blocks. Such modes allow very-high performance implementations.

- *Randomness addition*: certain modes require some additional randomness, like an initialization vector, independently of the plaintext or of the key.

In the next section, we recall briefly the five most important modes of operation, namely ECB, CBC, OFB, CFB, and CTR.

Electronic Codebook Mode

The *Electronic Codebook Mode (ECB)*, standardized in [137, 243], is the most straightforward (and the most insecure) way to employ a block cipher for encrypting (or decrypting) a sequence of messages. It just consists in encrypting them one another separately.

Definition 2.2.2 (Electronic Codebook Mode). *Let $x = x_1 || \dots || x_m$ be a message made of m blocks of size n, and let $\mathsf{e}(.)$ be a block cipher having a block length equal to $n = |x_i|$. The encryption of x in the* Electronic Codebook Mode (ECB) *under a key k is defined as*

$$y_i = \mathsf{e}_k(x_i) \qquad 1 \leq i \leq m$$

while the decryption operation is

$$x_i = \mathsf{d}_k(y_i) \qquad 1 \leq i \leq m$$

where $\mathsf{d}(.)$ is the inverse of $\mathsf{e}(.)$.

The ECB mode is deterministic, meaning that two encryptions under the same key of the same message x will result twice in the same ciphertexts y. Furthermore, the ECB mode may leak some statistical information about the plaintext, if the latter is very redundant: for instance, equal plaintexts result in equal ciphertexts. As each block is treated separately and does not depend on other blocks, the ECB mode can be implemented in a highly parallel way; bit-flipping errors cause harm only on the block in which they occur, while slip errors cause infinite error expansion. In practice, mainly due to its deterministic nature, the use of ECB mode is only recommended to encrypt a single block of data.

Cipher Block Chaining Mode

The *Cipher Block Chaining Mode (CBC)*, standardized in [137,243], defines a dependence between a block to be encrypted and the previous encrypted block and make use of an *Initialization Vector (IV)*.

Definition 2.2.3 (Cipher Block Chaining Model). *Let $x = x_1 || \dots || x_m$ be a message made of m blocks of size n, and let $\mathsf{e}(.)$ be a block cipher having*

a block length equal to $n = |x_i|$. *The encryption of* x *in the* Cipher Block Chaining Mode (CBC) *under a key* k *is defined as*

$$\begin{aligned} y_0 &= \text{IV} \\ y_i &= \mathsf{e}_k(x_i \oplus y_{i-1}) \qquad 1 \leq i \leq m \end{aligned}$$

while the decryption operation is

$$\begin{aligned} y_0 &= \text{IV} \\ x_i &= \mathsf{d}_k(y_i) \oplus y_{i-1} \qquad 1 \leq i \leq m \end{aligned}$$

where $\mathsf{d}(.)$ *is the inverse of* $\mathsf{e}(.)$.

The security of CBC mode has been studied by Bellare *et al.* [19]. They conclude that, in order that the CBC mode resists to chosen-plaintext attacks, it is required that the underlying block cipher can be modeled as a pseudo-random permutation, and that the initialization vector is secret and chosen uniformly at random for each new message and key. However, CBC is insecure in the chosen-ciphertext attack model; furthermore, it suffers from a confidentiality limitation [162], as when two ciphertext blocks are equal, one can recover the XOR of the subsequent plaintexts. This phenomenon is likely to occur when one encrypts message whose size is about equal to the *square root* of the block size of the underlying cipher.

The CBC mode resists quite well to bit-flipping errors, which cause only the current and the next block to be incorrectly decrypted. However, slip errors result in infinite error expansion.

Output Feedback Mode

The *Output Feedback Mode (OFB)* [137, 243] features feeding the successive output blocks from the underlying block cipher back to it. These feedback bits form a sequence of bits which are used as a key for the Vernam cipher. This mode requires an initialization vector, which however does not need to be secret.

Definition 2.2.4 (Output Feedback Mode). *Let* $x = x_1||\ldots||x_m$ *be a message made of* m *blocks of size* n, *and let* $\mathsf{e}(.)$ *be a block cipher having a block length equal to* $n = |x_i|$. *The encryption of* x *in the* Output Feedback Mode (OFB) *under a key* k *is defined as*

$$\begin{aligned} \kappa_0 &= \text{IV} \\ \kappa_i &= \mathsf{e}_k(\kappa_{i-1}) \qquad 1 \leq i \leq m \\ y_i &= x_i \oplus \kappa_i \end{aligned}$$

while the decryption operation is

$$\begin{aligned} \kappa_0 &= \text{IV} \\ \kappa_i &= \mathsf{e}_k(\kappa_{i-1}) \qquad 1 \leq i \leq m \\ x_i &= y_i \oplus \kappa_i \end{aligned}$$

where $\mathsf{d}(.)$ *is the inverse of* $\mathsf{e}(.)$.

Note that we have defined here the version of OFB which produces a keystream by steps which have the same size as the block size of the underlying algorithm. This mode can be generalized to (and is usually defined as) *r-bit OFB* which generates a keystream by steps of $r \leq n$ bits. The OFB mode is relatively resistant to bit-flipping errors, as errors are limited to the same segment and there is no error expansion. Slip errors cause an infinite error propagation.

Cipher Feedback Mode

The *Cipher Feedback Mode (CFB)* [137, 243] features feeding the successive cipher segments which are output from the mode back as input to the underlying block cipher. Like CBC and OFB, CFB requires an initialization vector.

Definition 2.2.5 (Cipher Feedback Mode). *Let* $x = x_1||\ldots||x_m$ *be a message made of* m *blocks of size* n*, and let* $\mathsf{e}(.)$ *be a block cipher having a block length equal to* $n = |x_i|$*. The encryption of* x *in the* Cipher Block Chaining Mode (CBC) *under a key* k *is defined as*

$$\begin{aligned} y_0 &= \text{IV} \\ y_i &= x_i \oplus \mathsf{e}_k(y_{i-1}) \end{aligned}$$

while the decryption operation is

$$\begin{aligned} y_0 &= \text{IV} \\ x_i &= y_i \oplus \mathsf{e}_k(y_{i-1}) \end{aligned}$$

where $\mathsf{d}(.)$ *is the inverse of* $\mathsf{e}(.)$.

Similarly to the OFB mode, the CFB mode can be generalized to (and is usually defined as) *r-bit CFB* which generates a keystream by steps of $r \leq n$ bits.

Counter Mode

The *Counter Mode (CTR)* has been proposed by Diffie and Hellman [94].

Definition 2.2.6 (Counter Mode). *Let $x = x_1||\dots||x_m$ be a message made of m blocks of size n, and let $\mathsf{e}(.)$ be a block cipher having a block length equal to $n = |x_i|$. The encryption of x in the* Counter Mode (CTR) *under a key k is defined as*

$$\begin{aligned} \kappa_0 &= \text{IV} \\ \kappa_i &= \mathsf{e}_k(\kappa_i + 1 \bmod 2^n) \\ y_i &= x_i \oplus \kappa_i \end{aligned}$$

while the decryption operation is

$$\begin{aligned} \kappa_0 &= \text{IV} \\ \kappa_i &= \mathsf{e}_k(\kappa_i + 1 \bmod 2^n) \\ x_i &= y_i \oplus \kappa_i \end{aligned}$$

where $\mathsf{d}(.)$ is the inverse of $\mathsf{e}(.)$.

The security of the CTR mode can be proven for adversaries able to mount a chosen-plaintext attack if the underlying block cipher can be modeled as a pseudo-random permutation and that the initialization counter is non-repeating for each new message and for each re-synchronization under the same key.

2.3 Attacks Against Block Ciphers

2.3.1 Attack Models and Terminology

The concept of *attack* against a block cipher includes several notions: its *outcome*, the *threat model* in which it can be realized, its *type*, and its *complexity*. In this part, we discuss each of these notions in a detailed way.

Outcome of an Attack

According to the type of information recovered during an attack, Knudsen [162] classified the possible outcomes of an attack in a hierarchical way, the first described outcome being the most favorable for an adversary.

- *Total break*: an adversary recovers (or reconstructs) the secret key k.
- *Global deduction*: an adversary finds an algorithm functionnaly equivalent to $\mathsf{e}_k(.)$ or $\mathsf{d}_k(.)$ without knowing the actual value of the key k. A global deduction is possible when a block cipher contains "block structures", i.e. if certain subsets of the ciphertext are independent of certain subsets of the plaintext; in this case, independently of the

key length, such a block cipher is vulnerable to a global deduction in a known-plaintext attack. Another possibility of global deduction is that an attack is able to recover the round subkeys but not the key, in the case where the key-schedule algorithm is designed to be a (secure) one-way function, for instance.

- *Instance (local) deduction*: an adversary finds the plaintext (or ciphertext) of an intercepted ciphertext (or plaintext) which (s)he did not obtain from the legitimate sender. An instance deduction may be as dangerous as a total break if the number of likely plaintexts (or ciphertexts) is small.

- *Distinguishing attack*: an adversary is able to tell whether the attacked block cipher is a permutation chosen uniformly at random from the set of all permutations or one of the 2^ℓ permutations specified by the secret key. Distinguishing attacks are often considered as the least serious threat in practice; however, they often can be transformed into a key-recovery attack which may lead to a total break (or a global deduction). Distinguishing attack are the corner stone of the Luby-Rackoff security approach which will be discussed in §2.4.2.

Additionally to these four outcomes, Knudsen [162] defines an "information deduction attack": an adversary gains some information (in the sense of Shannon's information theory [294]) about the secret key, the plaintexts or the ciphertexts (s)he did not had *a priori*. For instance, an adversary, after an attack, may learn that some plaintexts are distributed according to ASCII English text or that the key comes from a subset of the set of all possible keys. In practice, an information deduction may be a serious problem if the plaintext (or ciphertext) possesses a low entropy. In contrast to the four above definitions, the latter one seems difficult to work with in a mathematical, formal sense, since, by definition, Shannon's information is a measure between probability distributions which is independent of the fact that an enemy knows the value of that measure or not.

Threat Model of a Block Cipher

A usual model of threats classification consists in building a hierarchy of attacks according to the adversary's potential (or assumed) capabilities, ranked from the least powerful attacks to the most powerful ones.

- *Ciphertext-only attack*: in this kind of passive attack, an adversary tries to deduce some information about the key (or about the plaintext) by only *observing* a certain amount of ciphertexts. Usually, one assumes some known property about the plaintext or the key; for instance the adversary may know that the plaintext consists of ASCII

characters. Block ciphers vulnerable to ciphertext-only attacks are considered to be completely broken.

- *Known-plaintext attack*: in this case, one assumes that an adversary knows a certain amount of plaintext-ciphertext pairs; the goal of this kind of passive attack consists in finding the key. Typically, one encounters known-plaintext attacks in scenarios where an adversary can observe encrypted version of well-known data, like the data exchanged during the setup phase of a protocol, for instance. A typical example of known-plaintext attack is the *linear cryptanalysis* (see §2.3.3, page 45 and §3).

- *Non-adaptive chosen-plaintext attack*: when performing this kind of *active* attack, the adversary is able to choose plaintexts and obtains the corresponding ciphertexts; the plaintext must not depend on the obtained ciphertexts: one can view them as submitted in a parallel way. Subsequently, the adversary uses any information deduced in order to recover either the key, or plaintext(s) corresponding to previously unseen ciphertext(s). One may encounter such a scenario for instance when a tamper-proof module implementing a block cipher with a fixed key falls in the hands of an adversary and where it is not possible to recover directly the key (e.g. with physical means). A typical example of a non-adaptive chosen-plaintext attack is the *differential cryptanalysis* (see §2.3.3, page 40).

- *Adaptive chosen-plaintext attack*: such an attack is a chosen-plaintext attack wherein the choice of the plaintext may depend on the ciphertext received from previous requests.

- *(Non-) Adaptive chosen-ciphertext attack*: one assumes that the adversary is able to decrypt arbitrary ciphertexts (in a adaptive way or not) and obtain the corresponding plaintext with the objective of recovering the key or to encrypt a (not previously observed) plaintext. In the context of block ciphers, this kind of attack is very similar to chosen-plaintext attacks.

- *Combined chosen-plaintext and chosen-ciphertext attack*: this extremely powerful type of adaptive attacks assumes that the adversary can encrypt and decrypt arbitrary texts as (s)he desires. A typical example of such an attack is Wagner's *boomerang attack* (see §2.3.3, page 44).

- *Related-key attack*: this model of attack assumes that the adversary knows (or can choose) additionally some mathematical relation between the *keys* used for encryption and decryption, but not their val-

ues. This kind of attack may be practical when a block cipher is used as a primitive for a hash function, for instance.

Even if a given attack may not be considered to be a practical threat against a block cipher, because it lives in a too strong threat model, such an attack may be viewed as a kind of *certificational attack* against the block cipher. To quote Winternitz and Hellman who have introduced this concept in [331] together with the one of related-key attack, "*if two systems are of approximately equal complexity, and if one system succumbs to a chosen-key cryptanalytic attack [...] while the second does not, the second is to be preferred*".

Type of Attack

Depending on the knowledge of the internal details of a block cipher, and depending on the information gathered when analyzing implementation details, one can classify attacks in an alternative way as follows:

- *Black-box attacks*: these are generic attacks which treat the block cipher as a black box taking plaintexts and a key in input and outputting ciphertext; as such attacks do not depend on any internal details of the algorithm, one can apply them against every block cipher, and their complexity depends only on parameters like the key length ℓ and the block length n of the block ciphers under consideration. Known black-box attacks, like exhaustive key search or generic time-memory tradeoffs, are discussed in further details in §2.3.2.

- *Shortcut attacks*: on the contrary to black-box attacks, shortcut attacks are based on a mathematical analysis of the internal details of the block ciphers under consideration. The most powerful known attacks are of course shortcut attacks; we discuss them from §2.3.3 to §2.3.6.

- *Side-Channel attacks*: inevitably, the fate of a block cipher is to be implemented either in software or in hardware. Side-channel attacks exploits various physical phenomenons generated by these implementations. For instance, timing attacks can be applied when the execution time of an algorithm is dependent of the data and/or the key value. Although proposed for the first time to attack public-key algorithms [170], timing attacks were demonstrated [57, 122, 172] against block ciphers as well. Another way to exploit weaknesses of physical implementations of block ciphers is to measure the power consumption of tamper-proof hardware [171] and infer some information about the key from these measures. Finally, *fault analysis* exploits the following idea, proposed by Biham and Shamir [34]: one can induce faults during the execution of a block cipher by using any physical mean (like

power glitches), to study the effects of these faults on the algorithm behavior, and to extract some information about the key.

Parameters of an Attack

Attacks against block ciphers, besides the threat model under consideration, depend on several parameters. Even if it is not always obvious to compare in practice the power of two attacks, since their success depends heavily on the context, these parameters are however inevitable central points in such a comparison.

- The *time complexity* of an attack is the amount of computational processing required to perform this attack successfully. The computational unit is often chosen such that one can compare the attack to an exhaustive key search. Furthermore, one sometimes divides this time complexity in two parts, namely the *pre-computation* and *post-computation* times, if the attack needs to perform computations off-line, i.e. before and after data are required.

- By *data complexity*, one means the number of data (like ciphertexts, known-plaintext, chosen-plaintext, ...) required to perform an attack in the threat model under consideration. Since these data must be obtained from the key holder, this has a direct influence on the communication complexity.

- The *success probability* of an attack measures the frequency at which the attack is successful when repeated a certain number of times in a (statistically) independent way.

- The *memory complexity* measures the amount of memory units necessary to store either pre-computed data necessary to perform the attack, or (possibly parts of) the data obtained in the threat model under consideration.

Usually, the *complexity of an attack* is often chosen to be the largest figure among the time, data and memory complexities, although there is no general consensus about this fact in the academic literature.

To decide whether a block cipher is broken or not is often more a matter of taste than something which is clearly defined. Usually, an attack is considered to be successful, and the attacked block cipher is considered to be *broken* if the time complexity is significantly smaller than 2^ℓ evaluations of the block ciphers, where ℓ denotes its key size; a block cipher is considered to be *partially broken* if some of the plaintext bits can be discovered in time faster than an exhaustive search.

In a similar way, for a fixed key, a block cipher can be completely characterized if the encryption of all possible 2^n plaintexts is available, where n

is its block size; this puts an upper bound on the data complexity. Quoting NESSIE's final security report [269],

> "A block cipher is considered secure if no attack requires both time and data complexity significantly less than 2^ℓ and 2^n, respectively."

However, it is worth mentioning that the above definition implies that all block ciphers could be considered as broken, since Hellman's time-memory tradeoff (see [129] and §2.3.2) needs only in the order of $2^{\frac{2\ell}{3}}$ operations and $2^{\frac{2\ell}{3}}$ space to succeed (after a 2^ℓ pre-computation, which can however be amortized on many attacks) ! We prefer adopt the following definition.

Definition 2.3.1 (Secure Block Cipher). *A block cipher is considered secure if no attack requires both time and data complexity significantly less than the respective complexities required by any generic attack.*

As it is often difficult (or it may even be impossible) to exhibit an attack against the full version of an iterative block cipher, another common mean to assess its security consists in taking into account the maximal number of rounds for which an attack is known; at least, this can give some feeling about the security margin of such a block cipher. An illustrative example is Fig. 2.12, page 24, which summarizes the currently best known attacks (without the attacks described in S3.4) on various reduced-round versions of IDEA.

2.3.2 Black-Box Attacks

As outlined in the previous part, some of the known attacks against block ciphers can be applied in a "black-box" fashion, i.e. without attacking the internal structure of the block cipher. These attacks include the *exhaustive key search*, attacks dedicated to *multiple encryption*, *key-collision attacks*, and *time-memory tradeoffs*.

Exhaustive Key Search

One of the simplest way to attack a block cipher consists in trying one key after the other until the right one is found. Typically, for a block cipher e having a key size ℓ and a block size n, and provided that a very small number of known plaintext-ciphertext pairs (slightly more than $\lceil \frac{\ell}{n} \rceil$, actually, provided the cipher is not badly flawed) encrypted under the same key k, one can recover this key k by exhaustive search; this operation has a worst case time complexity equal to 2^ℓ evaluations of e and an average time complexity of $2^{\ell-1}$. If the underlying plaintext space is known to contain some redundancy (for instance, it is ASCII text), then one can even consider a ciphertext-only exhaustive search.

One of the interesting properties of an exhaustive key search is that it is an attack which can be executed in parallel on many processors or dedicated machines, each one testing disjoint subsets of the key space. The success probability of an exhaustive key search is equal to the fraction of the key space searched: if one searches one tenth of the key space, then one has roughly a 10% probability to succeed. In other terms, a fixed key size ℓ defines an *upper bound* on the security of a block cipher. Thus, for any secure block cipher, ℓ should be large enough to thwart exhaustive key search attacks.

The minimal necessary key size to offer a comfortable security margin has been the subject of much debates in the literature. A prominent example is the key size of DES (see §2.2.1), which possesses a relatively small key size (56 bit) and very early, concerns about the resistance of DES to an exhaustive key search were raised by researchers. In 1977, Diffie and Hellman [93] estimate that a US$ 20'000'000 worth dedicated machine could be built to attack one DES key. In an unpublished[7] paper presented during Eurocrypt'87 [88], Desmedt and Quisquater propose the design of a dedicated machine based on a chip described in [133] which could be able to break one million DES keys in 4 weeks, or about 3000 keys each hour in average. They estimate the cost of this machine to US$ 3'000'000. In 1990, Garon and Outerbridge [110] estimate that a machine using special-purpose chips costing US$ 1'000'000 could break a DES key in nine days using 1995's technology and in 43 hours in 2000. At Crypto'92, Eberle [96] proposes the design of a chip able to encrypt 1 Gbit/s and estimates that an exhaustive attack against DES would take less than 16 days using a US$ 1'000'000 worth machine. At the same conference, Wayner [325] estimates that a very simple parallel architecture using a content-addressable memory can be used to build a US$ 30'000'000 worth machine able to recover a DES key in one day in average. In 1993, Wiener [328, 329] presented the detailed design of a dedicated machine costing about US$ 1'000'000 able to find a DES key in an average of 3.5 hours; in 1997, he revised [330] his estimation down to a time of 35 minutes.

Interestingly, the small key size of DES motivated on the one hand the construction of a dedicated machine in the spirit of the ones described in the works cited above, and on the other hand, several projects based on distributed software and using the idle time of many volunteer computers spread over the Internet, for the sole purpose of demonstrating the weakness of such a small key length. In 1998, the Electronic Frontier Foundation (EFF) [97] announced the construction of a US$ 210'000 worth[8] exhaustive-search machine dedicated to attack DES, the design of the machine being

[7] This paper appears as Chapter 9 in [98].

[8] It is worth noticing that Standaert [302] estimates that in 2004, a FPGA-based machine costing US$ 12'000 should break a DES key in 3 days.

Challenge	Team	Date	Time
DES I	Rocke Verser *et al.*	June 1997	96d
DES II-1	`distributed.net`	February 1998	41d 17h 18m
DES II-2	EFF DES cracker	July 1998	2d 8h
DES III	EFF + `distributed.net`	January 1999	22h 15m

Figure 2.13: RSA Security DES Challenges

Year	Key Size (bits)
1982	56
1990	63
2000	70
2010	78
2020	86
2030	93

Figure 2.14: Equivalent symmetric key lengths according to Lenstra and Verheul [185]

thoroughly described in [98]; this machine is able to recover a DES key in about 109 hours in average. The company RSA Security [281] has proposed a sequence of challenges consisting in recovering a DES key. Fig. 2.13 summarizes the results obtained, the most impressive result being a DES key recovered in less than 24 hours. The `distributed.net` [95] project recovered a RC5 64-bit key on July 14th, 2002 after 1757 days of work and is attempting, at the time of writing, to recover a RC5 72-bit key.

An attempt by a group of cryptographers to address the key size problem in a rigorous way is [40]. In this document, published in 1996, they recommend a minimal key length of 75 bits for providing adequate protection against the most serious threats. For protecting information in a secure way during the next 20 years, they estimate that a minimal key length of 90 bits should be sufficient.

Lenstra and Verheul give in [185] a rigorous estimation of the trend in the future of the computing power needed to break symmetric keys by exhaustive search, taking into account the financial aspects as well as an estimation of the technology power increase. One of their conclusions is that breaking a 78-bit symmetric key in 2010 will be as "hard" as it was to break a 56-bit DES key in 1982. Fig. 2.14 summarizes some of their estimations.

Nowadays, modern block ciphers usually allow keys of 128, 192, 256 bits or even more; this allows to thwart easily an exhaustive search of the key space. Note that in all of these considerations, we have assumed a classical model of computation. Actually, a quantum computer, implementing Grover's

Number of precomputed ciphertexts	2^{28}	2^{32}	2^{40}	2^{48}	2^{56}
Expected number of keys found	1	2^{8}	2^{24}	2^{40}	2^{56}

Figure 2.15: Complexity of a key-collision attack against DES

algorithm [120], is able to find a given item in an unsorted list of size n in time $O(\sqrt{n})$ instead of $O(n)$. Actually, Grover's algorithm can be shown to be optimal. Thus, a quantum computer will have to execute in the order of 2^{64} (quantum) operations to break a 128-bit key by exhaustive search. However, the current state of the technology is far to be sufficient to build a quantum computer.

Key-Collision Attack

Another interesting black-box attack, called *key-collision attack*, has been described by Biham [20]. The principles are very simple and are based on the birthday paradox. We assume that we are attacking a block cipher with a key length ℓ and a block length $n < \ell$; furthermore, we assume that a known plaintext p is encrypted under many distinct keys. We can build a table of p encrypted under $2^{\frac{\ell}{2}}$ random distinct keys and, by the birthday paradox, we expect that, after observing about $2^{\frac{\ell}{2}}$ ciphertexts, the probability to recover at least one key becomes non-negligible. Fig. 2.15 gives the relation between the number of precomputed encryptions and the expected number of keys found in the case of a key-collision attack against DES.

Multiple Encryption

A few black-box attacks against block ciphers using multiple encryption (in the sense of Def. 2.1.3 and Def. 2.1.4) have been discovered so far.

In 1977, Diffie and Hellman [93] noted a *meet-in-the-middle* (MITM) attack on *double encryption* (see Def. 2.1.3), suggesting that one should using at least three-fold encryption in a multiple-encryption scenario. This attack works as follows: we assume first that we have a few known plaintext-ciphertext pairs (p_i, c_i) encrypted with the same (unknown) key at disposal, i.e. such that $c_i = \mathsf{e}_{k_2}(\mathsf{e}_{k_1}(p_i))$. Given (p_1, c_1), we compute $m_s = \mathsf{e}_s(p_1)$ under all possible 2^ℓ possible key values s and we store the pairs (m_s, s), indexed on the m_s's in a table. Then, we decipher c_1 under all 2^ℓ possible key values t and for each pair $m_t = \mathsf{d}_t(c_1)$, we look for an equality $m_t = m_s$ in the first table. Each solution clearly identifies a possible solution key pair (s, t). Using the few other known plaintext-ciphertext pairs, one can finally isolate the right key (k_1, k_2). This attacks breaks a block cipher with a ℓ-bit key used in double-encryption mode in time $O(2^\ell)$ and uses $O(2^\ell)$ memory

cells. This attack has been generalized to a cascade of $c \geq 2$ block ciphers by Even and Goldreich in [100].

According to [221, page 272], Merkle describes in his PhD thesis [222] (see [224] as well) a chosen-plaintext attack against two-key triple encryption (see Def. 2.1.4) which needs $O(2^\ell)$ block cipher evaluations, $O(2^\ell)$ memory cells and $O(2^\ell)$ chosen plaintexts, where ℓ is the key size. The idea is to reduce a two-key triple encryption to a double encryption in the following way: for all possible 2^ℓ possible key values s, compute $p_s = \mathsf{d}_s(0)$. Submit each resulting p_s as a chosen-plaintext to the encryption module and obtain the corresponding c_s. For each c_s, compute $x_s = \mathsf{d}_s(c_s)$. This value represents an intermediate value x after the second of the three encryption stages, as it is the case for the p_s values. Then, sort the values p_s and x_t in a table. The cases where $p_s = x_t$ propose a candidate key pair (s, t). One finally isolates the solution among the candidates using a few more known pairs.

This attack was turned in 1990 into a known-plaintext attack by van Oorschot and Wiener [309]. Basically, given q known plaintext-ciphertext pairs, their attack requires $O(q)$ memory cells and a time complexity equal to $O\left(2^{\ell - \log_2 q}\right)$, where ℓ is the total key length. In [310], the same authors consider ways to decrease the (costly) memory needs while increasing the time complexity; Lucks studies in [193] the inverse strategy.

Time-Memory Tradeoffs

Hellman [129] proposed a time-memory tradeoff which can be applied to an exhaustive key search. The idea here consists in precomputing some information and to use it in order to speed up key searches. Practically, this attack is able to recover an ℓ-bit secret key after $O\left(2^{\frac{2\ell}{3}}\right)$ encryption operations by using $O\left(2^{\frac{2\ell}{3}}\right)$ words of memory, whose content is initialized in a unique precomputation step needing 2^ℓ encryptions. According to Denning [87, page 100], Rivest proposed to use distinguished points to reduce the search time. Some extensions of these works (both theoretical and practical) may be found in [5, 105, 106, 175, 304].

The efficiency of Hellman's time-memory tradeoff has been improved by Oechslin [256] (although the asymptotic time complexity remains the same): in his paper, he proposes a new way to pre-calculate the data which allows to reduce significantly the number of operations during the attack itself.

2.3.3 Statistical Attacks

In this part, we present briefly a sequence of attacks exploiting undesirable probabilistic properties of block ciphers.

Differential Cryptanalysis and Variants

A very important attack against block ciphers is the *differential cryptanalysis*. It was proposed[9] by Biham and Shamir in 1990 to attack DES [31–33] and then applied against several ciphers by many cryptographers with success. In this part, we present the ideas behind differential cryptanalysis together with some known variations and generalizations.

Differential cryptanalysis is a method which looks at ciphertext pairs whose corresponding plaintexts have particular *differences*. More precisely, let $\diamond$ denote the group law on the groups of bit strings used to combine the key with the text in a block cipher and where $(p')^{-1}$ is the inverse element of p' in the group. Then a difference Δ is defined as $\Delta p = p \diamond (p')^{-1}$ where $(p')^{-1}$ is the inverse element of p' in the group. Indeed, $\diamond$-based differences are invariant under the $\diamond$ operation used to mix a (fixed) key, since

$$(p \diamond k) \diamond \left(p' \diamond k\right)^{-1} = p \diamond k \diamond k^{-1} \diamond (p')^{-1} = p \diamond (p')^{-1}$$

for all p, p' and (fixed) key k. Closely related concepts are the *differential probability* and the *maximal differential probability*.

Definition 2.3.2 (Differential Probability). *The* differential probability *of a function f_k relatively to a pair of differences $\Omega = (a, b)$ and a group operation $\diamond$, denoted $\mathrm{DP}^{\mathsf{f}}(a, b)$, is defined as*

$$\mathrm{DP}^{\mathsf{f}}(a, b) = \Pr_X \left[\mathsf{f}_k \left(X \diamond a\right) = \mathsf{f}_k(X) \diamond b\right]$$

and the maximal differential probability is defined to be

$$\mathrm{DP}^{\mathsf{f}}_{\max} = \max_{a \neq 0, b} \mathrm{DP}^{\mathsf{f}}(a, b)$$

As this group operation is an exclusive-or (XOR), denoted $\oplus$, in a majority of block ciphers, we will restrict ourselves to this operation when speaking about differential cryptanalysis.

One can concatenate two difference pairs $\Omega_1 = \left(\omega_1^{(1)}, \omega_2^{(1)}\right)$ and $\Omega_2 = \left(\omega_1^{(2)}, \omega_2^{(2)}\right)$ if $\omega_2^{(1)} = \omega_1^{(2)}$. In this case, one may often approximate the differential probability of the resulting differential pair by the product of the differential probabilities of Ω_1 and of Ω_2, respectively.

Let $\mathsf{f} = \mathsf{f}^{(r)} \circ \ldots \circ \mathsf{f}^{(1)}$ be an iterated block cipher made of r rounds; we are interested in pairs (x, x') such that $\mathsf{f}\left(x \oplus a\right) = \mathsf{f}\left(x'\right) \oplus b$ with *high probability* for some fixed $a \neq 0$ and b. The sequence of differences induced by the pair $\Omega = (a, b)$ in the r rounds of f is called [33] a *differential characteristic*.

[9] In fact, differential cryptanalysis was known before 1990. Coppersmith, which was a member of the DES [242] design team at IBM in the early 70's, revealed that his team was aware of this attack back in 1974 and that they designed DES S-boxes and the permutations in order to optimally defeat it.

Definition 2.3.3 (r-rounds differential characteristic). *An* r-rounds differential characteristic Ω *is a sequence of differences defined as an* $(r+1)$-*tuple* $(\omega_0, \omega_1, \ldots, \omega_r)$ *where* $\Delta x = \omega_0$ *and* $\Delta y_i = \omega_i$ *for* $1 \leq i \leq r$.

A heuristic assumption used to mount a differential cryptanalysis states that the propagation of differences is independent of the (unknown) subkey values and that the associated differential probability is the same than the average over all possible subkey values. This assumption is known as *hypothesis of stochastic equivalence* and was formally stated by Lai [179, 182].

Assumption 2.3.1 (Hypothesis of stochastic equivalence). *For virtually all high-probability* r*-round differentials* (ω_0, ω_r),

$$\Pr_{X_1 X_2 \mathbf{K}}\left[\Delta Y^{(r)} = \omega_r \middle| \Delta X = \omega_0\right] = \Pr_{X_1 X_2 | \mathbf{K}}\left[\Delta Y^{(r)} = \omega_r \middle| \Delta X = \omega_0, \mathbf{K} = \mathbf{k}\right]$$

holds for a substantial fraction of the subkey values $\mathbf{k} = (k_1, \ldots, k_r)$.

In other words, one assumes that the variance of $\mathrm{DP}^{\mathsf{f}}(\omega_0, \omega_r)$ (taken over the key distribution) is very small for every "interesting" pair (ω_0, ω_r). Thus, an essential measure is the one of *expected differential probability.*

Definition 2.3.4 (Expected Differential Probability). *The* expected differential probability *of a function* f_k *relatively to a pair of differences* $\Omega = (a, b)$ *is defined by*

$$\mathrm{EDP}^{\mathsf{f}}(a, b) = \mathrm{E}\left[\mathrm{DP}^{\mathsf{f}_K}(a, b)\right]$$

where the expectation is taken over the key distribution.

In order to study in a formal way differential cryptanalysis, which is essentially heuristic, we need a good probabilistic model of the block cipher f under review. Lai's PhD thesis [179] (see [182] as well) introduces the concept of *Markov cipher.*

Definition 2.3.5 (Markov Cipher). *An iterated block cipher with round function* $y = \mathsf{f}_k(x)$ *is a* Markov cipher *if there is a group operation* $\diamond$ *for defining differences such that, for all choices of* $a \neq 0$, $b \neq 0$, *and of* x, *the equality*

$$\Pr_K\left[\mathsf{f}_K(x \diamond a) = \mathsf{f}_K(x) \diamond b\right] = \mathrm{EDP}^{\mathsf{f}}(a, b)$$

holds.

In an iterated r-round Markov cipher, the differential probability of a differential characteristic $\Omega = (\omega_0, \omega_1, \ldots, \omega_r)$ is independent of the actual input

of a given round, and provided the round subkeys are statistically independent and uniformly distributed, the individual round probabilities are independent and may be computed as

$$\mathrm{E}\left[\Pr_{X_1 X_2}\left[\Delta Y^{(i)} = \omega_i, 1 \leq i \leq r \middle| \Delta X = \omega_0\right]\right] = \prod_{i=1}^{r} \mathrm{EDP}^{\mathsf{f}^{(i)}}(\omega_{i-1}, \omega_i).$$

where the expectation is taken over the key.

Conceptually, no real block cipher is a Markov cipher, since most modern design use a key-schedule algorithm to generate round subkeys out of the key. Thus, round subkeys are virtually always statistically dependent. Anyway, Biham and Shamir [32] observe that the product of the 1-round differential probabilities seems to result in a good approximation in the case of a differential cryptanalysis of DES.

As observed for the first time by Lai, Massey and Murphy [182], it may sometimes be useful to consider an r-rounds differential characteristic depending *only* of ΔX and $\Delta Y^{(r)}$: one calls such a "characteristic" a *differential*.

Definition 2.3.6 (r-rounds differential). *An* r-rounds differential *is defined as a pair of differences* (ω_0, ω_r) *where* $\Delta x = \omega_0$ *and* $\Delta y^{(r)} = \omega_r$.

In other words, the intermediate differences $\omega_1, \ldots, \omega_{r-1}$ are allowed to take any value and the cryptanalyst takes thus advantage of the *cumulative effect* of many differential characteristics. The probability of an r-round differential in a Markov cipher f is then equal to

$$\mathrm{DP}^{\mathsf{f}}(\omega_0, \omega_r) = \sum_{\omega_1} \cdots \sum_{\omega_{r-1}} \prod_{i=1}^{r} \Pr\left[\Delta y^{(i)} = \omega_i \middle| \Delta y^{(i-1)} = \omega_{i-1}\right]$$

with the convention that $\Delta y^{(0)} = \Delta x$.

Typically, the first step towards a successful differential cryptanalysis against a block cipher consists is finding a high-probability differential characteristic or differential. In order to be able to mount a *key-recovery attack* against an r-rounds, one usually exploits a characteristic active on a $r - s$ rounds, with s small (e.g. $s = 1$ or $s = 2$). The second step consists in "guessing" the k key bits relevant to the s first (or last, or a mixture of the two) rounds and linked to the differential characteristic, which allows us to in some sense to "peel" these s rounds off. By managing a counter for each of the 2^k (sub-) key candidates, one counts the number of times where a (sub-) key leads to the expected difference pair.

In their seminal paper, Biham and Shamir [33] introduce the concept of *signal-to-noise ratio*: it is the ratio, for a fixed subkey candidates, between the number of *right pairs* it generates, i.e. the pairs of plaintexts which lead to the expected output difference, and the average number of *wrong*

pairs taken over all (sub-) key candidates, i.e. pairs which don't lead to this precise difference. To isolate the correct (sub-) key, one needs obviously a sufficiently large number of right pairs. Biham and Shamir observed that high values of signal-to-noise ratio (i.e. significantly larger than one) lead to a small need in right pairs, while values equal or smaller than one lead to an unreasonably large number of needed right pairs.

Impossible Differentials A variant of differential cryptanalysis using so-called *impossible differentials* was proposed later by Biham, Biryukov and Shamir [26] against 31 rounds of Skipjack; it is a sieving attack which looks at differentials having a probability equal to zero (or unexpectedly low) to occur. If a pair is decrypted to such a difference under some key, then this key is certainly not the correct one and can be eliminated from the set of candidates. A typical technique (called "*miss-in-the-middle*") to construct an impossible differential is to combine two differentials holding with probability 1, but which cannot be simultaneously satisfied.

Borst, Knudsen and Rijmen [45] observed in an attack against a variant of IDEA that in the case where the signal-to-noise ratio is larger than one, the right (sub-)key value is among the most suggested while, when the signal-to-noise ratio is smaller than one, the right (sub-)key is the least suggested one. According to Nakahara [237], "*the overall consensus is that the case where the signal-to-noise ratio is equal to 1 does not allow to distinguish the right subkey from the wrong ones*". We will come back to this issue in §3.3.4, prove this assertion and show that the signal-to-noise ratio is indeed a quantity linked to the optimal distinguisher.

Higher-Order Differentials The concept of difference has been generalized by Lai in [180] and used to define an attack framework called "*higher-order differentials*" by Knudsen [161].

Definition 2.3.7 (Higher-order differential). *A* one-round differential of order i *is an* $(i+1)$*-tuple* $(\alpha_1, \ldots, \alpha_1, \beta)$ *such that*

$$\Delta^{(i)}_{\alpha_1,\ldots,\alpha_i} \mathsf{f}(x) = \beta$$

where f *is a function from an additive Abelian group* $\mathfrak{S}$ *to an additive Abelian group* $\mathfrak{T}$ *and*

$$\begin{aligned} \Delta_a \mathsf{f}(x) &= \quad \mathsf{f}(x+a) - \mathsf{f}(x) \\ \Delta^{(i)}_{a_1,\ldots,a_i} &= \Delta_{a_i} \left(\Delta^{(i-1)}_{a_1,\ldots,a_{i-1}} \mathsf{f}(x) \right) \end{aligned}$$

A higher-order differential attacks works similarly as a basic differential cryptanalysis, i.e. it exploits a high-probability higher-order difference pair. Higher-order differential attacks have then been applied with success by Moriai, Shimoyama and Kaneko [230] against CAST and against reduced-round versions of Misty1 and Kasumi (see [13, 56, 127, 306, 307]).

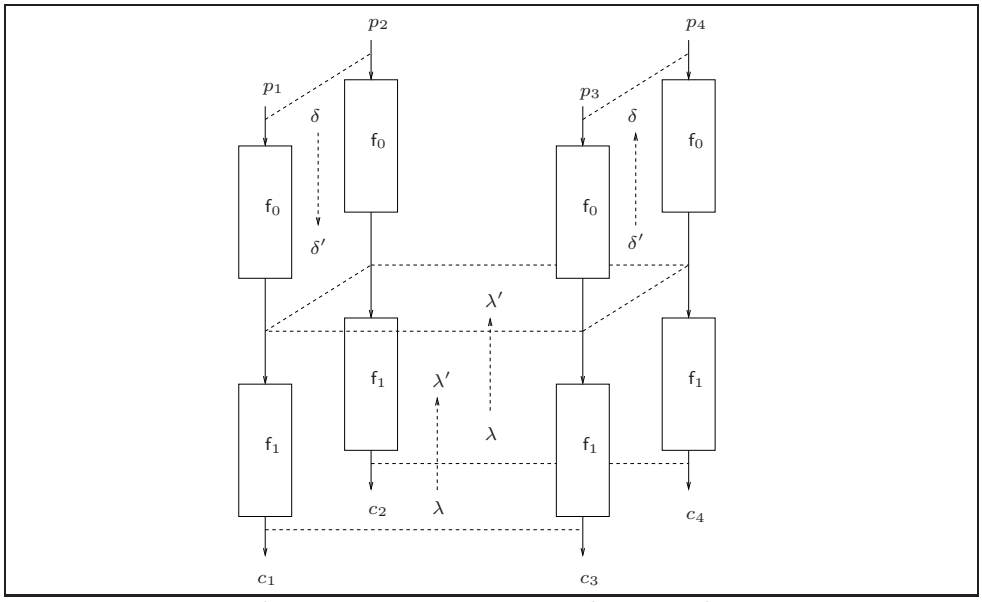

Figure 2.16: Boomerang distinguisher

Truncated Differentials Another variant of differential cryptanalysis is the so-called *truncated differential cryptanalysis* (see Def. 2.3.8) which have been proposed by Knudsen in [161]. In this paper, it is shown that it can be advantageous to predict *parts* of the differences after each round of a cipher. A truncated differential can be seen as a collection of common differentials.

Definition 2.3.8 (Truncated differential). *Let $\Omega = (\omega_0, \omega_r)$ be a r-round differential. If ω'_0 is a subsequence of ω_0 and ω'_r is a subsequence of ω_r, then $\Omega' = (\omega'_0, \omega'_r)$ is called a* r-round truncated differential.

Truncated differentials have been used in an attack [165] against 5 of the 6 rounds of SAFER K-64, to attack [45] 3.5 rounds of IDEA and Skipjack [117, 168, 272].

Boomerang attack A more adaptive differential-like attack is Wagner's *boomerang attack* [322]. Usually, a block cipher designer may argue that a blocker cipher is immune against differential cryptanalysis because he obtains an upper bound on the probability π of *any* differential characteristic and that one needs supposedly at least $\frac{1}{\pi}$ texts to break the cipher. A boomerang distinguisher allows the attacker to beat the $\frac{1}{\pi}$ bounds in certain cases.

This attack, in contrast to a classical differential attack, does not require that the full cipher is spanned by a single differentials; it exploits high probability differentials, possibly uncorrelated, but that *jointly* cover the full cipher. To perform this attack, an attacker needs two oracles, namely

an encryption and a decryption one. More formally, let f be a block cipher that can be written[10] as $\mathsf{f} = \mathsf{f}_1 \circ \mathsf{f}_0$. let us assume that there exists a differential characteristic (or a differential, or a truncated differential) (δ, δ') with respect to $\oplus$ which spans f_0 with a non-negligible probability $\pi_{(\delta,\delta')}$: then $\delta' = \mathsf{f}_0(p_1) \oplus \mathsf{f}_0(p_2)$. Let $c_1 = \mathsf{f}(p_1)$ and $c_2 = \mathsf{f}(p_2)$ and let us define $\lambda = c_1 \oplus c_3 = c_2 \oplus c_4$, where the differential characteristic (λ, λ') spans f_1 with a non-negligible probability $\pi_{(\lambda,\lambda')}$ (see Fig. 2.16 for an illustration); then,

$$\lambda' = \mathsf{f}_1^{-1}(c_1) \oplus \mathsf{f}_1^{-1}(c_3) = \mathsf{f}_1^{-1}(c_2) \oplus \mathsf{f}_1^{-1}(c_4)$$

The attack works as follows: choose a pair (p_1, p_2) such that $\delta = p_1 \oplus p_2$ and submit it to the encryption oracle to get the pair (c_1, c_2). Then submit $(c_1 \oplus \lambda, c_2 \oplus \lambda)$ to the decryption oracle and check that the corresponding plaintext pair (p_3, p_4) satisfies $\lambda = p_3 \oplus p_4$. Following Wagner's terminology, a *right quartet* is defined as a 4-tuple (p_1, p_2, c_3, c_4) for which all four differential patterns defined previously hold simultaneously. The probability π_{b} to observe a boomerang $\delta = p_3 \oplus p_4$ can be estimated as

$$\pi_{\mathrm{b}} = \pi^2_{(\delta,\delta')} \times \pi^2_{(\lambda,\lambda')}$$

Improvements of the boomerang attack are the *amplified boomerang attack* [158] of Kelsey, Kohno and Schneier and the *rectangle attack* of Biham, Dunkelman and Keller [28].

Linear Cryptanalysis and Variants

Another generic way to exploit non-linearities in block ciphers is the *linear cryptanalysis*. Based on earlier observations by Shamir [292] and attacks against FEAL of Gilbert, Tardy-Corfdir, Chassé, Matsui and Yamagishi [113, 206, 308], this known-plaintext attack was proposed by Matsui in [202] in a version dedicated to DES, which was refined later and implemented in [203].

The goal of a linear cryptanalysis is to find *unbalanced linear relations*: they are Boolean equations involving the sum in GF(2) of some plaintext, ciphertext and key bits which holds with a probability $\pi \neq \frac{1}{2}$. By linear relation, we mean an equation

$$\mathbf{a} \cdot x \oplus \mathbf{b} \cdot \mathsf{f}_k(x) = 0$$

where $\mathbf{a}$, $\mathbf{b}$ are vectors over GF (2) (and usually called *masks*), x and $\mathsf{f}_k(x)$ are seen as vectors over GF(2) and "$\cdot$" denotes the inner dot-product. In a similar way as in the case of linear cryptanalysis, we can define the concepts of *linear probability*.

[10] The parts do not need to have the same size.

Definition 2.3.9 (Linear probability). *The* linear probability *of a function* f *relatively to a pair of masks* $(\mathbf{a}, \mathbf{b})$ *of vectors over* $\mathrm{GF}(2)$*, denoted* $\mathrm{LP}^{\mathsf{f}}(\mathbf{a}, \mathbf{b})$*, is defined as*

$$\mathrm{LP}^{\mathsf{f}}(\mathbf{a}, \mathbf{b}) = \left(2 \Pr_X[\mathbf{a} \cdot X = \mathbf{b} \cdot \mathsf{f}_k(X)] - 1\right)^2$$

and the maximal linear probability *is defined to be*

$$\mathrm{LP}^{\mathsf{f}}_{\max} = \max_{\mathbf{a},\mathbf{b} \neq \mathbf{0}} \mathrm{LP}^{\mathsf{f}}(\mathbf{a}, \mathbf{b})$$

Similarly to the definition of expected differential probability, one can define the expected linear probability.

Definition 2.3.10 (Expected Linear Probability). *The* expected linear probability *of a function* f_k *relatively to a pair of masks* $(\mathbf{a}, \mathbf{b})$ *is defined by*

$$\mathrm{ELP}^{\mathsf{f}}(\mathbf{a}, \mathbf{b}) = \mathrm{E}\left[\mathrm{LP}^{\mathsf{f}_K}(\mathbf{a}, \mathbf{b})\right]$$

where the expectation is taken over the key distribution.

We can define the concepts of *r-rounds linear characteristic* in a similar way as for differential cryptanalysis.

Definition 2.3.11 (Linear characteristic). *Let* $\mathsf{f} = \mathsf{f}^{(r)} \circ \ldots \circ \mathsf{f}^{(1)}$ *be an iterated block cipher made of* r *rounds. An* r*-round* linear characteristic Λ *is a sequence of masks defined as an* $r+1$*-tuple* $(\lambda_0, \lambda_1, \ldots, \lambda_r)$ *for* $1 \leq i \leq r$*.*

The *linear characteristic probability* is then defined as follows.

Definition 2.3.12 (Linear Characteristic Probability). *The* linear characteristic probability *of an* r*-round iterated function* f_k *relatively to an* r*-rounds linear characteristic* $\Lambda = (\lambda_0, \lambda_1, \ldots, \lambda_r)$ *is defined as*

$$\mathrm{LCP}^{\mathsf{f}_k}(\Lambda) = \prod_{i=1}^{r} \mathrm{LP}^{\mathsf{f}_k^{(i)}}(\lambda_{i-1}, \lambda_i)$$

where $\mathsf{f}^{(i)}$ *denotes the* i*-th round function equipped with a round subkey. The* expected linear characteristic probability *is defined as*

$$\mathrm{ELCP}^{\mathsf{f}}(\Lambda) = \prod_{i=1}^{r} \mathrm{ELP}^{\mathsf{f}^{(i)}}(\lambda_{i-1}, \lambda_i)$$

Similar to the concept of differential in a differential cryptanalysis, Nyberg [250] has introduced the concept of *linear hulls*.

Definition 2.3.13 (Linear hull). *Let* $\mathsf{f} = \mathsf{f}^{(r)} \circ \ldots \circ \mathsf{f}^{(1)}$ *be an iterated block cipher made of* r *rounds. An* r*-round* linear hull $(\mathbf{a}, \mathbf{b})$ *is the set of all* r*-round linear characteristics of the form* $(\lambda_0, \ldots, \lambda_r)$ *where* $\lambda_0 = \mathbf{a}$ *and* $\lambda_r = \mathbf{b}$*.*

As in the case of a differential, the intermediate masks are allowed to take any value; thus, one takes advantage of the cumulative effect of many linear characteristics. The effect of linear hull explains why linear cryptanalysis' efficiency could be underestimated in certain situations.

Some generalizations have been proposed, without any significant practical improvements: linear cryptanalysis with multiple approximations [153], with non-linear approximations [167] or with quadratic expressions [297]. We will come back in §3 on several issues about linear cryptanalysis.

Differential-Linear Cryptanalysis

In certain precise situations, one can combine different attack techniques to form a new attack. A nice illustration is the *differential-linear cryptanalysis* proposed by Langford and Hellman [183]. In this attack, one uses a differential through a fraction of the cipher which is used to create a linear relation holding with probability 1. This linear relation is then concatenated with other linear relations on the remaining of the cipher. This technique was refined recently by Biham et al. [29] in order to create linear relations holding with a probability strictly less than 1.

χ^2 Attacks

This attack was proposed for the first time by Vaudenay [312] in the context of a statistical cryptanalysis of DES: in this paper, it is shown that one can obtain an attack slightly less powerful than a linear cryptanalysis, but *without knowing precisely what happens in the block cipher*. The idea consists in looking for relations of any kind which produce a significant deviation from what one should expect from a uniformly distributed permutation. A distinguisher is then built using a χ^2 statistical analysis. We will come back on the foundations of this attack in §3.

Non-Surjective Attacks

Davies' attack[11] [83] (which is probably the first statistical attack in the modern cryptanalysis era) against DES is a statistical attack which exploits

[11] Note that, although this attack was published in 1995, it has been already communicated by Davies in 1987 to some cryptographers. According to the PhD thesis of Gilbert [112], it may be due to the fact that DES was (and is still) broadly used in the banking system and that it was useless to drop the confidence in this algorithm.

the fact that the outputs from neighboring S-boxes are not uniformly distributed. This attack, which has been optimized in [24, 25] by Biham and Biryukov, can derive a DES secret key if 2^{50} known plaintext-ciphertext pairs are available. Although it can theoretically be applied to any Feistel cipher, this attack has merely historical interest: it was the third attack breaking DES faster than an exhaustive key search.

This attack framework was generalized in 1997 by Rijmen, Preneel, and De Win [276]. It applies to Feistel ciphers (see Fig. 2.3). More formally, let us consider an r-round Feistel cipher on n-bit strings, where $r \geq 4$ and n and r are even numbers. Plaintexts and ciphertexts consists in two halves x_{l} and x_{r} of size $\frac{n}{2}$ each. Round i of the Feistel scheme takes a n-bit input $(x_{\mathsf{l}}^{(i)}, x_{\mathsf{r}}^{(i)})$ and a round subkey $k^{(i)}$ and returns a n-bit output computed as follows:

$$\begin{cases} x_{\mathsf{l}}^{(i+1)} &= x_{\mathsf{r}}^{(i)} \\ x_{\mathsf{r}}^{(i+1)} &= x_{\mathsf{l}}^{(i)} \oplus \mathsf{f}\left(k^{(i)} \oplus x_{\mathsf{r}}^{(i)}\right) \end{cases}$$

Let us furthermore define the quantity

$$\sigma_r = \bigoplus_{i=1}^{r/2} \mathsf{f}\left(k^{(2i)} \oplus x_{\mathsf{r}}^{(2i-1)}\right)$$

We can note that $\sigma_r = x_{\mathsf{r}}^{(0)} \oplus x_{\mathsf{l}}^{(r)}$. If the function $\mathsf{f}^{(i)}$ is unbalanced, i.e. if it does not take all the outputs in its range equally often, or in other words, it is not surjective, then σ_r will be unbalanced as well, provided the subkeys are independent and uniformly distributed. If not all values of σ_r have the same probability to occur, then one can gather statistical information about the key.

2.3.4 Integral Attacks

Basically, differential cryptanalysis and variants thereof consist in studying the propagation of differences between pairs of plaintexts (or ciphertexts). The *integral cryptanalysis* [134, 169] extends this view by considering sums of (many) values. Older and less general variants of this attack are known as the *square attack*, proposed in the specifications of Square, *structural cryptanalysis* of Biryukov and Shamir [36], and the *saturation attack* of Lucks [195]. A central concept is the one of *integral*.

Definition 2.3.14 (Integral). *Let $\mathfrak{G}$ be a finite Abelian group of order k and let us consider the product group $\mathfrak{G}^n = \mathfrak{G} \times \ldots \times \mathfrak{G}$ equipped with the component-wise addition. Let $\mathcal{M}$ be a multiset of vectors. An* integral *over $\mathcal{M}$ is defined as*

$$\int \mathcal{M} = \sum_{v \in \mathcal{M}} v$$

where the summation is defined in terms of the group operation for $\mathfrak{G}^n$.

In an integral attack, one will typically try to predict the values of integrals after a certain number of rounds. One usually is interested in one of the three following cases: all i-th components are equal, all components are different or all i-th components sum to a (predicted in advance) constant.

2.3.5 Algebraic Attacks

In his seminal paper [295], Shannon stated that breaking of a block cipher should "*require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type*". Some preliminary attempts in this direction is the early work of Schaumüller-Bichl [282] in an application to DES, but without much practical success.

Interpolation Attack

A purely algebraic (and practical) way to break block ciphers is the *interpolation attack* proposed by Knudsen and Jakobsen [142]. It is based on the well-known Lagrange's formula: if $\mathfrak{F}$ is a field, the unique polynomial $\mathsf{p}(x) \in \mathfrak{F}[x]$ of degree at most $n-1$ such that $\mathsf{p}(x_i) = y_i$ for n pairs $(x_i, y_i) \in \mathfrak{F}^2$ is equal to

$$\mathsf{p}(x) = \sum_{i=1}^{n} y_i \prod_{\substack{1 \leq j \leq n \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

In an interpolation attack, one is interested in constructing polynomials using inputs and outputs of the block cipher. The attack's idea is that if the constructed polynomials have a small degree, only few plaintexts and the corresponding ciphertexts are necessary to solve for the key-dependent coefficients of the polynomial. A probabilistic version of the interpolation attack has been published later by Jakobsen [139, 143].

Courtois-Pieprzyk Attack

Under a purely algebraic approach, a first step is to express a given block cipher in a system of equations; actually, every component of a block cipher can be described with help of a set of algebraic equations. If one collects these descriptions, one gets a large system which mathematically defines the complete block cipher. If it is possible to solve this system faster than an exhaustive key search, then the cipher may be considered to be broken. Surprisingly, Ferguson, Schroeppel and Whiting [104] have managed to do in the case of Rijndael. Actually, they express Rijndael as a single equation made of 2^{50} terms. Undoubtedly, such a huge equation is extremely difficult to work with, but it is a nice proof-of-concept.

In a second step, Courtois and Pieprzyk [70] show that the S-boxes of Rijndael and Serpent can be written as an *overdefined* system of algebraic

equations on GF (2); this fact is due to the algebraic essence of the S-box (it is based on the inverse operation in a finite field) in the case of Rijndael and to the small size (4-bit) of the substitution boxes of Serpent. In their paper, they claim that Serpent with 256-bit may be broken faster with exhaustive search using their so-called *XSL-attack*. They exploit previous results of Shamir and Kipnis [293] developed to attack public-key cryptosystems based on multivariate quadratic equations; the key idea is to consider nonlinear terms as independent linear variables and to solve the obtained system using a Gaussian reduction. This process is named *re-linearization* and has been extended later by Courtois et al. [68]. At the time of writing, the exact complexity of these attacks is not known, and an accurate analysis remains an open problem; however, it is worth noticing that such attacks are clearly effective against stream ciphers [65, 66, 69].

Interestingly, Murphy and Robshaw [235] show that it is possible to reduce the complexity of the system of equations describing Rijndael by working on GF $\left(2^8\right)$ instead of GF (2). At the time of writing attacks against the 128-bit key version of Rijndael expressed as a multivariate quadratic equations system over GF $\left(2^8\right)$ are claimed to succeed within about 2^{80} to 2^{100} operations, but this complexity is far to be accepted by the research community.

2.3.6 Other Attacks

Slide Attacks

An interesting concept of attack is the *slide attack* and variations thereof proposed by Biryukov and Wagner [37,38]. An important and unusual property of this attack is that it is independent of the block cipher rounds number. It exploits the degree of self-similarity of the block cipher: for instance, one can apply it against block ciphers having a periodic key schedule. Let $\mathsf{f} = \mathsf{f}_r \circ \ldots \circ \mathsf{f}_1$ be an r-round block cipher and let us denote x_i the encrypted value after i rounds (x_0 denotes thus the plaintext and x_r the ciphertext). If the relation $x_1 = \mathsf{f}_1(x_0, k^{(1)})$ holds, where $k^{(1)}$ is the round subkey used during the first round, then (x_0, x_1) is called a *slid pair*. An adversary gets two relations $x_1 = \mathsf{f}_1(x_0, k^{(1)})$ and $\mathsf{f}_r(x_r, k^{(r)}) = x_{r+1}$ which involve only a single round function, and which can potentially be solved for $k^{(1)}$ and $k^{(r)}$.

This attack can be applied to variants of DES, like the one proposed by Brown and Seberry [52], the Even-Mansour scheme [101], or every Feistel scheme with a 4-rounds periodic key-schedule, for instance. Furthermore, it has been shown that certain *auto-key ciphers* (where the choice of the round key is dependent of the data) are also vulnerable to this attack.

Related Cipher Attack

Certain block ciphers allow a variable round number. Wu demonstrated in [332] that block ciphers with such a property, whose key-schedule does not depend on the total number of rounds, and which share the same round functions may succumb to a *related-cipher attack*.

A related-cipher attack can be applied when the key is fixed, and when the block cipher is used with different round numbers; it can recover the key when the differences between the round numbers is small. More formally, let f_r be a block cipher made of r rounds and f_{r+1} the same block cipher with one round more. Let us admit that an adversary is able to get the decryption $p = \mathsf{f}_r^{-1}(c)$ of a ciphertext c under a key k and the encryption $c' = \mathsf{f}_{r+1}(p)$ of p under the same key by the same cipher with one more round. Then, the adversary knows that $\mathsf{f}(c) = c'$ under the key k. As a single round function is insecure in most of block ciphers, there is a high probability that the adversary is able to recover the round subkey. More generally, the adversary may use this method to learn the remaining round subkeys, and eventually the key, provided the key-schedule algorithm is not one-way. This attack could still be applied against more than one round, exploiting an elaborated attack against this reduced-round cipher.

2.4 Security Models

To precisely quantify the *security* of a block cipher, and thus prove that it fulfills given security requirements, is an extremely difficult task with nowadays knowledge (ultimately, one should not forget that up to now, *very little* can be proved with respect to the practical security of block ciphers). A first task consists in defining the precise *security model* in which one would like to prove the security of a primitive. A rather intuitive definition of the security of a block cipher is the *K-security* concept of Daemen and Rijmen [81]; obviously, this definition is extremely difficult to work with in a mathematical sense.

Definition 2.4.1 (K-Security). *A block cipher is* K-secure *if all possible attack strategies for it have the same expected work factor and storage requirements as for the majority of possible block ciphers with the same dimensions. This must be the case for all possible modes of access for the adversary (known / chosen / adaptively chosen plaintext / ciphertext, known / chosen / adaptively chosen key relations, ...) and for any* a priori *key distribution.*

The NESSIE project [247] considered furthermore two additional informal (but pragmatic) security models, namely *practical security* and *historical security*: in the first model, which includes most of the cryptanalysis performed nowadays, a block cipher is considered secure if the best-known at-

tack requires too much resources by an acceptable margin. It is a very practical model as one can test the block cipher with different known attacks and assess a certain security level to it. However, it is not possible to predict the security of the underlying block cipher with respect to yet unknown attacks. The *historical security* of a block cipher is derived according to the amount of cryptanalytic work on the ciphers performed over the years. An old block cipher which resists to all cryptanalytical attacks since a long time will inevitably inspire a larger security feeling than a new block cipher which has not been extensively cryptanalysed.

In this section we present additionally different formal approaches described so far in the academic literature; we consider *Shannon's perfect security* model, also named information-theoretical security, the security against *polynomially bounded adversary*, and *ad-hoc proofs of security*.

2.4.1 Perfect Secrecy

In his seminal paper about cryptography, Shannon [295] introduced a model of security known as *perfect security* or *unconditional security*. In this very strong model, one assumes that the adversary is infinitely powerful, but restricted to a ciphertext-only attack.

Let us model a block cipher as two statistically independent probability distributions on the plaintext X and on the key Y, respectively. Then, a cryptosystem f, where $Y = \mathsf{f}_K(X)$ is said to have the property of *perfect secrecy* if the two probability distributions, for all x and y, satisfy

$$\Pr_{X|Y}[X = x|Y = y] = \Pr_{X}[X = x]$$

Perfect security means that the *a posteriori* distribution of the plaintext X after viewing the ciphertext Y is equal to the *a priori* distribution of the plaintext, or in other words, the adversary learns nothing more about the plaintext after having viewed the ciphertext than she knew before. Let us recall Shannon's notion of entropy [294].

Definition 2.4.2 (Entropy of a random variable). *Let X be a discrete random variable defined on a finite set $\mathcal{X}$. The* entropy *of X is defined to be the quantity*

$$H(X) = -\sum_{\substack{x\in\mathcal{X}\text{ s.t.}\\ \Pr[X=x]\neq 0}} \Pr[X = x]\log_2 \Pr[X = x]$$

One can express the concept of perfect secrecy with help of entropy.

Theorem 2.4.1 (Shannon [295]). *Let X and Y be random variables distributed according to the probability distribution of the plaintext and the ciphertext, respectively. Perfect secrecy is equivalent to*

$$H(X) = H(X|Y)$$

A well-known example of perfectly secure cipher is the *one-time pad*, or *Vernam cipher* [321]. Let us assume that the plaintext is a string $x \in \{0,1\}^\ell$ of ℓ bits. The key is defined to be an ℓ-bit string drawn uniformly at random from the set of all ℓ-bit strings and statistically independent of the plaintext, and the ciphertext is just the bitwise XOR operation between the plaintext and the key. Then, provided the key is used to encrypt a *single* plaintext (otherwise, the one-time-pad would suffer from a trivial known-plaintext attack), the Vernam cipher is a perfect cipher. Unfortunately, the one-time-pad suffers from two major drawbacks: the key is very large, and one cannot reuse it. However, it is the price to have perfect secrecy, as stated by the following theorem, due to Shannon [295].

Theorem 2.4.2. *Let X and Y be random variables distributed according to the probability distribution of the plaintext and the ciphertext, respectively. Perfect secrecy implies*

$$H(K) \geq H(X)$$

In other words, in a perfect cipher, the key must be at least as large as the plaintext.

2.4.2 Security against Bounded Adversaries

Mainly due to the fact that perfect secrecy is a too strong model of security which leads to unpractical constructions, modern cryptography assumes usually that the adversary's power (like computational resources, memory resources, or the number of queries to an oracle), are bounded.

Bounded-Storage Model

In the bounded-storage model, proposed by Maurer [207, 208], one assumes that an adversary has an infinite computational power, but is limited in terms of storage capacity. Known ciphers in the bounded-storage model make largely use of a *randomizer*, i.e. a large amount of public randomness which can be tampered by the adversary, but not stored completely.

Luby-Rackoff Model

The field of provable security for block ciphers has probably been initiated by Luby and Rackoff in their seminal article [191] where they proved that a three-round Feistel cipher (see Def. 2.2.1) involving statistically independent pseudorandom functions results in a pseudorandom permutation. Since then, many researchers have studied, improved and extended their results (see [164, 192, 209, 213, 241, 260, 261, 263, 265, 271, 336]).

In the Luby-Rackoff security model, a distinguisher $\boldsymbol{\delta}^\nu$ is a computationally (and memory) unbounded Turing machine which can play with an

oracle $\mathbf{\Omega}$ implementing a permutation. This permutation is, with probability $\pi_0 = \frac{1}{2}$ the block cipher C under review (*i.e.* a random permutation defined by a block cipher on $\{0,1\}^m$, where m is the block size and the randomness comes from the choice of the key), or with probability $\pi_1 = \frac{1}{2}$ a permutation C^* chosen uniformly at random from the set of all permutations on $\{0,1\}^m$; the latter is usually called the "*perfect cipher*".

The distinguisher $\boldsymbol{\delta}^\nu$ can submit a *bounded* number ν of queries to $\mathbf{\Omega}$ and ultimately outputs a decision bit "0" (if it guesses that C^* was implemented by $\mathbf{\Omega}$) or "1" (if it guesses that C was implemented by $\mathbf{\Omega}$); one is interested in characterizing and computing its *advantage*

$$\mathrm{Adv}_{\boldsymbol{\delta}^\nu}(\mathsf{C}, \mathsf{C}^*) = \left| \Pr_{\mathsf{C}}[\boldsymbol{\delta}^\nu(\mathbf{x}) = 1] - \Pr_{\mathsf{C}^*}[\boldsymbol{\delta}^\nu(\mathbf{x}) = 1] \right|$$

where $\mathbf{x} = (x_1, \ldots, x_\nu)$ is the vector of the values queried to the oracle; another important measure is the *best advantage* of any distinguisher:

$$\mathrm{BestAdv}^\nu(\mathsf{C}, \mathsf{C}^*) = \max_{\boldsymbol{\delta}^\nu} \mathrm{Adv}_{\boldsymbol{\delta}^\nu}(\mathsf{C}, \mathsf{C}^*)$$

Here, the maximum is taken over the set of all possible (non-adaptive and computationally unbounded) distinguishers between C and C^*. There is an important difference between *adaptive* and *non-adaptive* distinguishers: an adaptive distinguisher is allowed to wait for an answer before submitting the next query, which may thus be a function of the previous answer. In this model, a "security proof" means that one is able to provide an acceptable upper-bound on $\mathrm{BestAdv}^\nu(\mathsf{C}, \mathsf{C}^*)$ for a given block cipher C. As outlined above, the current state of research is able to give security proofs only for very few constructions, mainly for high-level schemes, like the Feistel cipher.

Vaudenay's *decorrelation theory* (see [313, 314, 316–318, 320]) is a set of mathematical tools which aims at studying and defining the security of block ciphers in the Luby-Rackoff model. Basically, a central concept of decorrelation theory is the so-called *decorrelation matrix of order* ν. Given a function f_K parametered by a random key K from a set $\mathcal{A}$ to a set $\mathcal{B}$, its decorrelation matrix of order ν is defined to be

$$[\mathsf{f}_K]^\nu_{(x_1,\ldots,x_\nu),(y_1,\ldots,y_\nu)} = \Pr_K[\mathsf{f}_K(x_1) = y_1 \wedge \ldots \wedge \mathsf{f}_K(x_\nu) = y_\nu]$$

where the probability is taken on the key K, and $[\mathsf{f}_K]^\nu_{(x_1,\ldots,x_\nu),(y_1,\ldots,y_\nu)}$ is a real matrix defined on $\mathcal{A}^\nu \times \mathcal{B}^\nu$ where the rows are numbered by input ν-tuples and the columns are numbered by output ν-tuples. Given a norm $||.||$ on the vector space of $\mathcal{A}^\nu \times \mathcal{B}^\nu$-type real matrices, the ν-wise *decorrelation bias* of f is defined by

$$\mathrm{Dec}^\nu(\mathsf{f}_K) = || \, [\mathsf{f}_K]^\nu_{(x_1,\ldots,x_\nu),(y_1,\ldots,y_\nu)} - [\mathsf{f}^*_K]^\nu_{(x_1,\ldots,x_\nu),(y_1,\ldots,y_\nu)} \, ||$$

where $[\mathsf{f}^*_K]^\nu_{(x_1,\ldots,x_\nu),(y_1,\ldots,y_\nu)}$ is the ν-wise decorrelation matrix of the canonical ideal random function f^*_K.

The link to the best advantage of any adaptive distinguisher limited to ν queries is the following: for a matrix $M \in \mathbb{R}^{\mathcal{A}^\nu \times \mathcal{B}^\nu}$, one defines the following norm:

$$||M||_a = \max_{x_1} \sum_{y_1} \max_{x_2} \sum_{y_2} \cdots \max_{x_\nu} \sum_{y_\nu} |M_{(x_1,\ldots,x_\nu)(y_1,\ldots,y_\nu)}|$$

Then, one can show that

$$\mathrm{BestAdv}^\nu(\mathsf{f}_K, \mathsf{f}_K^*) = \frac{1}{2} \cdot \left|\left| [\mathsf{f}_K]^\nu_{(x_1,\ldots,x_\nu),(y_1,\ldots,y_\nu)} - [\mathsf{f}_K^*]^\nu_{(x_1,\ldots,x_\nu),(y_1,\ldots,y_\nu)} \right|\right|_a$$

As a constructive example of the application of decorrelation theory, one can mention the *Peanut construction* (over which DFC is based) which is basically an r-round Feistel scheme with decorrelation modules as a round function. If these decorrelation modules achieve a ν-wise decorrelation bias of ε, by using the multiplicative properties of the matrix norm and the triangular inequality with a truly random 3-round Feistel construction, one obtains from the Luby-Rackoff Theorem [191] that

$$\mathrm{Dec}^\nu(\mathsf{f}_K) \leq \left(2\nu^2 2^{-\frac{m}{2}} + \varepsilon\right)^{\lfloor \frac{r}{3} \rfloor}$$

where m is the block length in bits.

2.4.3 Ad-Hoc Proofs of Security

Theoretical notions of security towards practical attacks like differential [31–33] and linear cryptanalysis [202, 203] have been defined quite early by Nyberg [249] and by Chabaud and Vaudenay [58], respectively. This approach leads to ad-hoc proofs where maximal (multi-path) linear and differential probabilities are upper-bounded for a given construction (as in [10, 155, 156, 251, 252, 257, 258], for instance). One of the first examples of practical, real-world block cipher based on such ad-hoc proofs of security with regards to linear and differential cryptanalysis is Misty1 [205].

However, one must remain fully aware of the fact that ad-hoc proofs of security are not *real* proofs of security: they are merely heuristic arguments that a given attack strategy can not apply on a certain block cipher, but they do not imply that this block cipher possesses any security property towards other (and maybe still unknown) kinds of attacks. For instance, the prototype cipher PURE, which is a variant of a family proposed by Knudsen and Nyberg [252] and shown to be secure against differential cryptanalysis has been attacked using an interpolation attack [142]; another example is the COCONUT98 block cipher proposed by Vaudenay [313] and immune to linear and differential cryptanalysis: it succumbs to Wagner's boomerang attack [322] and to a linear-differential cryptanalysis [29].

Furthermore, we have to note the difference which exists between "proofs of security" and "arguments for the security" of a block cipher. For instance, block cipher designers often compute upper bounds on the maximum differential probability of any differential *characteristic* and conclude that their cipher is immune to differential cryptanalysis: it is not really a mathematical proof of resistance towards differential cryptanalysis, since it does not take into account the cumulative aspect of differentials. However, it is still a strong argument in favor of the immunity of that cipher towards differential attacks. And finally, we have to keep in mind that such "proofs of security" often assume hypotheses which are not formally true in the reality: for instance, one often assumes that, in an iterative block cipher, the round subkeys are uniformly distributed and statistically independent; although such an assumption is usually not likely to be true, because of the nature of the key-schedule algorithm, it often captures that reality in an acceptable way.

Chapter 3

Statistical Cryptanalysis of Block Ciphers

Historically, statistical procedures have always been associated with cryptanalytic attacks against block ciphers. In the "modern" era of cryptanalysis, one of the first ever published attack exploiting statistical correlations in the core of DES [242] is Davies and Murphy's attack [83]. Biham and Shamir's differential cryptanalysis [31–33], Matsui's attack against DES [202, 203], Vaudenay's statistical and χ^2 cryptanalysis [312], Harpes and Massey's partitioning cryptanalysis [126], Gilbert-Minier stochastic cryptanalysis [225], and Wagner's boomerang attack [322], as well as all the attacks derived from those, are attacks using statistical procedures in their core (we refer the reader to §2.3). To the best of our knowledge, Murphy *et al.* proposed for the first time in an unpublished report [234] a *general statistical framework* for the analysis of block ciphers using the technique of *likelihood estimation*. Other examples can be found in the field of cryptology: recently, Coppersmith, Halevi and Jutla [61] have devised a general statistical framework for analysing stream ciphers; they used the concept of statistical hypothesis testing for systematically distinguishing a stream cipher from a random function. Other examples (this list being non-exhaustive) include Maurer's analysis of Simmon's authentication theory [210,211] and Cachin's theoretical treatment of steganography [54, 55]. In a parallel way, some attempts to formalize the resistance of block ciphers towards cryptanalytic attacks have been proposed: for instance, Pornin [267] derived a general criterion of resistance against the Davies and Murphy attack; for this purpose, he made use of statistical hypothesis testing. Vaudenay, in a sequence of papers (e.g. [313, 316, 320]) proposed the *decorrelation theory* as a generic technique for estimating the strength of block ciphers against various kinds of attacks. In these papers, he notably derived bounds on the best advantage of any linear and differential distinguishers, however without using

statistical hypothesis testing concepts.

As pointed out by many authors, statistical hypothesis tests are convenient in the analysis of statistical problems, since, in certain cases, well-known optimality results (like the Neyman-Pearson lemma, for instance) can be applied. In this chapter, we consider the resistance of block ciphers against linear and differential cryptanalysis, as a first step, and against general iterated attacks as a second step, as a statistical hypothesis testing problem. This allows us to derive several bounds on the best advantage of a wide class of attacks and to state optimality results on the decision processes involved during these attacks.

3.1 The Neyman-Pearson Paradigm

Statistical hypothesis testing is a formal way for *distinguishing* between probability distributions on the basis of random variables generated from one of these distributions; obviously, this is a frequently encountered situation in cryptanalysis. A useful and well-known statistical framework is the *Neyman-Pearson approach*; it will build the mathematical foundations of our considerations about deriving optimal attacks. In this framework, the probability distributions are grouped into two aggregates, one of which is called the *null hypothesis*, and is denoted by $\mathcal{H}_0$, and the other of which is called the *alternative hypothesis* and is denoted by $\mathcal{H}_1$. For instance, $\mathcal{H}_0$ might state that the distribution of a random variable modeling the information obtained during an attack was a normal law with mean μ_0 and variance σ, while the alternative hypothesis $\mathcal{H}_1$ might state that the random variable was distributed according to a normal law with mean $\mu_1 \neq \mu_0$ and the same variance σ. When both hypotheses are completely determined, as it is the case in this simple example, such hypotheses are called *simple hypotheses*. When a hypothesis does not completely specify the probability distribution under consideration, one calls it a *composite hypothesis*. We now describe formally the Neyman-Pearson paradigm.

According to the Neyman-Pearson paradigm, a decision as to whether or not reject the null hypothesis $\mathcal{H}_0$ in favor of $\mathcal{H}_1$ is made on the basis of a *statistic*[1] $\mathsf{M}(x)$, where x denotes the sample value taking values on a set $\mathcal{X}$. The sets of values of M for which $\mathcal{H}_0$ is accepted or rejected are called the *acceptance region*, denoted $\mathcal{A}$, and the *rejection region*, denoted $\mathcal{A}^c$, respectively. More formally,

$$\mathcal{A} = \{x : \boldsymbol{\delta}(\mathsf{M}(x)) = 0\} \text{ and } \mathcal{A}^c = \{x : \boldsymbol{\delta}(\mathsf{M}(x)) = 1\}$$

[1] A *statistic* is a function on samples, such that any possible sample from a population is paired with a value of the statistic. A statistic is in fact a random variable, and there is a probability distribution that identifies for all its different possible values the probability of each being measured. This distribution is called the *sampling distribution* to avoid confusion with the distribution of values in the population.

where $\boldsymbol{\delta} : \mathbb{R} \to \{0, 1\}$ is a *decision rule* (or a *test*) mapping any value of $\mathsf{M}(x)$ either to $\mathcal{H}_0$ or to $\mathcal{H}_1$. The decision rule $\boldsymbol{\delta}$ might make two types of error.

Definition 3.1.1 (Type I/II Error Probabilities). *The probability that a decision rule $\boldsymbol{\delta}$ reject the null hypothesis $\mathcal{H}_0$ when it is actually true is called* type I error *and is noted*

$$\alpha = \Pr_{X \leftarrow \mathsf{D}} [\boldsymbol{\delta}(\mathsf{M}(X)) = 1 | \mathsf{D} \in \mathcal{H}_0] . \tag{3.1}$$

The probability that a decision rule $\boldsymbol{\delta}$ accept the null hypothesis $\mathcal{H}_0$ when it is actually false is called type II error *and is noted*

$$\beta = \Pr_{X \leftarrow \mathsf{D}} [\boldsymbol{\delta}(\mathsf{M}(X)) = 0 | \mathsf{D} \in \mathcal{H}_1] . \tag{3.2}$$

If $\mathcal{H}_0$ is simple, one calls α the *significance level*, or the *size* of the test; if it is composite, α generally depends on which particular member of $\mathcal{H}_0$ is true: in this case, the significance level is defined to be the least upper bound of these probabilities. Similarly, if $\mathcal{H}_1$ is composite, β depends on which particular member of $\mathcal{H}_1$ holds as well. Finally, the probability that $\mathcal{H}_0$ is rejected when it is false is called the *power* of the test; hence, the power equals $1 - \beta$. An *ideal* test would have $\alpha = \beta = 0$, but this can be achieved only in trivial cases [274]. In practice, it is furthermore always the case that, for a fixed sample size, β must be increased in order to decrease the significance level, and vice-versa. The Neyman-Pearson approach resolves this conflict by imposing an *asymmetry* between the two hypotheses: the significance level is fixed in advance and then an attempt is made to construct a test yielding a small value for β.

3.1.1 Likelihood-Ratio Tests

In the simplest case of hypothesis testing, we have to decide between two fully characterized distributions. Let X be a random variable with $X \leftarrow \mathsf{D}$, and let us consider two hypotheses: $\mathcal{H}_0 = \{\mathsf{D}_0\}$ and $\mathcal{H}_1 = \{\mathsf{D}_1\}$, where D_0 and D_1 are known. Thus, we have to decide between $\mathsf{D} = \mathsf{D}_0$ and $\mathsf{D} = \mathsf{D}_1$. The *Neyman-Pearson lemma* [248] derives the shape of the optimum test between two probability distributions[2].

Lemma 3.1.1 (Neyman-Pearson). *Let X be a random variable distributed according to $X \leftarrow \mathsf{D}$ on a finite set $\mathcal{X}$ and let the decision problem corresponding to the hypotheses $\mathcal{H}_0 = \{\mathsf{D}_0\}$ and $\mathcal{H}_1 = \{\mathsf{D}_1\}$. For $\tau \geq 0$, let the acceptance region $\mathcal{A}_\tau$ be defined by*

$$\mathcal{A}_\tau = \left\{ x : \Pr_{\mathsf{D}_0}[X = x] \geq \tau \cdot \Pr_{\mathsf{D}_1}[X = x] \right\} . \tag{3.3}$$

[2]We state the Neyman-Pearson lemma for discrete probability distributions, but the same results hold for continuous distributions as well.

Let α and β be the corresponding probabilities of error defined as in Eq. (3.1) *and Eq.* (3.2)*, respectively. Let $\mathcal{B}$ be any other decision region with associated probabilities of error α' and β'. If $\alpha' \leq \alpha$, then $\beta' \geq \beta$.*

Proof. Let $1_{\mathcal{A}_\tau}$ and $1_{\mathcal{B}}$ be the indicator functions of the decision regions $\mathcal{A}_\tau$ and $\mathcal{B}$, respectively. Then, for all $x \in \mathcal{X}$,

$$(1_{\mathcal{A}_\tau}(x) - 1_{\mathcal{B}}(x)) \cdot \left(\Pr_{X \leftarrow \mathsf{D}_0}[X = x] - \tau \cdot \Pr_{X \leftarrow \mathsf{D}_1}[X = x] \right) \geq 0$$

since it is always the product of two terms with the same sign. Let

$$\pi_{\mathsf{D}_0}(x) = \Pr_{X \leftarrow \mathsf{D}_0}[X = x] \text{ and } \pi_{\mathsf{D}_1}(x) = \Pr_{X \leftarrow \mathsf{D}_1}[X = x].$$

Then, we sum over all $x \in \mathcal{X}$ and obtain

$$\begin{aligned} 0 &\leq \sum_{x \in \mathcal{X}} (1_{\mathcal{A}_\tau}(x) - 1_{\mathcal{B}}(x)) \cdot (\pi_{\mathsf{D}_0}(x) - \tau \cdot \pi_{\mathsf{D}_1}(x)) \\ &= \sum_{x \in \mathcal{A}_\tau} (\pi_{\mathsf{D}_0}(x) - \tau \pi_{\mathsf{D}_0}(x)) - \sum_{x \in \mathcal{B}} (\pi_{\mathsf{D}_0}(x) - \tau \pi_{\mathsf{D}_0}(x)) \\ &= (1 - \alpha) - \tau\beta - (1 - \alpha') + \tau\beta' \\ &= \tau(\beta' - \beta) - (\alpha - \alpha'). \end{aligned}$$

The lemma follows from $\tau \geq 0$. □

Let us now express Eq. (3.3) in an alternate way:

$$\mathsf{M}_{\mathrm{lr}} : \left\{ \begin{array}{ccc} \mathcal{X} & \longrightarrow & [0, +\infty] \\ x & \mapsto & \frac{\Pr_{X \leftarrow \mathsf{D}_0}[X=x]}{\Pr_{X \leftarrow \mathsf{D}_1}[X=x]} \end{array} \right. \tag{3.4}$$

with the convention[3] that $\frac{x}{0} = +\infty$ for $0 < x < 1$. Interpreted in terms of M_{lr}, the Neyman-Pearson lemma indicates that the optimum test (regarding error probabilities) in case of a binary decision problem is a test based on the statistic which is called a *likelihood-ratio*, and that the test simply consists in comparing the value of $\mathsf{M}_{\mathrm{lr}}(x)$ to a fixed threshold τ. Practically, there still exists the problem of choosing τ and the desirable type I or type II error probabilities. Another possibility consists in following a *Bayesian approach* and to assign *prior* probabilities π_0 and $\pi_1 = 1 - \pi_0$ to both hypotheses. If we assume that correct decisions are not penalized and incorrect decisions are penalized equally, which is a desired property in a completely symmetrical approach, then one can show [71, page 314] that an optimal decision rule is based on the statistic

$$\mathsf{M}_{\mathrm{blr}} : \left\{ \begin{array}{ccc} \mathcal{X} & \longrightarrow & [0, +\infty] \\ x & \mapsto & \frac{\pi_0 \Pr_{X \leftarrow \mathsf{D}_0}[X=x]}{\pi_1 \Pr_{X \leftarrow \mathsf{D}_1}[X=x]} \end{array} \right. \tag{3.5}$$

[3] Note that the case $\frac{0}{0}$ must never occur when dealing with discrete probability spaces.

and chooses $\mathcal{H}_0$ if and only if $\mathsf{M}_{\mathrm{blr}}(x) \geq 1$. It is optimal in the sense that it minimizes the *overall error probability* defined by

$$\pi_{\mathrm{e}} = \pi_0 \alpha + \pi_1 \beta.$$

3.1.2 Generalized Likelihood-Ratio Tests

As outlined in the previous paragraph, the test based on the likelihood ratio M_{lr} is optimal for choosing between two simple hypotheses. It is actually possible [274, page 308] to develop generalizations of this test for use in situations in which the hypotheses are not simple. Such tests are not generally optimal, but they are typically not optimal in situations for which no optimal test exists.

It is often the case that the hypotheses under consideration specify, or partially specify, the values of parameters of the probability distribution according to which the sample data are distributed. Let $\boldsymbol{X} = (X_1, \ldots, X_\nu)$ be a random vector[4] representing the sample data, and let $\mathsf{f}_{\mathsf{D}}(\boldsymbol{x}, \theta)$ be their joint probability density function (which depends on a parameter θ). Then, for instance, the null hypothesis $\mathcal{H}_0$ may specify that $\theta \in \mathcal{P}_0$, where $\mathcal{P}_0 \subset \mathcal{P}$ is a subset of the set $\mathcal{P}$ of all possible values for θ, and the alternate hypothesis may specify that $\theta \in \mathcal{P}_1$, where $\mathcal{P}_1 \subset \mathcal{P}$ and $\mathcal{P}_0 \cap \mathcal{P}_1 = \emptyset$. Let us furthermore define $\mathcal{P}' = \mathcal{P}_0 \cup \mathcal{P}_1$. Based on these data, a plausible measure to compare both hypotheses may be the ratio of their likelihood. If the hypotheses are composite, each likelihood ratio is evaluated at the value of θ which maximizes it:

$$\mathsf{M}'_{\mathrm{glr}}(\boldsymbol{x}) = \frac{\max_{\theta \in \mathcal{P}_0} \mathsf{f}_{\mathsf{D}}(\boldsymbol{x}, \theta)}{\max_{\theta \in \mathcal{P}_1} \mathsf{f}_{\mathsf{D}}(\boldsymbol{x}, \theta)} \tag{3.6}$$

Hence, small values of $\mathsf{M}'_{\mathrm{glr}}(.)$ tends to discredit the null hypothesis. Note that it is preferable, for some technical reasons [274], to use the statistic

$$\mathsf{M}_{\mathrm{glr}}(\boldsymbol{x}) = \frac{\max_{\theta \in \mathcal{P}_0} \mathsf{f}_{\mathsf{D}}(\boldsymbol{x}, \theta)}{\max_{\theta \in \mathcal{P}'} \mathsf{f}_{\mathsf{D}}(\boldsymbol{x}, \theta)} \tag{3.7}$$

rather than the one defined in Eq. (3.6). Since $\mathsf{M}_{\mathrm{glr}} = \min\{\mathsf{M}'_{\mathrm{glr}}, 1\}$, small values of $\mathsf{M}'_{\mathrm{glr}}$ correspond to small values of $\mathsf{M}_{\mathrm{glr}}$. In order for this likelihood-ratio test to have a significance level equal to α, the threshold τ must be chosen such that

$$\Pr_{\boldsymbol{X} \leftarrow \mathsf{D}} [\mathsf{M}_{\mathrm{glr}} \leq \tau | \mathsf{D} \in \mathcal{H}_0] = \alpha.$$

Provided the sampling distribution of $\mathsf{M}_{\mathrm{glr}}$ is known under the null hypothesis, then it is sometimes possible to determine the corresponding threshold τ, but this sampling distribution is generally not of a simple form; however,

[4] We refer the reader to the definition of a random variable (Def. A.1.3) and to the generalization to random vectors described thereafter.

in many situations, the following (informally stated) theorem [274] allows to gain some insight to derive a valid approximation. The *dimensions* of $\mathcal{P}_0$ and $\mathcal{P}'$ are the numbers of *free* parameters fully determining the distributions in $\mathcal{H}_0$ and in $\mathcal{H}_1$.

Theorem 3.1.1. *Under certain smoothness conditions on the involved probability density function, the null distribution of* $-2\log \mathsf{M}_{\text{glr}}$ *tends to a* χ^2 *distribution with degrees of freedom equal to the dimension of* $\mathcal{P}'$ *minus the dimension of* $\mathcal{P}_0$ *as the sample size tends to infinity.*

An example of generalized likelihood-ratio test particularly useful in the context of cryptanalytical attacks is the following: we would like to test the goodness of fit of a model for *a given multinomial probability distribution*[5]. More formally, we would like to judge the plausibility of a multinomial distribution (described by a vector $\boldsymbol{p}(\theta)$ which possibly depends on a parameter[6] θ) relatively to an alternative hypothesis $\mathcal{H}_1$ which contains all possible vectors $\boldsymbol{p}'$ under the sole constraint that their components sum up to 1. If the vectors are m-valued, $\mathcal{P}'$ is thus the set consisting of m nonnegative numbers whose sum is equal to 1. In this case, the numerator of Eq. (3.7) can be written as

$$\max_{\boldsymbol{p}\in\mathcal{P}_0}\left(\frac{n!}{\hat{x}_1!\cdots\hat{x}_m!}\right)p_1(\theta)^{\hat{x}_1}\cdots p_m(\theta)^{\hat{x}_m} \tag{3.8}$$

where the $\hat{x}_i$'s are the observed counts in the m components of the vector. In case where Eq. (3.8) depends on θ, this equation has to be maximized in terms of θ, leading to an "optimal" $\bar{\theta}$. We will denote the corresponding probabilities by $p_i(\bar{\theta})$. Since the probabilities are unrestricted under $\mathcal{P}'$, one can show that the denominator of Eq. (3.7) is maximized by $\bar{p}_1^i = \frac{\hat{x}_i}{n}$. Rewriting Eq. (3.7), we get

$$-2\log \mathsf{M}_{\text{glr}} = 2\sum_{i=1}^{m}\hat{x}_i\log\left(\frac{\hat{x}_i}{np_i(\bar{\theta})}\right)$$

Under $\mathcal{P}'$, the cell probabilities are allowed to be free, with the constraint that they sum to 1, so the dimension of $\mathcal{P}'$ is equal to $m-1$. Provided that the probabilities $\boldsymbol{p}(\bar{\theta})$ depend on a k-dimensional parameter θ, the dimension of $\mathcal{P}_0$ is equal to k. Then, according to Theorem 3.1.1, the sampling distribution of $-2\log \mathsf{M}_{\text{glr}}$ is thus a χ^2 distribution with $m-k-1$ degrees of freedom.

Note furthermore that, with help of a Taylor series argument, it possible to give an heuristic argument [274] indicating that, at order 2, Eq. (3.7) and

[5] We refer the reader to Def. A.2.3.

[6] In a cryptanalytical context, $\boldsymbol{p}(\theta)$ will be most of the time the uniform distribution, and thus, do not even depend on θ.

Pearson's $\hat{\chi}^2$ statistic

$$\hat{\chi}^2 = \sum_{i=1}^{m} \frac{\left(\hat{x}_i - np_i(\bar{\theta})\right)^2}{np_i(\bar{\theta})} \tag{3.9}$$

are *asymptotically equivalent* under $\mathcal{H}_0$.

3.2 Linear Cryptanalysis of DES Revisited

In this section, we describe in detail, discuss and improve several aspects of Matsui's *linear cryptanalysis*, which is a generic technique invented to attack block ciphers; some basic aspects were already covered in §2.3.3. First, we recall some historical aspects as well as required preliminaries about linear cryptanalysis in §3.2.2; then we present a statistical modelization of its success probability and we show how to fully exploit the information gathered during a linear cryptanalysis by deriving generic and optimal key ranking procedures. Second, we describe and discuss in §3.2.5 experimental results about the real complexity of a linear cryptanalysis when applied to DES.

3.2.1 Historical Perspectives

Ideas inducing to linear cryptanalysis can be traced back to an early observation of Shamir[7] [292] presented during the conference Crypto'85; in this short paper, Shamir demonstrates that there is a clear statistical correlation between the XOR of all the output bits of DES's S-boxes and the second input bits; he was however not able to exploit these correlations. He observes furthermore that DES's S-box S_5 is the most *biased* one according to this criterion. Later, Tardy-Corfdir and Gilbert[8] [308] presented a *known-plaintext* attack of statistical nature against 4-rounds and 6-rounds FEAL. This attack is based on the fact that one can approximate the addition modulo 256 using a *linear approximation* involving the XOR of some well-chosen input and output bits, i.e. a Boolean equation being probabilistically true. According to [308], the principles of the attack can be described as follows:

> "[...] *Our attack is a statistical variant of the well-known* meet-in-the-middle *method. It is based on two kinds of relations :*

[7] According to [292], correlations between the XOR of the output bits and the second input bit were independently observed by Franklin in his M.Sc. thesis submitted two months before Crypto'85.

[8] Actually, the nomenclature "*Matsui's linear cryptanalysis*" seems to be an instance of Stigler's law of eponymy, stating that "*no law, theorem, or discovery is named after its originator*", as Tardy-Corfdir-Gilbert [308] attack against FEAL is nothing but a linear cryptanalysis. Note that Stigler is a statistician who did not discover the eponymy law!

1. *It uses some key-independent statistics which involve the plaintext and intermediate block of the* FEAL *data randomizer (say the block $x^{(n-1)}$ which appears as an input to the last round of the Feistel scheme).*
2. *In addition, the deciphering algorithm provides a key-dependent relation between this intermediate block and the ciphertext.*

An exhaustive search for the value optimizing the agreement between the a-priori expected statistics (1) and the statistics deduced from the ciphertext (2) provides the part of the expanded key involved in (2)."

Matsui and Yamagishi [206] presented then at EUROCRYPT'92 a variant of Tardy-Corfdir and Gilbert's attack using *exact* linear expressions, i.e. equations holding with probability one, to attack the same cipher, but with much less necessary known plaintext-ciphertext pairs. The technique has been considerably extended and refined by Matsui [202] in an attack against DES. Finally, key-ranking procedures have been introduced by Matsui [203], as well as a report on the first[9] successful experimental attack against DES.

3.2.2 Basic Attack

In this part, we describe thoroughly the versions published in [202, 203] of both attacks of Matsui against DES [242]. As outlined in §2.3.3, linear cryptanalysis exploits the imbalance of *linear approximations*. Such a probabilistic equation involves the sum (where the addition is the one of GF (2), namely the XOR operation) of some input bits, of some output bits, and of some key bits. For the sake of convenience, the selection of the involved input, output, and key bits, which are seen as vectors over GF (2), is often expressed in terms of the inner dot-product of bit masks with the input, output, and key variables, respectively. As the inner dot-product is usually defined for vectors over $\mathbb{R}^n$ or $\mathbb{C}^n$ (and not for vectors over finite fields), we give here a formal definition thereof. Note that in this thesis, when using the inner dot-product operation, we will always assume that the underlying field has a characteristic equal to 2.

Definition 3.2.1 (Inner Dot-Product over $\mathrm{GF}(2^n)^m$). *Let the finite field of characteristic 2 with 2^n elements be denoted by $\mathrm{GF}(2^n) = (\mathcal{F}, +, \times)$. Let $\mathbf{x}$ and $\mathbf{y}$ be m-valued vectors over $\mathcal{F}^m$. Then, the* inner dot-product *of $\mathbf{x}$ and $\mathbf{y}$, denoted $\mathbf{x} \cdot \mathbf{y}$ is defined by*

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{m} x_i y_i$$

[9] To be more precise, it is at least the first publicly reported attack.

where x_i denotes the i-th component of $\mathbf{x}$.

Note that the addition in a field of characteristic 2 is frequently denoted $\oplus$ and we will adopt this convention in the following. Let us now denote by $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{k}$ the vectors over GF (2) representing the plaintext, the ciphertext, and the key, respectively. Then, a linear relation can be written, with help of Def. 3.2.1, as

$$\mathbf{a} \cdot \mathbf{x} \oplus \mathbf{b} \cdot \mathbf{y} = \mathbf{c} \cdot \mathbf{k} \tag{3.10}$$

with $\mathbf{y} = \mathsf{f}_{\mathbf{k}}(\mathbf{x})$ for a block cipher $\mathsf{f}(.)$. Another convenient way to write Eq. (3.10) frequently encountered in the literature is

$$\mathbf{x}_{\{i_1,\ldots,i_a\}} \oplus \mathbf{y}_{\{j_1,\ldots,j_b\}} = \mathbf{k}_{\{k_1,\ldots,k_c\}}$$

where $\mathbf{z}_{\{l_1,\ldots,l_d\}}$ means "sum of the bits of $\mathbf{z}$ numbered $l_1, \ldots, l_d$":

$$\mathbf{z}_{\{l_1,\ldots,l_d\}} = \bigoplus_{i=1}^{d} \mathbf{z}_{l_i}$$

As the substitution boxes of DES are the sole non-linear components in the entire cipher, we obviously need to study their non-linearity to begin a linear cryptanalysis. Matsui introduced in [202] the concept of *local S-box approximation* for this purpose.

Definition 3.2.2 (Local S-box approximation). *Let* S *be an S-box defined as* $\mathsf{S} : \{0,1\}^{t_1} \rightarrow \{0,1\}^{t_2}$*; let* $\mathbf{a} \in \{0,1\}^{t_1}$ *and* $\mathbf{b} \in \{0,1\}^{t_2}$ *be fixed bit masks. Then, the* local approximation *of* S *for the masks* $\mathbf{a}$ *and* $\mathbf{b}$ *is the number defined by*

$$\mathrm{NS}^{\mathsf{S}}(\mathbf{a}, \mathbf{b}) = \#\left\{\mathbf{x} \in \mathrm{GF}\,(2)^{t_1} : \mathbf{a} \cdot \mathbf{x} = \mathbf{b} \cdot \mathsf{S}(\mathbf{x})\right\} \tag{3.11}$$

Clearly, there exists an effective linear statistical correlation as soon as $\mathrm{NS}^{\mathsf{S}}(\mathbf{a}, \mathbf{b}) \neq 2^{t_1}$ for some $\mathbf{a}$ and $\mathbf{b} \neq \mathbf{0}$. In the case of DES, it is possible to compute the $\mathrm{NS}^{\mathsf{S}_i}(\mathbf{a}, \mathbf{b})$ for all S-boxes S_i and all the possible masks $\mathbf{a}$ and $\mathbf{b}$, since the S-boxes have a relatively small size. The most biased S-box of DES is S_5 for the masks $\mathbf{a} = \texttt{0x10}$ and $\mathbf{b} = \texttt{0xF}$:

$$\mathrm{NS}^{\mathsf{S}_5}(\texttt{0x10}, \texttt{0xF}) = 12$$

One would expect a value of 32 in an unbiased S-box. Note that the masks exactly correspond to the above mentioned observation by Shamir [292]. Using the definition of DES's round function (see §2.2.1), one can write a biased linear approximation of a round function $\mathsf{f}^{(i)}$ of DES derived from the

local approximation of S_5 as follows[10]:

$$\mathbf{x}^{(i)}_{\{16\}} \oplus \mathsf{f}^{(i)}_{\{2,7,13,24\}}(\mathbf{x}^{(i)}) = \mathbf{k}^{(i)}_{\{9\}} \tag{3.12}$$

where $\mathbf{k}^{(i)}$ is the subkey used in round i. Hence, this linear relation holds with probability $\frac{12}{64}$. To extend Eq. (3.12) to a round, one needs the following simple results, which show how behave masks in the case of a XOR and a "branching" operation.

Lemma 3.2.1. *Let* $\mathbf{a}, \mathbf{b} \in \{0,1\}^t$ *be two bit masks. Let* $\mathsf{f} : \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^t$ *defined by* $\mathsf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{x} \oplus \mathbf{y}$ *and* $\mathsf{g} : \{0,1\}^t \to \{0,1\}^t \times \{0,1\}^t$ *defined by* $\mathsf{g}(\mathbf{x}) = \mathbf{x}||\mathbf{x}$. *Then the following equalities hold:*

$$\begin{aligned} \mathbf{a} \cdot \mathbf{x} \oplus \mathbf{a} \cdot \mathbf{y} &= \mathbf{a} \cdot \mathsf{f}(\mathbf{x}, \mathbf{y}) \\ \mathbf{a} \cdot \mathbf{x} &= (\mathbf{b}||\mathbf{a} \oplus \mathbf{b}) \cdot \mathsf{g}(\mathbf{x}) = (\mathbf{a} \oplus \mathbf{b}||\mathbf{b}) \cdot \mathsf{g}(\mathbf{x}) \end{aligned}$$

The above lemma allows us to rewrite Eq. (3.12) for the i-th round of DES.

$$\mathbf{x}^{(i)}_{\mathsf{l}\{2,7,13,24\}} \oplus \mathbf{x}^{(i)}_{\mathsf{r}\{16\}} \oplus \mathbf{x}^{(i+1)}_{\mathsf{r}\{2,7,13,24\}} = k^{(i)}_{\{9\}} \tag{3.13}$$

where $\mathbf{x}_\mathsf{l}$ and $\mathbf{x}_\mathsf{r}$ denote the left and right 32-bit halves in the Feistel scheme, respectively. The linear relation Eq. (3.13) holds also with probability $\frac{12}{64}$. The next natural step consists in extending the linear relation Eq. (3.13) to multiple rounds: for this purpose, one can concatenate single-round relations if the output bit mask of a given relation is the same as the input bit mask of the following relation in order to construct a *linear characteristic* (see Def. 2.3.11). Under the assumption that the input values are statistically independent, one can use Matsui's *piling-up lemma* [202] to compute the bias of the resulting linear approximation.

Lemma 3.2.2 (Piling-up lemma). *Let* $X_1, \ldots, X_\nu$ *be* n *statistically independent Boolean random variables with* $\Pr[X_i = 0] = \frac{1}{2} + \varepsilon_i$. *Then*

$$\Pr\left[\bigoplus_{i=1}^{n} X_i = 0\right] = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i \tag{3.14}$$

A particularly fruitful approach to find linear characteristic consists in finding an *iterative linear characteristic*, i.e. a characteristic having the same input and output masks; such linear approximations are useful because we can apply them on *any* number of rounds. Since the sole components of

[10] We have purposely chosen not to follow Matsui's and FIPS' bit notations to be consistent with the notations used throughout this thesis. Matsui [202, 203] denotes the bit indices from right to left and begins with the index 0, while the standard describing DES [242] numbers the bit from left to right and begins with the index 1. We use a C-like notation, which means that we number the bits from the left to right, and we begin with the index 0.

DES which are non-linear are the S-boxes, one wishes to minimize the number of *active* S-boxes in the linear relation, i.e. S-boxes which have a non-zero output bit mask. Indeed, using an algorithm based on a branch-and-bound technique described in [204], Matsui exhibited [202] the best (i.e. the most biased) r-rounds linear characteristics for $3 \leq r \leq 20$; in [204], Matsui proved that these characteristics are actually the best ones and he showed that such characteristics have *at most* one active S-box per round. Furthermore, an interesting fact is that for certain number of rounds, there exists *two* best linear characteristics having a kind of palindromic shape due to the symmetry of the DES skeleton. The following four linear approximations are the best ones on 14 and 15 rounds of DES.

Theorem 3.2.1 (Best Linear Approximations on 14-Rounds DES). *The best linear approximations on 14-rounds* DES *are*

$$\mathbf{x}_{\mathsf{r}\{7,13,24\}} \oplus \mathbf{y}_{\mathsf{l}\{2,7,13,24\}} \oplus \mathbf{y}_{\mathsf{r}\{16\}} = \mathbf{k}^{(2)}_{\{25\}} \oplus \mathbf{k}^{(3)}_{\{3\}} \oplus \mathbf{k}^{(4)}_{\{25\}} \oplus \mathbf{k}^{(6)}_{\{25\}} \oplus \mathbf{k}^{(7)}_{\{3\}} \oplus \mathbf{k}^{(8)}_{\{25\}} \oplus \mathbf{k}^{(10)}_{\{25\}} \oplus \mathbf{k}^{(11)}_{\{3\}} \oplus \mathbf{k}^{(12)}_{\{25\}} \oplus \mathbf{k}^{(14)}_{\{25\}}$$

and

$$\mathbf{x}_{\mathsf{l}\{2,7,13,24\}} \oplus \mathbf{x}_{\mathsf{r}\{16\}} \oplus \mathbf{y}_{\mathsf{r}\{7,13,24\}} = \mathbf{k}^{(2)}_{\{25\}} \oplus \mathbf{k}^{(4)}_{\{25\}} \oplus \mathbf{k}^{(5)}_{\{3\}} \oplus \mathbf{k}^{(6)}_{\{25\}} \oplus \mathbf{k}^{(8)}_{\{25\}} \oplus \mathbf{k}^{(9)}_{\{3\}} \oplus \mathbf{k}^{(10)}_{\{25\}} \oplus \mathbf{k}^{(12)}_{\{25\}} \oplus \mathbf{k}^{(13)}_{\{3\}} \oplus \mathbf{k}^{(14)}_{\{25\}}$$

where $\mathbf{k}^{(i)}_{\{\mathcal{B}\}}$ *denote the set* $\mathcal{B}$ *of the ith-round subkey. The above two linear approximations hold with probability* $\frac{1}{2} - 1.19 \cdot 2^{-21}$.

Theorem 3.2.2 (Best Linear Approximations on 15-Rounds DES). *The best linear approximations on 15-rounds* DES *are*

$$\mathbf{x}_{\mathsf{l}\{7,13,24\}} \oplus \mathbf{x}_{\mathsf{r}\{15,19\}} \oplus \mathbf{y}_{\mathsf{l}\{2,7,13,24\}} \oplus \mathbf{y}_{\mathsf{r}\{16\}} = \mathbf{k}^{(1)}_{\{24,28\}} \oplus \mathbf{k}^{(3)}_{\{25\}} \oplus \mathbf{k}^{(4)}_{\{3\}} \oplus \mathbf{k}^{(5)}_{\{25\}} \oplus \mathbf{k}^{(7)}_{\{25\}} \oplus \mathbf{k}^{(8)}_{\{3\}} \oplus \mathbf{k}^{(9)}_{\{25\}} \oplus \mathbf{k}^{(11)}_{\{25\}} \oplus \mathbf{k}^{(12)}_{\{3\}} \oplus \mathbf{k}^{(13)}_{\{25\}} \oplus \mathbf{k}^{(15)}_{\{25\}}$$

and

$$\mathbf{x}_{\mathsf{l}\{2,7,13,24\}} \oplus \mathbf{x}_{\mathsf{r}\{16\}} \oplus \mathbf{y}_{\mathsf{l}\{7,13,24\}} \oplus \mathbf{y}_{\mathsf{r}\{15,19\}} = \mathbf{k}^{(1)}_{\{25\}} \oplus \mathbf{k}^{(3)}_{\{25\}} \oplus \mathbf{k}^{(4)}_{\{3\}} \oplus \mathbf{k}^{(5)}_{\{25\}} \oplus \mathbf{k}^{(7)}_{\{25\}} \oplus \mathbf{k}^{(8)}_{\{3\}} \oplus \mathbf{k}^{(9)}_{\{25\}} \oplus \mathbf{k}^{(11)}_{\{25\}} \oplus \mathbf{k}^{(12)}_{\{3\}} \oplus \mathbf{k}^{(13)}_{\{25\}} \oplus \mathbf{k}^{(15)}_{\{24,28\}}$$

where $\mathbf{k}^{(i)}_{\{\mathcal{B}\}}$ *denote the set* $\mathcal{B}$ *of the i-th round subkey. The above two linear approximations hold with probability* $\frac{1}{2} - 1.19 \cdot 2^{-22}$.

1: **Input**: an oracle $\mathbf{\Omega}$, a data complexity ν, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, ε.
2: **Output**: a guess about $\mathbf{c} \cdot \mathbf{k}$
3: Initialize a counter $\hat{m}$ to 0.
4: **for** $i \leftarrow 1$ to $i = \nu$ **do**
5: Generate a plaintext $\mathbf{x}_i$ uniformly at random and independently of the other queries. Submit $\mathbf{x}_i$ to $\mathbf{\Omega}$ and get $\mathbf{y}_i = \mathsf{f}_{\mathbf{k}}(\mathbf{x}_i)$.
6: **if** $\mathbf{a} \cdot \mathbf{x}_i \oplus \mathbf{b} \cdot \mathbf{y}_i = 0$ **then**
7: Increment $\hat{m}$.
8: **end if**
9: **end for**
10: **if** $\varepsilon > 0$ **then**
11: **if** $\hat{m} > \frac{\nu}{2}$ **then**
12: Output "$\mathbf{c} \cdot \mathbf{k} = 0$".
13: **else**
14: Output "$\mathbf{c} \cdot \mathbf{k} = 1$".
15: **end if**
16: **else**
17: **if** $\hat{m} > \frac{\nu}{2}$ **then**
18: Output "$\mathbf{c} \cdot \mathbf{k} = 1$".
19: **else**
20: Output "$\mathbf{c} \cdot \mathbf{k} = 0$".
21: **end if**
22: **end if**

Algorithm 3.1: Matsui's First Algorithm [202]

Information Extraction about the Key

Let us assume that we make use of a biased linear approximation on a block cipher f for a fixed key $\mathbf{k}$:

$$\mathbf{a} \cdot \mathbf{x} \oplus \mathbf{b} \cdot \mathsf{f}_{\mathbf{k}}(\mathbf{x}) = \mathbf{c} \cdot \mathbf{k} \tag{3.15}$$

with

$$\Pr_{\boldsymbol{X}} \left[\mathbf{a} \cdot \boldsymbol{X} \oplus \mathbf{b} \cdot \mathsf{f}_{\mathbf{k}}(\boldsymbol{X}) = \mathbf{c} \cdot \mathbf{k}\right] = \frac{1}{2} + \varepsilon \tag{3.16}$$

Let us furthermore assume that we make use of a random source generating some $(\mathbf{x}, \mathsf{f}_{\mathbf{k}}(\mathbf{x}))$ pairs. Equivalently, we can assume that we make use of an oracle $\mathbf{\Omega}$ implementing $\mathsf{f}_{\mathbf{k}}(.)$ which is queried by a uniformly distributed random plaintext source (i.e. it is a known-plaintext attack). It is then possible to extract one bit of information about the key using *Matsui's First Algorithm* [202] described in Alg. 3.1. This bit $\mathbf{c} \cdot \mathbf{k}$ of information about the key could be used for instance to speed up an exhaustive key search. A major drawback of Alg. 3.1 is the small amount of extracted information

1: **Input**: an oracle $\mathbf{\Omega}$, a data complexity ν, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, ε.
2: **Output**: a guess about $\mathbf{c} \cdot \mathbf{k}$ and $\mathbf{k}^{(r)}$
3: Initialize $2^{\ell_{r'}}$ counters $\hat{m}_i$ with $1 \leq i \leq 2^{\ell_{r'}}$ to 0.
4: For each candidate $\hat{\mathbf{k}}_i^{(r)}$ with $1 \leq i \leq 2^{\ell_{r'}}$, let $\hat{m}_i$ be the number of uniformly and statistically independent plaintext-ciphertext pairs queried to $\mathbf{\Omega}$ for which the left side of Eq. (3.17) is equal to 0.
5: Let $\hat{m}_{\max} \leftarrow \max_i \hat{m}_i$ and $\hat{m}_{\min} \leftarrow \min_i \hat{m}_i$
6: **if** $|\hat{m}_{\max} - \frac{\nu}{2}| > |\hat{m}_{\min} - \frac{\nu}{2}|$ **then**
7: Output the key candidate corresponding to $\hat{m}_{\max}$ and $\mathbf{c} \cdot \mathbf{k} = 0$ (if $\varepsilon > 0$) or $\mathbf{c} \cdot \mathbf{k} = 1$ (if $\varepsilon < 0$).
8: **else**
9: Output the key candidate corresponding to $\hat{m}_{\min}$ and $\mathbf{c} \cdot \mathbf{k} = 1$ (if $\varepsilon > 0$) or $\mathbf{c} \cdot \mathbf{k} = 0$ (if $\varepsilon < 0$).
10: **end if**

Algorithm 3.2: Matsui's Second Algorithm [202]

about the key. A possible improvement[11] is the following: let us assume that we make use of a linear relation on $r-1$ rounds of a block cipher. The idea consists in *guessing* a subset of the ℓ_r key bits used as the last round subkey and to partially decrypt the known plaintext-ciphertext pairs with these key guesses. Under this scenario, one can rewrite Eq. (3.15) as

$$\mathbf{a} \cdot \mathbf{x} \oplus \mathbf{b} \cdot \left(\mathsf{f}^{(r)^{-1}}(\mathbf{c}, \hat{\mathbf{k}}^{(r)}) \right) = \mathbf{c} \cdot \mathbf{k} \tag{3.17}$$

where $\mathsf{f}^{(r)^{-1}}(\mathbf{c}, \hat{\mathbf{k}}^{(r)})$ denotes the inverse of the last round function fed with a subkey candidate $\hat{\mathbf{k}}^{(r)}$. Let us assume that $\ell_{r'}$ key bits are necessary to recover the effective output bits of the linear relation on $r-1$ rounds. We have then $2^{\ell_{r'}} - 1$ wrong subkey candidates and a single correct one. Intuitively, if we decrypt the last round with the right subkey candidate, we will evaluate the linear approximation properly, and this approximation should be biased as expected. If we decrypt with a wrong subkey candidate, we can see this operation as equivalent to encrypt the ciphertext with a supplementary round, and the linear approximation bias should be rendered far more uniform. This idea is formally expressed as Alg. 3.2 which is named *Matsui's Second Algorithm*. Once the cryptanalyst has a guess for the effective bits of the last round subkey, the other key bits can be recovered using an exhaustive key search. A further improvement which generalize Alg. 3.2, named *key ranking*, was introduced by Matsui in his second paper [203]. Instead of taking the most biased subkey candidate as the right one, and then to

[11] The idea of "peel off" one or more rounds originate from Biham and Shamir differential cryptanalysis [31]; Matsui [202] seems to be the first one to use the idea in a different context; however, the key idea is implicitly present in the paper of Tardy-Corfdir and Gilbert [308].

1: **Input**: an oracle $\mathbf{\Omega}$, a data complexity ν, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, ε.
2: **Output**: a guess about $\mathbf{c} \cdot \mathbf{k}$ and $\mathbf{k}^{(r)}$
3: Initialize $2^{\ell_{r'}}$ counters $\hat{m}_i$ with $1 \leq i \leq 2^{\ell_{r'}}$ to 0.
4: For each candidate $\hat{\mathbf{k}}_i^{(r)}$ with $1 \leq i \leq 2^{\ell_{r'}}$, let $\hat{m}_i$ be the number of uniformly and statistically independent plaintext-ciphertext pairs queried to $\mathbf{\Omega}$ for which the left side of Eq. (3.17) is equal to 0.
5: Rank the $\hat{m}_i$ by decreasing value of $|\hat{m}_i - \frac{\nu}{2}|$ and rename $\hat{\mathbf{k}}_i^{(r)}$ and $\hat{m}_i$ by $\tilde{\mathbf{k}}_i^{(r)}$ and $\tilde{m}_i$, i.e. such that $|\tilde{m}_1 - \frac{\nu}{2}| > |\tilde{m}_2 - \frac{\nu}{2}| > \cdots > |\tilde{m}_{2^{\ell_{r'}}} - \frac{\nu}{2}|$.
6: **for** $i \leftarrow 1$ to $2^{\ell_{r'}}$ **do**
7: Fix the key bits defined by $\tilde{\mathbf{k}}_i^{(r)}$ and look exhaustively for the remaining key bits. Check the key candidates with a few plaintext-ciphertext pairs.
8: **if** the right key is found **then**
9: Break and output "Right key found"
10: **end if**
11: **end for**

Algorithm 3.3: Matsui's Third Algorithm [203]

search exhaustively for the remaining unknown key bits, Matsui proposed to rank the subkey candidates from the most likely to the least likely, and, in this order, to search exhaustively for the remaining unknown key bits until the right key is found. We name this algorithm *Matsui's Third Algorithm* and it is formally described in Alg. 3.3.

3.2.3 Analysis of Matsui's Attacks

In this part, we analyze the success probability and discuss the optimality of the three versions of Matsui's attack as described earlier.

Statistical Analysis of Matsui's First Algorithm

The success probability of Alg. 3.1 has been approximated by Matsui [202]; we give here a more precise analysis and we derive tight bounds.

Theorem 3.2.3 (Success Probability of Alg. 3.1). *Let ν be the number of given known plaintext-ciphertext pairs; let ε be the bias of Eq.* (3.15)*. Then, the success probability of Alg. 3.1 satisfies*

$$\Pr[\text{"Alg. 3.1 is successful"}] \approx \int_{-\infty}^{\frac{2\sqrt{\nu}|\varepsilon|}{\sqrt{(1-4\varepsilon^2)}}} \phi(t)dt \tag{3.18}$$

and

$$\lim_{\nu \to \infty} \Pr[\text{"Alg. 3.1 is successful"}] = 1.$$

Proof. Let us assume without lost of generality that $\varepsilon > 0$, that $\mathbf{c} \cdot \mathbf{k} = 0$ (the other three cases are symmetric situations), and that ν is odd. Let us denote by $\hat{M}$ a discrete random variable modeling the value of the counter $\hat{m}$. Then the success probability of Alg. 3.1 is $\Pr\left[\hat{M} > \frac{\nu}{2}\right]$, thus it satisfies

$$\Pr[\text{“Alg. 3.1 is successful”}] = \sum_{i=\left\lceil \frac{\nu}{2} \right\rceil}^{\nu} \binom{\nu}{i} \left(\frac{1}{2}+\varepsilon\right)^{i} \left(\frac{1}{2}-\varepsilon\right)^{\nu-i}. \quad (3.19)$$

Since ε is small, one can approximate the above probability using the central-limit theorem (see Th. A.2.1) and noting that $\hat{M}$ follows approximately a normal distribution with parameters

$$\mu = \nu\left(\frac{1}{2}+\varepsilon\right) \text{ and } \sigma = \sqrt{\nu\left(\frac{1}{4}-\varepsilon^2\right)}.$$

Hence,

$$\begin{aligned} \Pr[\text{“Alg. 3.1 is successful”}] &\approx \Phi\left(-\frac{2\left(\frac{\nu}{2}-\frac{\nu(1+2\varepsilon)}{2}\right)}{\sqrt{\nu\left(1-4\varepsilon^2\right)}}\right) \\ &= \Phi\left(\frac{2\sqrt{\nu}|\varepsilon|}{\sqrt{(1-4\varepsilon^2)}}\right) \end{aligned}$$

which concludes the proof. □

We note that the above theorem gives an extremely good approximation of the success probability of Alg. 3.1 for any “practical” value of ν and ε in the context of a cryptanalysis and reduce to Matsui's result given in [202] for small ε:

$$\Pr[\text{“Alg. 3.1 is successful”}] \approx \Phi\left(2\sqrt{\nu}|\varepsilon|\right)$$

For estimating the tightness of Eq. (3.18), we need to derive non-asymptotic lower and upper bounds on the probability function of a binomial law; we will use the following approximation (see [262]):

$$\sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i} \approx \Phi(z) - \frac{(1-2p)(z^2-1)\phi(z)}{6\sqrt{np(1-p)}} \quad (3.20)$$

where

$$z = \frac{k+\frac{1}{2}-np}{\sqrt{np(1-p)}} \quad (3.21)$$

and $k = 0, \ldots, n$. We can bound the *maximum absolute error* of this approximation using the following result, due to Raff [270].

Theorem 3.2.4 (Raff). *Let $\eta_{\max}(n,p)$ be the maximum absolute error when the cumulative distribution function of a binomial law with parameters n and p is approximated by Eq.* (3.20) *and Eq.* (3.21)*. Then,*

$$\eta_{\max}(n,p) \leq \frac{0.056}{\sqrt{np(1-p)}} \tag{3.22}$$

The following bounds are then a straightforward application of Eq. (3.20), Eq. (3.21) and of Th. 3.2.4 to our scenario.

Theorem 3.2.5. *Let ν be the (even) number of given known plaintext-ciphertext pairs; let ε be the bias of Eq.* (3.15)*. Then, the success probability of Alg. 3.1 satisfies*

$$\left| \Pr[\textit{“Alg. 3.1 is succ.”}] - \Phi\left(\frac{\varepsilon\sqrt{\nu}}{\sigma}\right) - \frac{\varepsilon(\nu\varepsilon^2-\sigma^2)\phi\left(\frac{\varepsilon\sqrt{\nu}}{\sigma}\right)}{6\sigma^3\sqrt{\nu}} \right| \leq \frac{0.056}{\sigma\sqrt{\nu}} \tag{3.23}$$

where $\sigma = \sqrt{\frac{1}{4}-\varepsilon^2}$.

Proof. Without loss of generality, we assume that $\varepsilon > 0$. According to Eq. (3.19), we have

$$\begin{aligned} \Pr[\text{“Alg. 3.1 is successful”}] &= \sum_{i=\frac{\nu}{2}-1}^{\nu} \binom{\nu}{i} \left(\frac{1}{2}+\varepsilon\right)^i \left(\frac{1}{2}-\varepsilon\right)^{\nu-i} \\ &= 1 - \sum_{i=0}^{\frac{\nu}{2}} \binom{\nu}{i} \left(\frac{1}{2}+\varepsilon\right)^i \left(\frac{1}{2}-\varepsilon\right)^{\nu-i} \end{aligned}$$

We substitute $k = \frac{\nu}{2}$ and $p = \frac{1}{2}+\varepsilon$ into Eq. (3.20), we note that $\Phi(-z) = 1-\Phi(z)$ and that $\phi(-z) = \phi(z)$, we write $\sigma = \sqrt{\frac{1}{4}-\varepsilon^2}$, and this results in

$$\Pr[\text{“Alg. 3.1 is successful”}] \approx \Phi\left(\frac{\varepsilon\sqrt{\nu}}{\sigma}\right) + \frac{\varepsilon(\nu\varepsilon^2-\sigma^2)\phi\left(\frac{\varepsilon\sqrt{\nu}}{\sigma}\right)}{6\sigma^3\sqrt{\nu}}. \tag{3.24}$$

The theorem follows from the straightforward application of Raff's bounds given in Th. 3.2.4 to Eq. (3.24). □

Fig. 3.1 plots the success probability of Alg. 3.1 in terms of $c = \nu\varepsilon^2$. Interestingly, we can note that Th. 3.2.3 implicitly assumes that the absolute bias $|\varepsilon|$ is independent of the key value $\mathbf{k}$ actually implemented by $\mathbf{\Omega}$. This implicit assumption was recognized early by Nyberg [250] and named *fixed-key equivalence hypothesis* by Harpes et al. [125].

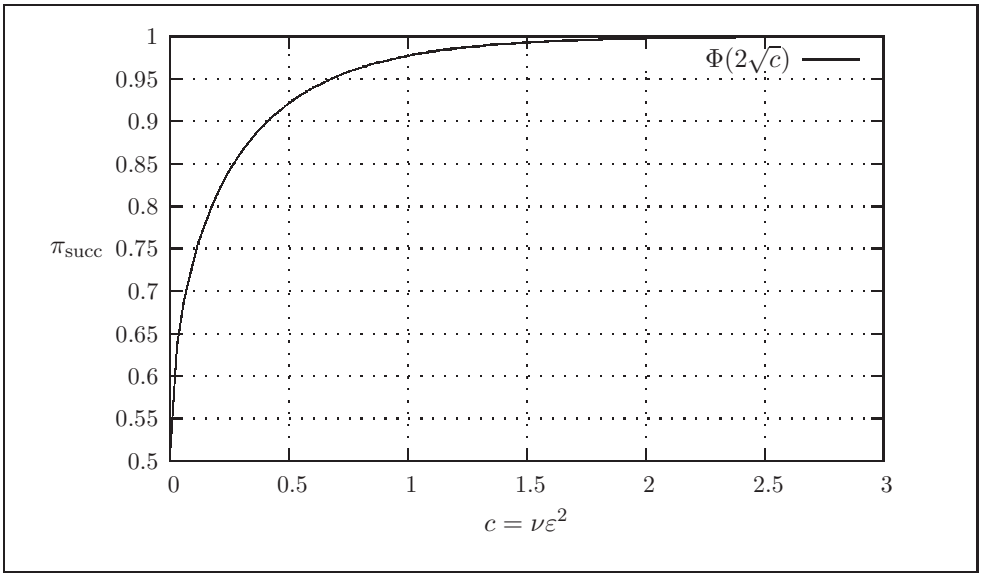

Figure 3.1: Success probability of Alg. 3.1

Assumption 3.2.1 (Fixed-Key Equivalence Hypothesis). *The absolute bias $|\varepsilon|$ of a linear expression as defined in Eq.* (3.15) *and Eq.* (3.16) *is independent of the key value* $\mathbf{k}$.

In other words, the above assumption states that a linear expression approximates the cipher in the same way for all key values, i.e. that $|\varepsilon|$ is not key-dependent; this assumption holds well in the case of DES, but not in the case of RC5 and RC6 [46, 290, 291], for instance, because of the presence of strong key-dependent components in the definition of these ciphers.

Let us now consider the statistical decision process on which Alg. 3.1 is built. For this, we construct an optimal statistical test to decide whether $\mathbf{c} \cdot \mathbf{k} = 0$ or $\mathbf{c} \cdot \mathbf{k} = 1$ using the Neyman-Pearson approach described in §3.1. We assume that we make use of a linear approximation

$$\mathbf{a} \cdot \mathbf{x} \oplus \mathbf{b} \cdot \mathsf{f}_{\mathbf{k}}(\mathbf{x}) = \mathbf{c} \cdot \mathbf{k}$$

with

$$\Pr_{\boldsymbol{X}}\left[\mathbf{a} \cdot \boldsymbol{X} \oplus \mathbf{b} \cdot \mathsf{f}_{\mathbf{k}}(\boldsymbol{X}) = \mathbf{c} \cdot \mathbf{k}\right] = \frac{1}{2} + \varepsilon \qquad \text{with } \varepsilon \neq 0$$

and that $\varepsilon > 0$ (the case $\varepsilon < 0$ leads to similar considerations). We assign to the null hypothesis $\mathcal{H}_0$ the probability distribution D_0

$$\mathsf{D}_0 : \begin{cases} \Pr[\mathbf{a} \cdot \boldsymbol{X} \oplus \mathbf{b} \cdot \mathsf{f}_{\mathbf{k}}(\boldsymbol{X}) = 0] = \frac{1}{2} + \varepsilon \\ \Pr[\mathbf{a} \cdot \boldsymbol{X} \oplus \mathbf{b} \cdot \mathsf{f}_{\mathbf{k}}(\boldsymbol{X}) = 1] = \frac{1}{2} - \varepsilon \end{cases}$$

and to the alternative hypothesis $\mathcal{H}_1$ the probability distribution D_1

$$\mathsf{D}_1 : \begin{cases} \Pr[\mathbf{a} \cdot \boldsymbol{X} \oplus \mathbf{b} \cdot \mathsf{f}_{\mathbf{k}}(\boldsymbol{X}) = 0] = \frac{1}{2} - \varepsilon \\ \Pr[\mathbf{a} \cdot \boldsymbol{X} \oplus \mathbf{b} \cdot \mathsf{f}_{\mathbf{k}}(\boldsymbol{X}) = 1] = \frac{1}{2} + \varepsilon \end{cases}$$

Note that $\mathcal{H}_0$ is equivalent to $\mathbf{c}\cdot\mathbf{k} = \mathbf{0}$ and that $\mathcal{H}_0$ is equivalent to $\mathbf{c}\cdot\mathbf{k} = \mathbf{1}$, since we know the sign of ε. As a second step, we build the likelihood-ratio corresponding to this decision problem. Let us denote by $\hat{E}_i$ the event

$$\hat{E}_i = 1_{\mathbf{a}\cdot\boldsymbol{X}_i \oplus \mathbf{b}\cdot \mathsf{f}_{\mathbf{k}}(\boldsymbol{X}_i)=0}$$

for the i-th query $\boldsymbol{X}_i$ to the oracle $\boldsymbol{\Omega}$. We have then

$$\begin{aligned}
\mathsf{M}_{\mathrm{lr}}(\hat{E}_1,\ldots,\hat{E}_\nu) &= \prod_{i=1}^{\nu} \frac{\left(\frac{1}{2}+\varepsilon\right)^{\hat{E}_i}\left(\frac{1}{2}-\varepsilon\right)^{1-\hat{E}_i}}{\left(\frac{1}{2}-\varepsilon\right)^{\hat{E}_i}\left(\frac{1}{2}+\varepsilon\right)^{1-\hat{E}_i}} \\
&= \frac{\left(\frac{1}{2}+\varepsilon\right)^{\hat{m}}\left(\frac{1}{2}-\varepsilon\right)^{\nu-\hat{m}}}{\left(\frac{1}{2}-\varepsilon\right)^{\hat{m}}\left(\frac{1}{2}+\varepsilon\right)^{\nu-\hat{m}}} \\
&= \frac{\left(\frac{1}{2}+\varepsilon\right)^{2\hat{m}}\left(\frac{1}{2}-\varepsilon\right)^{\nu}}{\left(\frac{1}{2}-\varepsilon\right)^{2\hat{m}}\left(\frac{1}{2}+\varepsilon\right)^{\nu}}
\end{aligned}$$

where $\hat{m} = \sum_i \hat{E}_i$. By taking the logarithm of $\mathsf{M}_{\mathrm{lr}}(\hat{E}_1,\ldots,\hat{E}_\nu)$, one can rewrite the above expression as

$$\log\left(\mathsf{M}_{\mathrm{lr}}(\hat{E}_1,\ldots,\hat{E}_\nu)\right) = (\nu - 2\hat{m}) \underbrace{\left(\log\left(\frac{1}{2}-\varepsilon\right) - \log\left(\frac{1}{2}+\varepsilon\right)\right)}_{<0} \quad (3.25)$$

Comparing $\mathsf{M}_{\mathrm{lr}}(\hat{E}_1,\ldots,\hat{E}_\nu)$ to 1 is thus equivalent to comparing $\hat{m}$ to $\frac{\nu}{2}$, which is the decision rule of line 11 in Alg. 3.1. Hence, we proved the following theorem.

Theorem 3.2.6. *For a fixed number ν of data queried to the oracle $\boldsymbol{\Omega}$, Matsui's First Algorithm (Alg. 3.1) is optimal in the sense that it maximizes the success probability over all algorithms based on the sample bit*

$$\mathbf{a}\cdot\boldsymbol{X}_i \oplus \mathbf{b}\cdot\mathsf{f}_{\mathbf{k}}(\boldsymbol{X}_i).$$

We can interpret the above results as follows: based on the statistical information gathered by Matsui's First Algorithm, and on the *a priori* knowledge about the sign of ε, one *cannot* define a decision rule on samples of $\mathbf{c}\cdot\mathbf{k}$ which performs better than Eq. (3.25); note that we take into account infinitely powerful adversaries as well.

Success Probability of Matsui's Second and Third Algorithms

As explained in page 69, one expects that during a key-ranking procedure, the right key will have a different behavior than all the other wrong keys. This intuition is summarized in the following assumption, named *hypothesis of wrong-key randomization*[12] by Harpes et al. [125].

[12]Interestingly, one can note that this assumption is (implicitly) stated in terms of a likelihood-ratio.

Assumption 3.2.2 (Wrong-Key Randomization Hypothesis). *For any linear relation Λ similar to 3.17 for which its bias is large for virtually all values $\mathbf{k}^{(1)}, \ldots, \mathbf{k}^{(r-1)}$ of the round subkeys, the following is true: for virtually all possible round subkeys $\mathbf{k}^{(1)}, \ldots, \mathbf{k}^{(r-1)}$, for all possible guesses $\boldsymbol{K}^{(r)}$ of the last round subkey,*

$$\frac{\left|\Pr\left[\Lambda \text{ holds } | \boldsymbol{K}^{(r)} = \hat{\mathbf{k}}_r^{(r)}\right] - \frac{1}{2}\right|}{\left|\Pr\left[\Lambda \text{ holds } | \boldsymbol{K}^{(r)} = \hat{\mathbf{k}}_w^{(r)}\right] - \frac{1}{2}\right|} \gg 1$$

where $\hat{\mathbf{k}}_w^{(r)}$ denotes a wrong guess and $\hat{\mathbf{k}}_r^{(r)}$ denotes the right guess.

We give now a new modelization of the success probability of Matsui's Second and Third Algorithms. For this, we will first need the definition of the *incomplete beta function of order* (a, b).

Definition 3.2.3 (Incomplete Beta Function). *The* incomplete beta function of order (a, b) *is defined by*

$$\mathsf{B}_{a,b}(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1}\, dt \tag{3.26}$$

where $\Gamma(x)$ denotes the gamma function[13].

As a first step, let us consider the following scenario: let $M_1, M_2, \ldots, M_n$ be n independent and identically distributed continuous random variables having f_M and F_M as density and distribution functions, respectively. We sort the values of $M_1, M_2, \ldots, M_n$ in strictly[14] increasing order and denote them by $\tilde{M}_1 < \tilde{M}_2 < \ldots < \tilde{M}_n$. The distribution function $\mathsf{F}_{\tilde{M}_i}$ of the i-th smallest random variable $\tilde{M}_i$ is given by the following lemma whose proof can be found in [273, page 205].

Lemma 3.2.3. *The distribution function of the i-th smallest among n iid random variables is*

$$\mathsf{F}_{\tilde{M}_i}(x) = \mathsf{B}_{i,n-i+1}\left(\mathsf{F}_M(x)\right)$$

where $\mathsf{B}_{a,b}(x)$ is the incomplete beta function of order (a, b).

By using the previous result and the independence between the involved random variables, we prove now the following theorem.

Theorem 3.2.7. *Let $M_1, \ldots, M_n$ be n iid random variables having f_M and and F_M as probability density and distribution functions, respectively. Let M^* be a random variable statistically independent of the M_i. Let R denote a random variable modeling the rank of M^* when the $n+1$ random variables*

[13] The gamma function is related to the factorial by $\Gamma(n) = (n-1)!$.

[14] The probability that equal values occur is 0.

$\{M_i : 1 \leq i \leq n\} \cup \{M^*\}$ are sorted in non-increasing order. For $r \in \mathbb{N}, 1 \leq r \leq n$, the distribution function of R is equal to

$$\Pr[R \leq r] = \int_{-\infty}^{+\infty} \mathsf{B}_{n+1-r,r}(\mathsf{F}_M(x))\mathsf{f}_{M^*}(x)dx$$

and

$$\mathrm{E}[R] = 1 + n\left(1 - \int_{-\infty}^{+\infty} \mathsf{F}_M(x)\mathsf{f}_{M^*}(x)dx\right)$$

where $\mathsf{B}_{a,b}(x)$ is the incomplete beta function of order (a, b).

Proof. Let us denote the $n+1$ sorted random variables by

$$\tilde{M}_1 > \cdots > \tilde{M}_i > M^* > \tilde{M}_{i+1} > \cdots > \tilde{M}_n$$

We can compute the distribution function $\mathsf{F}_R(x)$ of R as follows:

$$\begin{aligned}
\Pr[R \leq r] &= \Pr\left[\tilde{M}_r < M^*\right] \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{y} \mathsf{f}_{\tilde{M}_r}(x)\mathsf{f}_{M^*}(y)\,dx\,dy \\
&= \int_{-\infty}^{+\infty} \mathsf{B}_{n+1-r,r}\left(\mathsf{F}_M(y)\right)\mathsf{f}_{M^*}(y)\,dy
\end{aligned}$$

where the last equality follows from Lem. 3.2.3. By definition of the expectation of a random variable, we have

$$\begin{aligned}
\mathrm{E}[R] &= \sum_{r=1}^{n+1} r \cdot \Pr[R = r] \\
&= \Pr[R = 1] + \sum_{r=2}^{n+1} r\left(\Pr[R \leq r] - \Pr[R \leq r-1]\right) \\
&= n + 1 - \sum_{r=1}^{n} \Pr[R \leq r]
\end{aligned}$$

where

$$\begin{aligned}
\sum_{r=1}^{n} \Pr[R \leq r] &= \sum_{r=1}^{n} \int_{-\infty}^{+\infty} \mathsf{B}_{n+1-r,r}\left(\mathsf{F}_M(y)\right)\mathsf{f}_{M^*}(y)\,dy \\
&= \int_{-\infty}^{+\infty} \mathsf{f}_{M^*}(y) \sum_{r=1}^{n} \mathsf{B}_{n+1-r,r}\left(\mathsf{F}_M(y)\right)\,dy
\end{aligned}$$

It is easy to see that

$$\begin{aligned}\sum_{r=1}^{n} \mathsf{B}_{n+1-r,r}\left(\mathsf{F}_M(y)\right) &= n\int_0^{\mathsf{F}_M(y)} \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-1-i}\, dt \\ &= n\int_0^{\mathsf{F}_M(y)} dt = n\mathsf{F}_M(y)\end{aligned}$$

and we can thus conclude with

$$\begin{aligned}\mathrm{E}\left[R\right] &= n+1-\int_{-\infty}^{+\infty} \mathsf{f}_{M^*}(y) \sum_{r=1}^{n} \mathsf{B}_{n+1-r,r}\left(\mathsf{F}_M(y)\right)\, dy \\ &= n+1-n\int_{-\infty}^{+\infty} \mathsf{f}_{M^*}(y)\mathsf{F}_M(y)\, dy \\ &= 1+n\left(1-\int_{-\infty}^{+\infty} \mathsf{f}_{M^*}(y)\mathsf{F}_M(y)\, dy\right)\end{aligned}$$

which proves the theorem. □

We compute now the probability distributions of the random variables $\hat{M}_i$ modeling the counters $\hat{m}_i$ used in Alg. 3.2 and Alg. 3.3. For this purpose, we denote by $\hat{M}_i$, with $1 \leq i \leq 2^{\ell_{r'}} - 1$, the (ordered) counters corresponding to wrong subkey candidates, and by $\hat{M}^*$ the counter corresponding to the right subkey candidate. We first take an assumption which is essentially of heuristic nature.

Assumption 3.2.3. *The $2^{\ell_{r'}}$ random variables $\hat{M}_i$, with $1 \leq i \leq 2^{\ell_{r'}} - 1$, and $\hat{M}^*$ are statistically independent.*

The two following assumptions are motivated, on the one hand, by the accuracy of the approximation by a normal distribution of a binomial distribution $\mathsf{D}_{\mathrm{Bin}}^{(n,p)}$ when n is large and $p \approx \frac{1}{2}$, and on the other hand, by both Ass. 3.2.1 and Ass. 3.2.2.

Assumption 3.2.4. *The random variables $\frac{2\hat{M}_i - \nu}{\sqrt{\nu}}$, with $1 \leq i \leq 2^{\ell_{r'}} - 1$ are distributed according to the normal distribution D_ϕ.*

Assumption 3.2.5. *The random variable $\frac{2\hat{M}^* - \nu(1-2\varepsilon)}{\sqrt{\nu}}$ is distributed according to the normal distribution D_ϕ.*

We note that Matsui's Third Algorithm considers the absolute deviation from the counters to $\frac{\nu}{2}$ (see line 5 of Alg. 3.3). To take this fact into account, as well as the three above assumptions, we define $\hat{B}$ to be a random variable

distributed according to the probability distribution of the absolute bias of a counter corresponding to a *wrong* subkey candidate

$$\hat{B} = \left| \frac{\hat{M}_i}{\nu} - \frac{1}{2} \right|$$

as well as

$$\hat{B}^* = \left| \frac{\hat{M}^*}{\nu} - \frac{1}{2} \right|.$$

where $\hat{B}^*$ is a random variable distributed according to the probability distribution of the absolute bias of a counter corresponding to the *right* subkey candidate. Note that the probability distribution of $Y = |X - a|$, provided X is normally distributed and a is a real constant, is often called the *folded normal distribution*. The computation of the probability distribution of $\hat{B}$ and $\hat{B}^*$ can be easily derived using the following lemma.

Lemma 3.2.4. *Let X be a continuous random variable distributed according to a normal distribution with parameters μ and σ; let $a \in \mathbb{R}$ be a real constant. The probability density function of $Y = |X - a|$, with $a \leq \mu$, is*

$$\mathsf{f}_Y(y) = \phi\left(\frac{y - \mu + a}{\sigma}\right) + \phi\left(\frac{a - y - \mu}{\sigma}\right)$$

for $0 \leq y \leq +\infty$ and $\mathsf{f}_Y = 0$ otherwise.

Proof. The cumulative distribution function of the random variable Y can be computed as

$$\begin{aligned} \mathsf{F}_Y(y) &= \Pr_Y[Y \leq y] = \Pr_X[|X - a| \leq y] \\ &= \Pr_X[-y + a \leq X \leq y + a] \\ &= \Phi\left(\frac{y - \mu + a}{\sigma}\right) - \Phi\left(\frac{a - y - \mu}{\sigma}\right) \end{aligned}$$

As the cumulative distribution function of the normal distribution is absolutely continuous, which ensures that it has a probability density function with respect to Lebesgue measure, one can write

$$\begin{aligned} \mathsf{f}_Y(y) &= \frac{\partial\left(\Phi\left(\frac{y+a-\mu}{\sigma}\right) - \Phi\left(\frac{a-y-\mu}{\sigma}\right)\right)}{\partial y} \\ &= \phi\left(\frac{y + a - \mu}{\sigma}\right) + \phi\left(\frac{a - y - \mu}{\sigma}\right) \end{aligned}$$

which concludes the proof. □

Under Ass. 3.2.3, Ass. 3.2.4, Ass. 3.2.5, using Lem. 3.2.4, and assuming that the linear approximation is unbalanced in case of a wrong key candidate (in the spirit of Ass. 3.2.2), we can compute the probability density function of $\hat{B}$ and $\hat{B}^*$ for the case of a linear cryptanalysis applying Matsui's Second and Third Algorithms as follows: $\hat{B}^*$ is distributed according to a folded normal law with[15] $a = \frac{1}{2}$, $\mu^* = \frac{1}{2} + \varepsilon$, and $\sigma^2 = \frac{1}{4\nu}$

$$\mathsf{f}_{\hat{B}^*}(x) = \phi\left(2(x-\varepsilon)\sqrt{\nu}\right) + \phi\left(2(x+\varepsilon)\sqrt{\nu}\right) \tag{3.27}$$

while $\hat{B}$ is distributed according to a folded normal law with $a = \frac{1}{2}$, $\mu = \frac{1}{2}$, and $\sigma^2 = \frac{1}{4\nu}$

$$\mathsf{f}_{\hat{B}}(x) = 2\phi\left(2x\sqrt{\nu}\right). \tag{3.28}$$

We get thus the following result, which is a straightforward application of Th. 3.2.7 in this setting.

Corollary 3.2.1. *Under assumptions 3.2.2, 3.2.3, 3.2.4, and 3.2.5, the probability distribution of the rank R of the right subkey candidate when using Alg. 3.3 is equal to*

$$\Pr\left[R \leq r\right] = \int_{-\infty}^{+\infty} \mathsf{B}_{n+1-r,r}(\mathsf{F}_{\hat{B}}(x))\mathsf{f}_{\hat{B}^*}(x)dx$$

and

$$\mathrm{E}\left[R\right] = 1 + n\left(1 - \int_{-\infty}^{+\infty} \mathsf{f}_{\hat{B}^*}(x)\mathsf{F}_{\hat{B}}(x)dx\right)$$

where $\mathsf{B}_{a,b}(x)$ is the incomplete beta function of order (a,b), $n = 2^{\ell_{r'}} - 1$, $\mathsf{f}_{\hat{B}^*}$ and $\mathsf{f}_{\hat{B}}$ are respectively defined in Eq. (3.27) and Eq. (3.28), and $\mathsf{F}_{\hat{B}^*}$ and $\mathsf{F}_{\hat{B}}$ are their respective cumulative distribution functions.

It is worth noticing that our results stated in Cor. 3.2.1 have been reworked by Selçuk and Bıçak [289] in a manner allowing a simpler numerical evaluation. Their result was derived with help of the following classical application of the central limit theorem (a proof thereof can be found in [273, page 490]).

Theorem 3.2.8. *Let $W_1, \ldots, W_n$ be n independent and identically distributed random variable with an absolutely continuous distribution function $\mathsf{F}_W(x)$. Suppose that the probability density function $\mathsf{f}_W(x)$ is continuous and positive on the interval $[a,b[$. If $0 < \mathsf{F}_W(a) < q < \mathsf{F}_W(b) < 1$, and if $\mathsf{k}(n)$ is a sequence of integers such that*

$$\lim_{n\to+\infty} \sqrt{n}\left|\frac{\mathsf{k}(n)}{n} - q\right| = 0$$

[15] Note that we do not need to know the sign of ε anymore when working with a folded normal distribution.

and if $\tilde{W}_i$ denotes the i-th order statistic of the sample $W_1, \ldots, W_n$, then $\tilde{W}_{\mathsf{k}(n)}$ is in the limit normally distributed:

$$\lim_{n \to +\infty} \Pr\left[\frac{\tilde{W}_{\mathsf{k}(n)} - \mu}{\sigma} < \frac{x}{\sqrt{n}}\right] = \Phi(x)$$

where

$$\mu = \mathsf{F}_W^{-1}(q) \text{ and } \sigma = \frac{1}{\mathsf{f}_W(\mu)}\sqrt{\mu(1-\mu)}$$

Taking $\mathsf{k}(n) = \lfloor qn \rfloor + 1$, this theorem states that the empirical sample quantile of order q of n elements is for sufficiently large n approximately normally distributed with mean $\mu_q = \mathsf{F}_W^{-1}(q)$ and standard deviation $\sigma_q = \frac{1}{\mathsf{f}_W(\mu_q)}\sqrt{\frac{q(1-q)}{n}}$. Selçuk and Bıçak obtained the following theorem holding under the same assumptions than those stated in Cor. 3.2.1.

Theorem 3.2.9 (Selçuk and Bıçak [289]). *Let ν be the number of given known plaintext-ciphertext pairs; let us assume that Matsui's Second Algorithm Alg. 3.2 furnishes $2^{\ell_{r'}}$ subkey candidates; let finally ε be the bias of Eq.* (3.15)*. Then, the success probability of Alg. 3.2 is approximately equal to*

$$\Pr[R \leq r] = \Phi\left(2\sqrt{\nu}|\varepsilon| - \Phi^{-1}\left(1 - \frac{r}{2^{\ell_{r'}+1}}\right)\right).$$

Fig. 3.2 expresses the success probability of Matsui's Second Algorithm in terms of the number of known plaintext-ciphertext pairs at disposal of an adversary in the context of a linear cryptanalysis of 16-round DES using a single best approximation on 14- and 15-rounds, respectively, as stated in Th. 3.2.1 and Th. 3.2.2. An interesting phenomenon occurs here: on the one hand, the best approximation on 15-rounds is less biased than the one on 14 rounds, but since the latter involves 12 unknown key bits, while the 15-rounds approximation involves only 6 bits, the success probability is on the other hand essentially better for the approximation on 15 rounds, as there is less noise from the wrong subkey candidates. However, note that the exhaustive search of the remaining unknown key bits is 2^6 times more costly.

Fig. 3.3 plots the cumulative distribution function of the rank r of the right subkey candidate in the same context, using Matsui's Third Algorithm and a single best approximation on 14-rounds for various amounts ν of known plaintext-ciphertext at disposal of the adversary.

3.2.4 Improvement of Matsui's Attack

Matsui has introduced [202, 203] the idea of taking a "soft decision" about the value of the right subkey by ranking subkey candidates by "maximum likelihood". We would like now to look more closely at this concept.

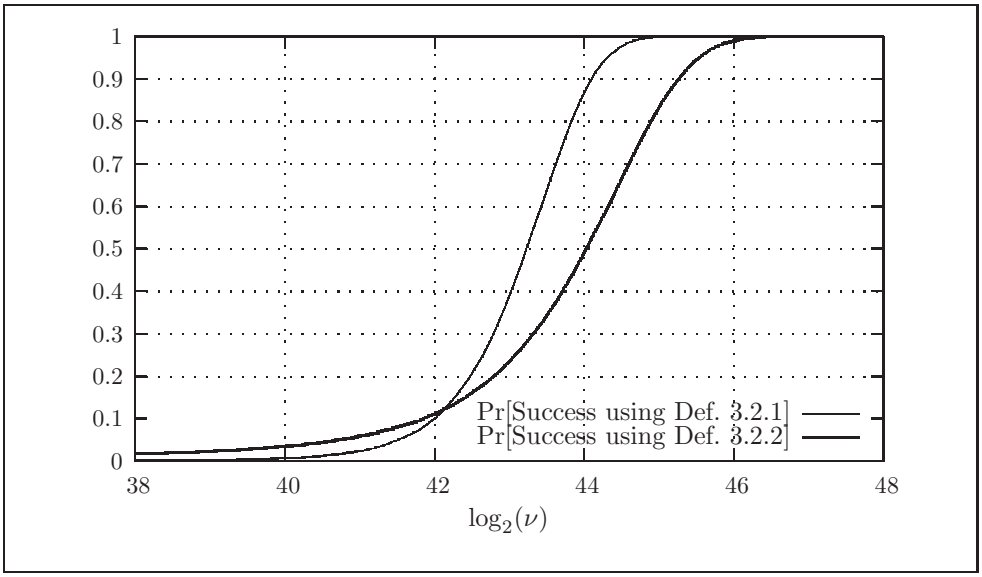

Figure 3.2: Success probability of Matsui's Second Algorithm (Alg. 3.2).

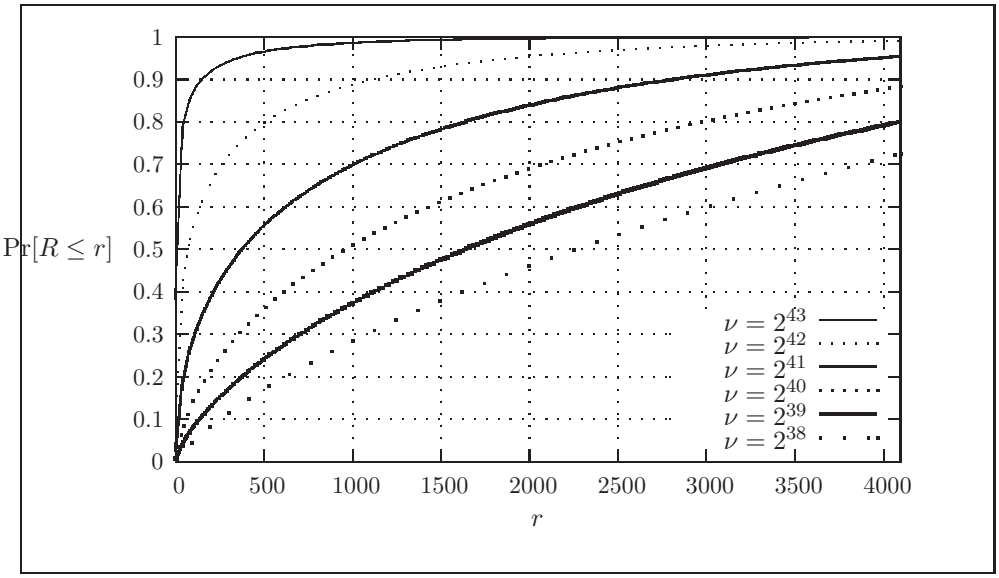

Figure 3.3: Success probability of Matsui's Third Algorithm (Alg. 3.3).

1. **Counting Phase**: Collect several random samples $s_j = \mathsf{f}_2(p_j, c_j)$, for $j = 1, \ldots, \nu$ and count all occurrences of all the possible values of the s_j's in $|\mathcal{S}|$ counters denoted $\hat{m}_i$ with $1 \leq i \leq |\mathcal{S}|$.
2. **Analysis Phase**: For each of the subkey candidates k_i, $1 \leq i \leq 2^{\ell_{r'}}$, count all the occurrences in all $x_i = \mathsf{f}_3(k_i, s_j)$ and give it a mark μ_{k_i} using the statistic $\Sigma(x_1, \ldots, x_\nu)$.
3. **Sorting Phase**: Sort all the candidates k_i using their mark μ_{k_i}. This list of sorted candidates is denoted $\mathcal{U}$.
4. **Searching Phase**: Exhaustively try all keys following the sorted list of all the subkey candidates.

Figure 3.4: Structure of a statistical cryptanalysis

Optimal Key-Ranking Procedure

First of all, we define what we mean by an "optimal key ranking procedure". As our considerations apply to any attack of statistical nature, we first recall the model of statistical cryptanalysis we will adopt. Let $\mathcal{P}$, $\mathcal{C}$ and $\mathcal{K}$ be the plaintext, ciphertext and key space, respectively. A statistical cryptanalysis uses three functions, denoted f_1, f_2 and f_3 which have the following role:

- $\mathsf{f}_1 : \mathcal{K} \to \mathcal{K}'$ is a function which extract the useful information of the key for the cryptanalysis. We assume here that $|\mathcal{K}'| = 2^{\ell_{r'}}$.
- $\mathsf{f}_2 : \mathcal{P} \times \mathcal{C} \to \mathcal{S}$, where $\mathcal{S}$ is called the *sample space*, extracts the useful information about the plaintext and ciphertext spaces for the attack.
- $\mathsf{f}_3 : \mathcal{K}' \times \mathcal{S} \to \mathcal{Q}$, where $\mathcal{Q}$ is a space summarizing information depending on intermediate results in the encryption.

In order to be efficient, a statistical cryptanalysis should fulfill the following conditions: the information $x = \mathsf{f}_3(k', s)$, where $k' \in \mathcal{K}'$, $s \in \mathcal{S}$ and $x \in \mathcal{Q}$, should be computable with a *small* piece of information on $(p, c) \in \mathcal{P} \times \mathcal{C}$ and $k \in \mathcal{K}$ (namely, s and k'); furthermore, the information $x = \mathsf{f}_3(s, k'_r)$ should be statistically distinguishable from $x' = \mathsf{f}_3(s, k'_w)$, where k'_r and k'_w is the information given by the right key and a wrong key, respectively. The main idea of the attack consists in assuming that we can distinguish the right key from wrong key with help of a statistical measurement Σ on the observed distribution of the x_i's. The attack is summarized in Fig. 3.4. The *data complexity* is then defined to be the number n of known plaintext-ciphertext pairs needed in step 1, while the *computational complexity* is usually[16] the number of operations in the last phase of the attack.

[16]The complexity of steps 2 and 3 is usually negligible, but it may not be the case in all situations.

The key ranking procedure corresponds to step 3 in Fig. 3.4: instead of returning the subkey $k'_{\max}$ possessing the highest mark $\mu_{k'_{\max}} = \max_i \mu_{k'_i}$ out of $|\mathcal{K}'|$ subkey candidates, the idea is to return a *sorted list* $\mathcal{U}$ containing key candidates ranked by likelihood and to search exhaustively for the remaining un-attacked bits in this order. Obviously, two central points in a statistical cryptanalysis are the definition of the statistic Σ and of the mark μ which has to be assigned to a subkey candidate. The first issue is the very essence of the attack: the cryptanalyst must find a "statistical weakness" in the cipher. In this part, we will be interested in the second issue, namely how to define an optimal key-ranking procedure, which is formally defined in Def. 3.2.4.

Definition 3.2.4 (Optimal Key Ranking Procedure). *Let a set of (sub-)key candidates be denoted $\mathcal{K}' = \{k_1, \ldots, k_{2^{\ell_{r'}}}\}$ and let R^* be a random variable which denotes the right one. Let $\mathcal{S}$ be a set of random measurements. An optimal key-ranking procedure is a function which maps $\mathcal{S}$ to a permutation ξ on $\{1, \ldots, 2^{\ell_{r'}}\}$ such that*

$$\sum_{i=1}^{2^{\ell_{r'}}} i \Pr[k'_{\xi(i)} = R^*] \tag{3.29}$$

is minimal.

In other words, an optimal key ranking procedure orders the subkey candidates in such a manner that the cost of finding the distinguished subkey (i.e. the right one) is minimal; here, we assume implicitly here that the cost of testing a candidate for "distinguishness" is the same for every candidate. The following simple lemma claims that an optimal key ranking procedure in the spirit of Def. 3.2.4 must order the (sub-)key candidates by decreasing probabilities of finding the distinguished candidate.

Lemma 3.2.5. *Let $\mathcal{K}' = \{k'_i : 1 \leq i \leq 2^{\ell_{r'}}\}$ be a set of (sub-)key candidates proposed by an attack, where one of the subkeys is a distinguished one, denoted by R^*. A permutation ξ on $\{1, \ldots, 2^{\ell_{r'}}\}$ which orders the (sub-)key candidates such that*

$$\Pr\left[R^* = \tilde{k}_1\right] \geq \ldots \geq \Pr\left[R^* = \tilde{k}_{2^{\ell_{r'}}}\right] \tag{3.30}$$

where $\tilde{k}_{\xi(j)} = k'_j$ is an optimal key ranking procedure.

Proof. Let us assume that there exists a permutation ξ *not* satisfying the condition stated in Eq. (3.30) which minimizes Eq. (3.29). Let us denote $\pi_i = \Pr\left[R^* = \tilde{k}_i\right]$ the probability that the right subkey is at rank i. Without

loss of generality, we assume that ξ implies that $\pi_i < \pi_j$ for some $i < j$. Then, by assumption, we must have

$$\sum_{u:u\notin\{i,j\}} u \cdot \pi_u + i \cdot \pi_i + j \cdot \pi_j \leq \sum_{u:u\notin\{i,j\}} u \cdot \pi_u + i \cdot \pi_j + j \cdot \pi_i$$

which is equivalent to

$$i \cdot \pi_i + j \cdot \pi_j \leq i \cdot \pi_j + j \cdot \pi_i \iff i(\pi_i - \pi_j) \leq j(\pi_i - \pi_j)$$

As $\pi_i - \pi_j \leq 0$ by assumption, the above equation implies that $i \geq j$, which is a contradiction, and the theorem follows by induction. □

Application to DES

Let us now apply optimal key-ranking procedure to the context of a linear cryptanalysis implemented as Matsui's Third Algorithm (Alg. 3.3), the probabilistic scenario is the one described in Th. 3.2.7 under Ass. 3.2.2, Ass. 3.2.3, Ass. 3.2.4, and Ass. 3.2.5: we have $2^{\ell_{r'}} - 1$ iid random variables $M_1, \ldots, M_n$ sharing f_M and F_M as probability density and distribution functions, respectively, which model the value of the absolute bias of the counters corresponding to wrong subkey candidates. Additionally, there is a random variable M^* distributed according f_{M^*}, statistically independent of the M_i's, which models the value of the absolute bias of the counter corresponding to the right subkey candidate. Let us assume that a subkey candidate is bound to each counter such that we can use the counters to denote the subkey candidates. Let us sort now sort the sample values $\hat{m}_i$ by decreasing value of $|\frac{\hat{m}_i}{\nu} - \frac{1}{2}|$, where ν is the number of samples, and rename them $\tilde{m}_i$, i.e.

$$\tilde{m}_1 \geq \tilde{m}_2 \geq \ldots \geq \tilde{m}_{2^{\ell_{r'}}}. \tag{3.31}$$

Let us now consider the two following hypotheses (out of the $2^{\ell_{r'}}$ possible ones) for the rank of the right subkey candidate in the sorted sequence given in Eq. (3.31): either the right subkey candidate has rank ψ, or rank ψ', with $1 \leq \psi, \psi' \leq 2^{\ell_{r'}}$ and $\psi \neq \psi'$. Then, the condition expressed in Th. 3.2.5 can be rewritten as

$$\mathsf{f}_{M^*}(\tilde{m}_\psi) \prod_{\substack{1\leq i\leq 2^{\ell_{r'}} \\ i\neq\psi}} \mathsf{f}_M(\tilde{m}_i) \quad \geq \quad \mathsf{f}_{M^*}(\tilde{m}_{\psi'}) \prod_{\substack{1\leq i\leq 2^{\ell_{r'}} \\ i\neq\psi'}} \mathsf{f}_M(\tilde{m}_i) \tag{3.32}$$

which is, after some algebraic manipulations, equivalent to

$$\frac{\mathsf{f}_{M^*}(\tilde{m}_\psi)}{\mathsf{f}_M(\tilde{m}_\psi)} \geq \frac{\mathsf{f}_{M^*}(\tilde{m}_{\psi'})}{\mathsf{f}_M(\tilde{m}_{\psi'})}.$$

This immediately drives us to rank the candidates by *decreasing likelihood-ratio values*: the greater the value, the more likely it is to be the looked-for candidate. We call this ranking procedure *Neyman-Pearson ranking procedure.*

Definition 3.2.5 (Neyman-Pearson Ranking Procedure). *To each candidate k', assign the mark*

$$\mu_{k'} = \frac{\mathsf{f}_{M^*}(\Sigma_{k'})}{\mathsf{f}_M(\Sigma_{k'})} \tag{3.33}$$

where $\Sigma_{k'}$ is the statistic produced by the candidate k', and f_{M^*} and f_M are the density functions of $\Sigma_{k'}$ in case of the right and a wrong key, respectively. Then, sort the candidates by decreasing values of $\mu_{k'}$.

The following result is then a straightforward consequence of Th. 3.2.5 and of Eq. (3.32).

Theorem 3.2.10. *Neyman-Pearson key-ranking procedures are optimal (in the sense of Def. 3.2.4).*

Multiple lists (note that these considerations are valid for more than two lists, too) giving information on disjoint subsets of the key bits can thus be optimally combined easily if the *joint distribution* of the underlying statistics is available. Usually, reasonable heuristic statistical independence assumptions can be taken.

We prove now that, in case of a linear cryptanalysis, Matsui's ranking procedure is equivalent to a Neyman-Pearson ranking procedure. Without loss of generality, we will consider a linear approximation having a bias equal to $\varepsilon > 0$; furthermore, we recall that ν denotes the number of samples at disposal. Approximations of the Σ distributions were derived in §3.2.3; for the sake of clarity, we recall them:

$$\mathsf{f}_M(x) = \sqrt{\frac{8}{\nu\pi}} e^{-\frac{2x^2}{\nu}}, \quad \text{for } x \geq 0 \tag{3.34}$$

and

$$\mathsf{f}_{M^*}(x) = \sqrt{\frac{2}{\nu\pi}} \left(e^{-\frac{2(x-\nu\varepsilon)^2}{\nu}} + e^{-\frac{2(x+\nu\varepsilon)^2}{\nu}} \right) \quad \text{for } x \geq 0 \tag{3.35}$$

The likelihood-ratio is then given by a straightforward calculation.

Lemma 3.2.6. *In the case of a linear cryptanalysis, the likelihood-ratio is given by*

$$\mu_{k'} = e^{-2\nu\varepsilon^2} \cdot \cosh(4\varepsilon\Sigma_{k'}), \quad \Sigma_{k'} \geq 0 \tag{3.36}$$

where $\Sigma_{k'}$ is defined according to

$$\Sigma_{k'} = \Big| \frac{\hat{m}_{k'}}{\nu} - \frac{1}{2} \Big|.$$

We can now state the following result for single-list ranking.

Theorem 3.2.11. *Matsui's single-list ranking procedure (following Alg. 3.3) is equivalent to a Neyman-Pearson ranking procedure and is therefore optimal in terms of the number of key tests.*

Matsui's refined attack against DES published in [203] actually makes use of *two* linear approximations involving disjoint subsets of key bits; one is the best linear approximations on 14 rounds of DES and is used for deriving the second one using a "reversing trick". Each of them gives information about 13 key bits, the remaining 30 unknown key bits having to be searched exhaustively. In order to combine the information coming from these two sources, Matsui computes two independent ranking and generate the final ranking by considering subkey candidates sorted according to the product of their ranks in the two primitive lists of candidates. We can easily observe that Matsui's double-list ranking procedure, as described, although very simple, is not a Neyman-Pearson key-ranking procedure (it is actually not a total ordering) and it does not make use of the whole information given by each subkey candidate (i.e. it does not use the experimental bias associated with each candidate, but only their respective ranks). The first observation leads to some ambiguity in the implementation of this double-list ranking: should the combination of two candidates having respective ranks equal to 1 and 4 be searched for the unknown key bits before or after the combination consisting of two candidates having both rank 2? We now illustrate the use of a Neyman-Pearson ranking procedure in the case of a linear cryptanalysis of DES. We have to compute the joint probability distribution of the statistics $\Sigma_{k'_1}$ and $\Sigma_{k'_2}$ furnished by the two linear approximations. As these statistics are dependant of *disjoint* subsets of the key bits, one can reasonably take the following assumption.

Assumption 3.2.6. *For each k'_1 and k'_2, $\Sigma_{k'_1}$ and $\Sigma_{k'_2}$ are statistically independent, where k'_1 and k'_2 denote subkey candidates involving* disjoint *key subsets.*

A second assumption neglects the effects of *semi-wrong* keys, i.e. keys which behave as the right one according to a list only. This is motivated by the fact that, in case of a linear cryptanalysis of DES, the number of such keys is small, and thus their effect on the joint probability distribution is negligible.

Assumption 3.2.7. *For each k'_1 and k'_2, $\mathbf{\Sigma} = (\Sigma_{k'_1}, \Sigma_{k'_2})$ is distributed according either to $\mathsf{D}_{M^*} = \mathsf{D}^{(1)}_{M^*} \times \mathsf{D}^{(2)}_{M^*}$ or to $\mathsf{D}_M = \mathsf{D}^{(1)}_M \times \mathsf{D}^{(2)}_M$, where $\mathsf{D}^{(1)}_{M^*}$ and $\mathsf{D}^{(2)}_{M^*}$ are the distributions of the right subkey for both key subsets, and $\mathsf{D}^{(1)}_M$ and $\mathsf{D}^{(2)}_M$ are the distributions of a right subkey for both key subsets, respectively.*

Using these two assumptions, the probability density functions defined in Eq. (3.34) and Eq. (3.35), and the fact that the bias of both linear expression is the same and equal to ε, one can derive the likelihood-ratio:

$$\mu_{(k'_1,k'_2)} = e^{-4\nu\varepsilon^2} \cdot \cosh(4\varepsilon\Sigma_{k'_1}) \cdot \cosh(4\varepsilon\Sigma_{k'_2}) \tag{3.37}$$

As Eq. (3.37) is not "numerically" convenient to use, we may approximate it using a Taylor development in terms of ε, which gives a very intuitive definition of the Neyman-Pearson ranking procedure:

$$\mu_{(k'_1,k'_2)} \approx 1 + (8\Sigma^2_{k'_1} + 8\Sigma^2_{k'_2} - 4\nu)\varepsilon^2 + O(\varepsilon^4) \tag{3.38}$$

Hence, we can note that it is sufficient to rank the subkey candidates by decreasing values of $\Sigma^2_{k'_1} + \Sigma^2_{k'_2}$, i.e. the final mark is just the *Euclidean distance* between an unbiased result and a given sample.

We may generalize this result to the case where the biases, which we denote ε_1 and ε_2, are different in both equations; in this case, the likelihood-ratio is given by

$$\mu_{(k'_1,k'_2)} = e^{-2\nu(\varepsilon_2^1+\varepsilon_2^2)} \cosh(4\varepsilon_1\Sigma_{k'_1}) \cosh(4\varepsilon_2\Sigma_{k'_2}) \tag{3.39}$$

A first order approximation is then given by

$$\mu_{(k'_1,k'_2)} \approx 1 + 8\Sigma^2_{k'_1}\varepsilon_1^2 + 8\Sigma^2_{k'_2}\varepsilon_2^2 - 2\nu(\varepsilon_1^2 + \varepsilon_2^2) \tag{3.40}$$

which is equivalent to put a grade equal to $\mu_{(k'_1,k'_2)} = \Sigma^2_{k'_1}\varepsilon_1^2 + \Sigma^2_{k'_2}\varepsilon_2^2$. We summarize these facts in the following theorem.

Theorem 3.2.12. *Under assumptions 3.2.6 and 3.2.7, in a linear cryptanalysis using t approximations on disjoint key bits subsets having each a bias equal to $\varepsilon_i, 1 \leq i \leq t$, a procedure which ranks the subkey candidates by decreasing*

$$\mu_{(k'_1,\ldots,k'_t)} = \sum_{i=1}^{t} \left(\Sigma_{k'_i}\varepsilon_i\right)^2 \tag{3.41}$$

is a Neyman-Pearson ranking procedure, and furthermore, it is optimal.

We have implemented Neyman-Pearson key-ranking procedures in the case of DES, and we report some experimental results in §3.2.5, page 94. Finally, we note that several published attacks (to the best of our knowledge, all are derived from Matsui's paper) use key ranking procedures or suggest them as potential improvement. In [297], Shimoyama and Kaneko use quadratic Boolean approximations of DES' S-boxes possessing a larger bias. The first part of their attack consists in a traditional linear cryptanalysis, and thus we can apply our optimal ranking procedure; furthermore, another part of their attack consists also in a sorting procedure using Matsui's heuristic.

In [166], Knudsen and Mathiassen show how to modify Matsui's attack into a *chosen-plaintexts* attack in order to reduce the needs of pairs. Their attack can also use the "reversing trick", i.e. one can apply the same linear characteristic on both encryption and decryption function, in order to derive twice as much key bits. A new time, one could use a key-ranking procedure and our optimal rule to define the order of the subkey candidates during the exhaustive search part.

3.2.5 Implementation of Matsui's Attack

The linear cryptanalysis attack as described in [203] (and in Alg. 3.3), except the exhaustive search part[17], has been implemented. The computational most intensive part of the attack is obviously the encryption of 2^{43} plaintexts. For this, we have written in pure assembly code a DES encryption routine designed for the Intel Pentium III microprocessor, which we now describe.

Bitslice Implementation of DES

One of the main problems arising during an implementation of DES in the classical way is to deal effectively with the bit permutations. One has to consider each bit in a CPU register separately; this is clearly an ineffective way to employ the CPU's power. It is possible to use lookup tables and streamlined operations, but these techniques are memory intensive and the data quickly do not fit anymore in the CPU memory cache.

The *bitslice technique* was first proposed in the cryptography field by Biham [21], although it seems to be an implementation trick well-known in the hardware field. The idea behind bitslice is quite simple: one allocates one register for each bit of data, instead of storing all the bits in an unique register. This allows to process in parallel a number of bits which is equal to the size of the available registers.

If we consider DES (as defined in §2.2.1), we note that it mainly consists of permutations and substitutions. When assigning a register to a single bit, the permutations are dealt with at compile time, as it is actually an addressing issue. In other words, it is not necessary to isolate a given bit, which is a costly operation, because we have the bit ready in a register, or in a memory location which is hard-coded in the program.

The only remaining problem concerns the S-boxes. Instead of using lookup tables, one has to express this non-linear stage in DES as their gate circuit, i.e. as a Boolean expression. Fortunately, as there is no loop in an S-box, one does not have to deal with conditional behavior. The evaluation

[17] After having determined the final rank r of the right subkey candidate, we have computed the expected complexity (in DES evaluations) of the exhaustive search part as $(r-1) \cdot 2^{30} + 2^{29}$.

of these Boolean expressions will typically be more expensive than lookup-tables, but the cost is amortized by the fact that we can evaluate 32 or 64 bits in parallel, provided the register size at disposal is 32- or 64-bit wide, respectively.

As outlined before, the main advantages [268] of a bitsliced implementation are, on the one hand, that the bit permutations cost nothing at execution time, and on the other had, that the CPU's logical unit is used at full rate. Obviously, one can list several disadvantages [268] as well: the data are usually not available nor usable when they are spread over a bunch of registers, which implies some conversion stage (this problem is known as *orthogonalization problem*). Second, table lookups are not possible anymore and have to be replaced by some logical computation which may be rather painful to calculate and slow to execute. Even finding the optimal Boolean evaluation of a given S-box is not a trivial task. Third, the resulting code is large and it is possible to loose some speed if the CPU's instruction cache is not large enough. Fourth, in order to get some benefit from this technique, many registers are needed, as memory is slow. And, last but not least, this technique is very painful when implemented by hand. Although the number of disadvantages seems larger that the number of benefits, Biham [21] was able to gain a speed factor equal to 3 on a 64-bit Alpha architecture compared to the best classical implementations of DES.

Optimizations for the Intel Pentium MMX Architecture Intel's MMX architecture has been designed with the goal to improve multimedia and intensive floating-point arithmetic applications. It achieves this goal by offering a new set of 8 dedicated 64-bit registers, a new instruction set which allows to process data in a parallel way, and a super-scalar architecture. In our bitsliced implementation of DES, we have only used 5 MMX logical operations, namely `MOVQ`, `PXOR`, `PAND`, `PANDN`, and `POR`. Unfortunately, there are no other available logical operations, as it is sometimes the case on certain RISC architectures. Another drawback is the lack of `NOT` operation.

Taking account of the temporal property of a linear cryptanalysis, we have made a heavy use of the new cache management instructions of the Intel Pentium III: the `PREFETCH` instruction is able to retrieve a minimum of 32 bytes of data prior to the data actually needed. This hides the latency for data access in the time required to process data already resident in the cache. This instruction does not change the user-visible semantics of a program, but it may affect considerably the performances of a program when used on purpose. Finally, our code has been fully unrolled, to avoid any latency resulting from branch mispredications.

Kwan's Boolean Representation of S-Boxes In his paper [21], Biham, describes a Boolean representation of the DES S-boxes with an average count

S-box	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
Gate count	63	56	57	42	62	57	57	54
Clock cycles	283	271	276	262	279	272	275	269

Figure 3.5: Gate Count of Kwan's Representation

of 100 gates (see Fig. 3.5). These results where later improved by Kwan [176] down to an average gate count equal to 51; our fast DES implementation was hence implemented with Kwan's representation.

Performance Results The number of clock cycles on an Intel Pentium III spent[18] by our bitslice implementation is given in Fig. 3.5 for each S-box. The whole DES routine needs 14893 clock cycles to encrypt 64 blocks of data (i.e. 4096 bits), which represents an amortized cost of 232.7 clock cycles per block. On a CPU clocked at 666 MHz, this results in a throughput equal to 183 Mbps. One can hardly compare this number with existing fast implementations of DES, because of optimizations specifically related to the linear cryptanalysis; however, using classical available implementations for our purposes would have resulted in significantly poorer performances.

Pseudo-Random Generator

We have used a linear feedback shift register (LFSR), as suggested by Matsui [203], to generate the known plaintext-ciphertext pairs. Let ξ be a root of an primitive polynomial $\mathsf{p}(x)$ of degree n over GF (2). Then, a LFSR can be seen as the iteration of the mapping

$$i \mapsto i \times \xi$$

where the multiplications is done in the representation of the finite field GF (2^n) defined by $\mathsf{p}(x)$. Matsui [203] reports to have worked with such a strategy in GF $\left(2^{64}\right)$ to generate the known plaintext-ciphertext pairs, since LFSRs are known to have very good statistical properties and that they generate maximum-length sequences. We furthermore observe that one can implement 64 LFSRs in parallel in a bitslice manner provided 64-bit registers are available.

We found desirable to split the attack on several processors with a fine granularity, mainly for ease of process management reasons. In order to achieve this goal, we have split the 2^{43} work load in 2048 partial jobs, each handling 64 sequences at a time in parallel. Taking into account the effects of the birthday scenario in this precise situation, it is quite insightful to estimate the probability of generating *overlapping* sequences with a LFSR.

[18] Using the measurement procedure described in [9].

Lemma 3.2.7. *Let $x \in \mathrm{GF}(2^n)$ with $x \neq 0$ be the seed value of a sequence of length 2^ℓ generated by a maximal-period LFSR. Let*

$$\alpha = \left\lfloor \frac{2^n - 1}{3 \cdot 2^\ell - 2} \right\rfloor .$$

Let $m < \alpha$ be the number of generated sequences in parallel. Then, the probability π_{overlap} of generating two overlapping sequences is upper bounded by

$$\pi_{\text{overlap}} \leq 1 - \left(1 - \frac{m-1}{\alpha}\right)^{m-1}$$

Proof. We consider a sequence somewhere in th cycle of length $2^n - 1$. Let us reserve space of length $2^\ell - 1$ at the left and a the right of a given subsequence. Then, we divide the space in α non-overlapping subsequences of $3 \cdot 2^\ell - 2$ elements, and a birthday paradox reasoning is finally applied. □

Using Lem. 3.2.7, we observe that GF $\left(2^{64}\right)$ is too small to avoid overlapping sequences, so we chose to work in GF $\left(2^{128}\right)$ represented by the primitive polynomial $x^{128} + x^7 + x^2 + x + 1$.

Experimental Results

We have performed the attack 21 times, using the idle time of 8 to 16 CPUs; this represents between 3 and 6 days for a single run, depending on the availability of the computers. An exhaustive table of our experimental results regarding the complexity of the key exhaustive search part of the attack is available in Fig. 3.6. This table gives the experimental complexity for various amounts ν of plaintext-ciphertext pairs, where a figure x means 2^x DES evaluations.

It is widely accepted that linear cryptanalysis of DES, given 2^{43} known plaintext-ciphertext pairs, has a success probability of 85% within a complexity of 2^{43} DES encryptions, which are values given by Matsui in [203] (they are based on extrapolations from experiments on 8-rounds DES). Our experimental results lead to the following observations:

- Given 2^{43} known plaintext-ciphertext pairs, our experiments have a complexity of less than 2^{41} DES evaluation with a success probability of 86 % where more than the half of the cases have a complexity less than 2^{39}. Furthermore, if an attacker is ready to decrease her success probability, the complexity drops dramatically (less than 2^{34} DES evaluations with a success probability of 10 %).

- Given $2^{42.5}$ known plaintext-ciphertext pairs (i.e. with 30 % less pairs), half of the experiments have a complexity less than 2^{42} DES evaluations.

Exp	$\nu = 2^{43}$	$\nu = 2^{42.5}$	$\nu = 2^{42}$	$\nu = 2^{41}$	$\nu = 2^{40}$
1	39.1836	38.4818	45.0307	51.3802	51.0533
2	33.2479	41.6346	43.6383	48.0928	43.1913
3	38.6055	41.8023	43.9622	48.5492	51.6012
4	38.1267	34.6147	41.3351	48.7240	51.2041
5	37.4878	29.0000	36.5157	46.1991	52.3685
6	34.0444	44.2753	46.6834	48.5221	50.1937
7	36.4676	45.5732	44.2949	47.3010	51.2913
8	36.1189	44.7722	41.4091	51.6338	52.1143
9	40.3515	47.0565	48.6184	52.1953	53.1000
10	41.6540	41.8682	45.7429	47.9120	41.9750
11	45.4059	51.2973	51.9932	51.8155	52.1972
12	36.1189	43.6633	46.7256	50.3949	49.2317
13	36.4009	36.1189	43.2183	47.0756	46.7680
14	39.0042	42.6736	44.3057	44.7116	47.3256
15	37.6330	39.8572	47.6536	49.5244	52.6439
16	38.9204	36.6653	41.5447	49.1082	49.9939
17	33.5236	38.8502	43.3128	46.1030	48.6798
18	39.8478	47.4938	52.3671	50.6770	50.3675
19	32.1699	31.8074	40.5093	43.8552	48.4968
20	40.7503	38.3729	40.3734	45.2436	52.3101
21	41.8721	44.9063	45.4147	52.0730	52.8571

Figure 3.6: Experimental Results about Linear Cryptanalysis of DES.

ν	2^{43}	$2^{42.5}$	2^{42}	2^{41}	2^{40}
$r \leq 5$	20 (22)	13 (13.5)	7 (7.6)	0 (2.3)	0 (0.8)
$r \leq 10$	27 (25.8)	16 (17.1)	9 (10.5)	2 (3.6)	0 (1.3)
$r \leq 50$	33 (33.6)	26 (26.2)	18 (18.8)	5 (8.6)	2 (3.9)
$r \leq 150$	38 (37.7)	34 (32.3)	24 (25.7)	10 (14.3)	5 (7.7)
$r \leq 300$	42 (39.4)	39 (35.7)	31 (30.3)	17 (19.2)	14 (11.6)
$r \leq 600$	42 (40.8)	40 (38.5)	35 (34.6)	25 (24.7)	22 (16.8)
$\mathrm{E}[R]$	38 (71)	129 (182)	302 (362)	654 (847)	1121 (1312)

Figure 3.7: Comparison between experimental and theoretical results.

- With only 2^{40} pairs at disposal, the complexity is still far lower than an exhaustive search.

Even if we have to take these experimental results carefully because of the relative small number of statistical samples, they suggest strongly a lower complexity than expected by Matsui in [203] and we risk the following conjecture:

Conjecture 3.2.1. *Given 2^{43} known plaintext-ciphertext pairs, it is possible to recover a* DES *key using Matsui's Third Algorithm (Alg. 3.3) within a complexity of 2^{41}* DES *evaluations with a success probability of 85 %.*

These experimental results give us the possibility to test the accuracy of Cor. 3.2.1 as well. Each of the 21 experiments provides two statistical samples of the rank R of the right subkey candidate. Fig. 3.7 table summarizes our results about the ranks of the right subkey candidates for various amounts ν of known plaintext-ciphertext pairs and compare them to the theoretical expectations (values in smaller characters) given by Cor. 3.2.1. We observe that Cor. 3.2.1 seems to give a pessimistic expected value for the rank. However, we have noticed that Cor. 3.2.1 is very sensitive numerically. For instance, the expected rank $\mathrm{E}[R]$ is equal to 113 and to 39 when we assume that $\varepsilon = 1.1 \cdot 2^{-21}$ and $\varepsilon_r = 1.3 \cdot 2^{-21}$, respectively.

Finally, a key parameter regarding the linear cryptanalysis success is of course the bias of the involved linear approximation(s). As it is infeasible to compute the exact bias of a linear approximation, one uses implicit assumptions, such as the wrong-key randomization one (Ass. 3.2.2) and the statistical independence of data between two successive rounds (in the spirit of Lem. 3.2.2). The incidence of these assumptions has been well discussed in the literature (see [42, 125, 174, 250]). Although several situations where these assumptions can fail have been suggested and discussed, it is accepted that the linear expression real bias should be well approximated in case the of DES.

Our experimental results go indeed in this direction. We have computed the sample mean of the experimental biases $\hat{B}^*$ and $\hat{B}$, which can be compared to the expected values of the densities given in Eq. (3.27) and in Eq. (3.28). In case of a right key, the sample mean (averaged over the 42 samples) is equal to $5.5 \cdot 10^{-7}$ with a standard deviation of $0.2 \cdot 10^{-7}$. Thus, under our statistical model, we can estimate that

$$\Pr\left[4.9 \cdot 10^{-7} \leq \mathrm{E}[B^*] \leq 6.1 \cdot 10^{-7}\right] = 2\Phi(3) - 1 \approx 0.997$$

using a classical central-limit theorem reasoning[19]. This interval has to be compared with the expected value of a random variable distributed according to Eq. (3.27) furnished by our statistical model: $\mathrm{E}\left[\hat{B}^*\right] \approx 5.674 \cdot 10^{-7}$. We can at least conclude that there is no significant linear hull effect in DES.

Our experiments provide furthermore a good opportunity to confirm the validity of Ass. 3.2.2. The sample mean in case of wrong subkey candidates, averaged over all the wrong subkeys and all experiments, is equal to $1.38 \cdot 10^{-7}$ with a standard deviation of $0.03 \cdot 10^{-7}$, a value has to be compared with $\mathrm{E}[\hat{B}] = 1.345 \cdot 10^{-7}$ given by $\varepsilon = 0$ where $\hat{B}$ is distributed according to (3.28) and the following confidence interval

$$\Pr\left[1.29 \cdot 10^{-7} \leq \mathrm{E}[B] \leq 1.46 \cdot 10^{-7}\right] \approx 0.997.$$

This seems to fully legitimate Ass. 3.2.2 in the case of DES.

Experimental Results for the Neyman-Pearson Key-Ranking

The Neyman-Pearson ranking procedure described in the previous section has been simulated in the context of 21 linear cryptanalysis of DES, using the data than those described in the previous section. The following table summarizes our experimental results on the complexity of the exhaustive search part of the attack given 2^{43} known plaintext-ciphertext pairs; we use the following notation: $\mu_{\mathcal{C}}$ denotes the average experimental complexity, $\mathcal{C}_{85\%}$ the maximal complexity given a success probability of 85 %, which is the success probability defined by Matsui in [203], $\mathcal{C}_{\mathrm{med}}$ the median, $\mathcal{C}_{\mathrm{min}}$ and $\mathcal{C}_{\mathrm{max}}$ being the extremal values.

	Matsui's Ranking	Optimal Ranking	Δ
$\log_2 \mu_{\mathcal{C}}$	41.4144	40.8723	-31.32 %
$\log_2 \mathcal{C}_{85\%}$	40.7503	40.6022	-9.75 %
$\log_2 \mathcal{C}_{\mathrm{med}}$	38.1267	36.7748	-60.71 %
$\log_2 \mathcal{C}_{\mathrm{min}}$	32.1699	31.3219	-40.00 %
$\log_2 \mathcal{C}_{\mathrm{max}}$	45.4059	44.6236	-41.86 %

[19] We still have to keep in mind that the number of samples (42) is rather low.

These results lead us to the following observations:

- The average complexity is decreased by a factor of about 30 %. Actually, the average complexity is not a good statistical indicator for the average behavior of the linear cryptanalysis, because most cases have a far lower complexity and only 3 cases have a complexity greater than the average. Thus, those three cases have a considerable influence on the average complexity and it is worth examining the median behavior.

- A perhaps more significant result is that the median complexity is decreased by a factor of about 60 %. Although one have to be careful with this result because of the small size of the statistical samples number, this value seems to be more accurate regarding the real impact of the improved rule as the average one.

- Although the optimal rule decreases the exhaustive search part complexity on average, "pathological" cases where Matsui's heuristic is better than the Neyman-Pearson ranking procedure can occur. One can explain this by the fact that the Σ densities are sometimes bad approximations of the real ones, several heuristic assumptions being involved.

As the data complexity and the computational complexity of a linear cryptanalysis are closely related, it is possible (and desirable in the context of a known-plaintext attack) to convert a gain in the first category to a gain in the second one: even if we decrease sensibly the number of known plaintext-ciphertext pairs, the complexity will remain within reasonable areas: for instance, given $2^{42.46}$ known plaintext-ciphertext pairs, $\hat{\mathcal{C}}_{85\%} = 2^{44.46}$ DES evaluations, and with only 2^{42} pairs, $\hat{\mathcal{C}}_{85\%} = 2^{46.86}$; these experimental values are summarized in the following table:

Data complexity	$2^{42.00}$	$2^{42.46}$	$2^{43.00}$
Time complexity	$2^{46.86}$	$2^{44.46}$	$2^{40.60}$
Success probability	85 %	85 %	85 %

3.2.6 Summary

In this section, we have proposed a new and precise statistical modelization of various versions of Matsui's linear cryptanalysis against DES. The accuracy of these results was then checked with experimental results obtained by implementing the attack 21 times, and even if the number of attacks does not allow us to draw definitive conclusions, they clearly tend to support the heuristic assumptions we took.

A second contribution of this section is the optimization of the most recent (and most powerful) version of Matsui's attack obtained when using an optimal key-ranking procedure. Due to their genericity, the latter can be used in various types of statistical attacks. Furthermore, we have presented experimental results which tend to indicate that optimal key-ranking procedure allow to gain a non-negligible factor in terms of computational complexity.

Finally, we can note that the Neyman-Pearson paradigm, in spite of its simplicity, allowed us to state optimality results for attacks based on linear cryptanalysis. An interesting consequence is that one can give strict bounds on the best performance of an adversary implementing such attacks. The next section aims at considering the theoretical modelizations of certain classes of distinguishers at the light of those tools.

3.3 Statistical Modelization of Distinguishers

The theoretical modelization of *generic* statistical cryptanalytic procedures against block ciphers has been studied by only few researchers. In [313, 316], Vaudenay introduces the general concept of *iterated attacks* and gives security results against this class of attacks, as well as refined measures of security towards linear and differential cryptanalysis. In an even more abstract framework, we can furthermore outline the recent works of Maurer *et al.* [212, 214, 215] where they develop a general theory of random systems and prove several results about their properties in terms of distinguishability.

In this section, we make use of the Neyman-Pearson paradigm described in §3.1 to study certain classes of generic attacks against block ciphers. This allows us to derive new and tight bounds on the best advantage of any distinguisher for these models of attack.

3.3.1 Preliminaries

First of all, we recall basic notions about cryptographic distinguishers and about the Luby-Rackoff model of security [191] (we refer the reader to the brief introduction of §2.4.2, page 53, and to the bibliographical references therein). In the Luby-Rackoff security model, a *distinguisher* $\boldsymbol{\delta}^{\nu}$ is a computationally (and memory) unbounded Turing machine which can play with an *oracle* $\boldsymbol{\Omega}$ implementing a permutation C over some alphabet $\mathcal{B}$ (e.g. $\mathcal{B} = \{0,1\}^n$ with $n = 64$ or $n = 128$). Typically, $\boldsymbol{\Omega}$ can implement either a permutation on $\mathcal{B}$ chosen uniformly at random in a given family of permutations (i.e. a block cipher indexed by a key[20]), or a permutation C^* chosen uniformly at random from the set of all permutations on $\mathcal{B}$; the latter is usually called the *perfect cipher*.

[20] In this case, the randomness provides from the choice of the key.

The distinguisher $\boldsymbol{\delta}^\nu$ can submit a *bounded* number ν of queries to $\boldsymbol{\Omega}$ and ultimately outputs a decision bit "0" (if it guesses that C^* was implemented by $\boldsymbol{\Omega}$) or "1" (if it guesses that C was implemented by $\boldsymbol{\Omega}$); we are then interested in characterizing and computing its *advantage*[21]

$$\mathrm{Adv}_{\boldsymbol{\delta}^\nu}(\mathsf{C}, \mathsf{C}^*) = \left| \Pr_{\mathsf{C}} [\boldsymbol{\delta}^\nu(\mathbf{x}) = 1] - \Pr_{\mathsf{C}^*} [\boldsymbol{\delta}^\nu(\mathbf{x}) = 1] \right| \tag{3.42}$$

where $\mathbf{x} = (x_1, \ldots, x_\nu)$ is the vector of the values queried to the oracle; another important measure is the *best advantage* of any distinguisher:

$$\mathrm{BestAdv}^\nu(\mathsf{C}, \mathsf{C}^*) = \max_{\boldsymbol{\delta}^\nu} \mathrm{Adv}_{\boldsymbol{\delta}^\nu}(\mathsf{C}, \mathsf{C}^*)$$

Here, the maximum is taken over the set of all possible distinguishers between C and C^* asking at most ν queries to $\boldsymbol{\Omega}$.

There is an important difference between *adaptive* and *non-adaptive* distinguishers: an adaptive distinguisher is allowed to wait for an answer before submitting the next query, which may thus be a function of the previous answer. Alg. 3.4 describes[22] formally a ν-limited *adaptive* distinguisher, while Alg. 3.5 do the same for a ν-limited *non-adaptive* distinguisher. We remark

1: **Input**: An oracle $\boldsymbol{\Omega}$ implementing an unknown permutation U on $\mathcal{B}$, a complexity ν, functions f_i, with $1 \leq i \leq \nu$, an acceptance function accept.
2: **Output**: A decision bit.
3: Select a message $x_1 = \mathsf{f}_1()$ and get $y_1 = \mathsf{U}(x_1)$ from $\boldsymbol{\Omega}$.
4: Compute a message $x_2 = \mathsf{f}_2(x_1, y_1)$ and get $y_2 = \mathsf{U}(x_2)$ from $\boldsymbol{\Omega}$.
5: ...
6: Compute a message $x_\nu = \mathsf{f}_\nu(x_1, \ldots, x_{\nu-1}, y_1, \ldots, y_{\nu-1})$ and get $y_\nu = \mathsf{U}(x_\nu)$ from $\boldsymbol{\Omega}$.
7: Output $\mathsf{accept}(x_1, \ldots, x_\nu, y_1, \ldots, y_\nu)$.

Algorithm 3.4: ν-Limited Adaptive Distinguisher

1: **Input**: An oracle $\boldsymbol{\Omega}$ implementing an unknown permutation U on $\mathcal{B}$, a complexity ν, an acceptance function accept.
2: **Output**: A decision bit.
3: Select ν messages $\mathbf{X} = (x_1, \ldots, x_\nu)$.
4: Get $\mathbf{Y} = (y_1 = \mathsf{U}(x_1), \ldots, y_\nu = \mathsf{U}(x_\nu))$.
5: Output $\mathsf{accept}(\mathbf{X}, \mathbf{Y})$.

Algorithm 3.5: ν-Limited Non-Adaptive Distinguisher

[21] Note that the probabilities are taken on the permutation distribution (i.e. on the key) *and* possibly on the distribution of the random coins needed by $\boldsymbol{\delta}^\nu$.

[22] These definitions are those of Vaudenay [320].

that the core of Alg. 3.4 and Alg. 3.5 consists of the accept function (and of the functions f_i for the adaptive case) which actually define the attack used to distinguish C from C^*.

It is worth noticing that the Luby-Rackoff security model is an extremely strong model, since we put *no restriction* on the computational power of the adversary. In this model, a "security proof" would mean that we are able to provide an acceptable (i.e. tight and small) upper-bound on $\mathrm{BestAdv}^{\nu}(\mathsf{C}, \mathsf{C}^*)$ for a given block cipher C. As outlined in §2.4.2, the current state of research is able to give security proofs only for very few constructions, mainly for high-level schemes, like the Feistel cipher, whose round functions are themselves extremely strong functions. For these reasons, one possibility to slightly weaken the attack model consists in focusing on *iterated distinguishers*.

Informally, the basic idea behind the concept of iterated distinguisher is that we make use of an elementary distinguisher (or "core distinguisher") limited to d queries between a block cipher C and the ideal cipher C^*, and that we iterate a certain number of times this core distinguisher (in a statistically independent way) with the goal of amplifying the advantage of the core distinguisher. Note that we will restrict ourselves to *non-adaptive* core distinguishers. First of all, we recall the definition of Vaudenay [316] of a non-adaptive iterated distinguisher of order d, which we will denote by $\boldsymbol{\delta}^{\nu}_{\mathrm{iter(d)}}$ (see Alg. 3.6).

Definition 3.3.1 (Non-Adaptive Iterated Distinguisher). *Let ν and d be strictly positive integers, and let $\mathcal{B}$ be a set. Then, a* non-adaptive iterated distinguisher of order d and complexity ν, *denoted $\boldsymbol{\delta}^{\nu}_{\mathrm{iter(d)}}$, for a permutation on $\mathcal{B}$, is a computationally unbounded Turing machine characterized by a probability distribution $\mathsf{D}_{\mathcal{B}}$ on $\mathcal{B}^d$, named the* plaintext distribution*, a (possibly randomized) function $\mathsf{test} : \mathcal{B}^{2d} \to \{0,1\}$, named a* test function*, and a (possibly randomized) function $\mathsf{accept} : \{0,1\}^{\nu} \to \{0,1\}$, named an* acceptance function*.*

Actually, non-adaptive iterated distinguishers model a large number of the known attacks against block ciphers.

- Linear cryptanalysis (see §2.3.3, page 45), as well as some of these variants (like non-linear cryptanalysis as proposed by Knudsen and Robshaw [167] and by Shimoyama and Kaneko [297]) is an non-adaptive iterated distinguisher of order $d = 1$.

- Differential cryptanalysis (see §2.3.3, page 40) as well as some of these variants (like impossible differential attacks [27] and truncated differential attacks [161]) is an non-adaptive iterated distinguisher of order $d = 2$.

1: **Input**: An oracle $\mathbf{\Omega}$ implementing an unknown permutation U on $\mathcal{B}$, a plaintext distribution $\mathsf{D}_{\mathcal{B}}$, a complexity ν, a test function test, and an acceptance function accept.
2: **Output**: A decision bit.
3: **for** i from 1 to ν **do**
4: Pick a random $\mathbf{x}_i = (x_1, \ldots, x_d)$ according to $\mathsf{D}_{\mathcal{B}}$ and submit them to the oracle $\mathbf{\Omega}$.
5: Get the corresponding $\mathbf{y}_i = (\mathsf{U}(x_1), \ldots, \mathsf{U}(y_d))$ from $\mathbf{\Omega}$.
6: Pick a random bit b_i with an expected value equal to $\mathsf{test}(\mathbf{x}_i, \mathbf{y}_i)$.
7: **end for**
8: Output a decision bit with an expected value equal to $\mathsf{accept}(b_1, \ldots, b_\nu)$.

Algorithm 3.6: Non-Adaptive Iterated Distinguisher of Order d

- Differential-linear cryptanalysis [183] and its generalization [29] (see §2.3.3, page 47) is an non-adaptive iterated distinguisher of order $d = 2$.

- Integral attacks (see §2.3.4, page 48) [134, 169] considering sums of α words is a non-adaptive iterated distinguisher of order $d = \alpha$.

There are statistical attacks which do not fit into this framework, mainly because they store more than a single bit between the executions of the core distinguisher: for instance, Vaudenay's χ^2 cryptanalysis [312], Davies' attack and all the non-surjective attacks (see §2.3.3, page 47), and several generalizations of linear cryptanalysis, like the partitioning cryptanalysis [126] of Harpes and Massey. Actually, all of these attacks are based on distinguishers working on probability distributions defined on sets of cardinality greater than two. Wagner's boomerang attack [322] (see §2.3.3, page 44) as well as its refinements (the amplified boomerang attack [158], and the rectangle attack of Biham, Dunkelman and Keller [28]) can be seen as an *adaptive* iterated distinguisher of order $d = 4$, meaning that the core distinguisher is adaptive.

3.3.2 Distinguishing Two Binary Random Sources

In the following, we will assume the following simple statistical game: we consider an oracle $\mathbf{\Omega}$ which first draws a bit b uniformly at random from $\{0, 1\}$. Depending on the value of b, $\mathbf{\Omega}$ implements either a random source distributed according to D_0, or a random source distributed according to D_1 (where D_0 is referred to as an "ideal" distribution). The oracle $\mathbf{\Omega}$ allows then a distinguisher $\boldsymbol{\delta}^\nu$ to submit at most ν uniformly distributed and statistically independent queries; at the end of the game, $\boldsymbol{\delta}^\nu$ must output a guess about the bit b. We are then naturally interested in bounding the advantage of *any* computationally unbounded distinguisher in this scenario.

We now interpret this framework as a statistical hypothesis test. When dealing with error probabilities, one usually proceeds as follows in the classical approach: one of the two possible error probabilities is fixed, and we minimize the other error probability if we use the Neyman-Pearson lemma to define the acceptance function. However, this approach lacks symmetry and it is quite inconvenient in our case. Another possibility is to follow a *Bayesian approach* and to assign *prior* probabilities π_0 and π_1 to both hypotheses, respectively, and to assume that correct decisions are not penalized, while incorrect decisions are penalized equally, then the optimal Bayesian decision rule is given by

$$\mathsf{accept}(x) = \begin{cases} 0 & \text{if} \quad \pi_0 \Pr_{\mathsf{D}_0}[x] > \pi_1 \Pr_{\mathsf{D}_1}[x] \\ c \in_U \{0,1\} & \text{if} \quad \pi_0 \Pr_{\mathsf{D}_0}[x] = \pi_1 \Pr_{\mathsf{D}_1}[x] \\ 1 & \text{if} \quad \pi_0 \Pr_{\mathsf{D}_0}[x] < \pi_1 \Pr_{\mathsf{D}_1}[x] \end{cases} \tag{3.43}$$

Here, $c \in_U \{0,1\}$ means that a bit[23] c is drawn uniformly at random. In this case, according to [274, page 583], the above rule minimizes the *overall error probability* defined by

$$\pi_e = \pi_0 \alpha + \pi_1 \beta \tag{3.44}$$

where α is the probability that accept outputs "1" when the samples are distributed according to D_0, and β is the probability that accept outputs "0" when the samples are actually distributed according to D_1.

The link between the overall error probability defined by Eq. (3.44) and the advantage defined by Eq. (3.42) is made clear by the following simple lemma.

Lemma 3.3.1. *Let π_e denote the overall probability of error of a distinguisher $\boldsymbol{\delta}$ as defined in Eq.* (3.44) *with $\pi_0 = \pi_1 = \frac{1}{2}$. Then,*

$$\mathrm{Adv}_{\boldsymbol{\delta}}(\mathsf{C}, \mathsf{C}^*) = 1 - 2\pi_e = 1 - (\alpha + \beta). \tag{3.45}$$

Proof. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the vector of the queries and of the answers given by an oracle $\boldsymbol{\Omega}$, respectively. First of all, by the definition of the advantage, we have

$$\begin{aligned} \mathrm{Adv}_{\boldsymbol{\delta}}(\mathsf{C}, \mathsf{C}^*) &= \left| \Pr_{\mathsf{C}}[\boldsymbol{\delta}(\boldsymbol{x}, \boldsymbol{y}) = 1] - \Pr_{\mathsf{C}^*}[\boldsymbol{\delta}(\boldsymbol{x}, \boldsymbol{y}) = 1] \right| \\ &= \left| 1 - \Pr_{\mathsf{C}^*}[\boldsymbol{\delta}(\boldsymbol{x}, \boldsymbol{y}) = 1] - \Pr_{\mathsf{C}}[\boldsymbol{\delta}(\boldsymbol{x}, \boldsymbol{y}) = 0] \right| \\ &= \left| 1 - \alpha - \beta \right| = \left| 1 - (\alpha + \beta) \right| \end{aligned}$$

[23] Actually, the acceptance function $\mathsf{accept}(.)$ does not need to be randomized: we can make it output 0 if and only if $\pi_0 \Pr_{\mathsf{D}_0}[X = x] \geq \pi_1 \Pr_{\mathsf{D}_1}[X = x]$ without modifying the overall error probability.

Now, in order to prove the equality (3.45), we have still to show that $\alpha+\beta \leq 1$. Let us assume the opposite, i.e. that $\alpha + \beta > 1$. By the Neyman-Pearson lemma, we know that the optimal Bayesian rule minimizes π_e. Let us "invert" the rule (*i.e.* we make it output the opposite result). Then,

$$\pi_e = \pi_0(1-\alpha) + \pi_1(1-\beta) = \frac{1}{2}(2-\alpha-\beta) < \frac{1}{2} \tag{3.46}$$

which implies that the inverted rule would possess a "better" π_e, which is a contradiction and proves the lemma. □

In the following, we will consider the case encountered during a linear cryptanalysis, namely the case where we have to distinguish between the two following random sources: one of the sources generating independent and uniformly distributed bits, while the second one is slightly biased. More formally, let D_0 be the uniform distribution on $\{0,1\}$ and let D_1 be a probability distribution defined as

$$\Pr_{\mathsf{D}_1}[X=0] = 1 - \Pr_{\mathsf{D}_1}[X=1] = \frac{1}{2} + \varepsilon$$

such that $0 < |\varepsilon| \ll \frac{1}{2}$. First of all, we can describe precisely the shape of the acceptance function of an optimal distinguisher between these two random sources.

Lemma 3.3.2. *Let* D_0 *be the uniform distribution on* $\{0,1\}$ *and* D_1 *be a probability distribution defined as* $\Pr_{\mathsf{D}_1}[X=0] = 1 - \Pr_{\mathsf{D}_1}[X=1] = \frac{1}{2} + \varepsilon$ *with* $\varepsilon \geq 0$. *Let* $\boldsymbol{\delta}^\nu$ *be a computationally unbounded distinguisher limited to* ν *queries implementing an acceptance function* accept *defined by*

$$\mathsf{accept}(u) = 1 \iff u \geq \nu \cdot \frac{\log(1-2\varepsilon)}{\log(1-2\varepsilon) - \log(1+2\varepsilon)} \tag{3.47}$$

where $0 \leq u \leq \nu$ *is the number of times that the source outputs 0. Then* $\boldsymbol{\delta}^\nu$ *maximizes* $\mathrm{Adv}_{\boldsymbol{\delta}^\nu}(\mathsf{D}_0, \mathsf{D}_1)$.

Proof. We know that optimal decision rule is defined by Eq. (3.43). In other words, $\boldsymbol{\delta}^\nu$ must decide for D_1 if and only if

$$\left(\frac{1}{2}+\varepsilon\right)^u \left(\frac{1}{2}-\varepsilon\right)^{\nu-u} \geq \frac{1}{2^\nu}$$

which is equivalent to

$$u \log_2\left(\frac{1+2\varepsilon}{1-2\varepsilon}\right) + \nu \log_2(1-2\varepsilon) \geq 0$$

The lemma follows by noticing that ε is positive by assumption. □

Lem. 3.3.2 can be easily adapted for the case $\varepsilon < 0$, and we obtain a symmetric acceptance function satisfying

$$\mathsf{accept}(u) = 1 \iff u \leq \nu \cdot \frac{\log(1+2\varepsilon)}{\log(1+2\varepsilon) - \log(1-2\varepsilon)}.$$

Note furthermore that, when ε is small and positive (the other case being similar), we can approximate very accurately Eq. (3.47) as

$$\mathsf{accept}(u) = 1 \iff u \geq \nu \left(\frac{1}{2} + \frac{\varepsilon}{2}\right).$$

A Chernoff-Like Bound

We now focus on the advantage of an optimal distinguisher. Vaudenay gives in [320] the following result.

Lemma 3.3.3 (Vaudenay [320]). *For any computationally unbounded distinguisher $\boldsymbol{\delta}^{\nu}$ limited to ν queries,*

$$\mathrm{Adv}_{\boldsymbol{\delta}^{\nu}}(\mathsf{D}_0, \mathsf{D}_1) \leq 4|\varepsilon|\sqrt{\nu} \tag{3.48}$$

where D_0 is the uniform distribution on $\{0,1\}$ and D_1 is a probability distribution defined as $\Pr_{\mathsf{D}_1}[X=0] = 1 - \Pr_{\mathsf{D}_1}[X=1] = \frac{1}{2} + \varepsilon$.

Note that Th. 3.3.3 indicates that ν should be in the order of ε^{-2} for having a non-negligible advantage. However, it does not seem to capture the asymptotic behavior of the advantage for $\nu \to +\infty$, since the right-hand side of Eq. (3.48) tends to the infinity when $\nu \to +\infty$ and since $\mathrm{Adv}_{\boldsymbol{\delta}^{\nu}}$ is upper bounded by 1. Let $\alpha^{(\nu)}$ and $\beta^{(\nu)}$ denote the respective error probabilities when an optimal distinguisher outputs a decision bit after having queries ν samples. Clearly, the overall error probability $\pi_e^{(\nu)} = \pi_0\alpha^{(\nu)} + \pi_1\beta^{(\nu)}$ of an optimal Bayesian distinguisher defined according to Eq. (3.43) must decrease towards zero as the number ν of samples increases (and thus the advantage must asymptotically go to 1). It turns out that the decrease asymptotically approaches an exponential in the number of samples drawn before the decision, the exponent being given by the so-called *Chernoff bound* (stated in Th. 3.3.1). We refer the reader to App. B for a very short introduction to the theory behind the method of types, which is very useful to prove Chernoff's bound.

Theorem 3.3.1 (Chernoff). *The best probability of error of the Bayesian decision rule defined in* (3.43) *satisfies*

$$\lim_{\nu \to +\infty} \frac{1}{\nu} \log \frac{\pi_e^{(\nu)}}{2^{-\nu\gamma}} = 0$$

where $\gamma = C(\mathsf{D}_0, \mathsf{D}_1)$ *is the* Chernoff information *between* D_0 *and* D_1 *defined by*

$$C(\mathsf{D}_0, \mathsf{D}_1) = -\min_{0 \leq \lambda \leq 1} \log_2 \left(\sum_{x \in \mathcal{X}} \Pr_{X_0}[x]^{\lambda} \Pr_{X_1}[x]^{1-\lambda} \right).$$

Note that the Bayesian error exponent does not depend on the actual value of π_0 and π_1, as long as they are non-zero: essentially, the effect of the prior probabilities is just washed out for large sample sizes. Using an improved version of Chernoff's theorem (Th. 3.3.1) adapted to probability distributions on binary sets, we can bound the advantage of the best linear-like distinguisher as follows.

Theorem 3.3.2. *For any computationally unbounded optimal iterated distinguisher* $\boldsymbol{\delta}^{\nu}$ *of order 1 limited to* ν *queries,*

$$1 - \frac{(\nu+1)}{2^{\nu\gamma-1}} \leq \mathrm{Adv}_{\boldsymbol{\delta}^{\nu}_{\mathrm{lin}}}(\mathsf{D}_0, \mathsf{D}_1) \leq 1 - \frac{1}{(\nu+1) \cdot 2^{\nu\gamma-1}} \tag{3.49}$$

where $\gamma = C(\mathsf{D}_0, \mathsf{D}_1)$ *is the Chernoff information between* D_0*, the uniform distribution on* $\{0,1\}$ *and* D_1*, a probability distribution defined as* $\Pr_{\mathsf{D}_1}[X = 0] = 1 - \Pr_{\mathsf{D}_1}[X = 1] = \frac{1}{2} + \varepsilon$.

Proof. In order to show the bounds given in Th. 3.3.2, we use an improved version of Sanov's Theorem than the one given in Th. B.2.1 tailored to binary random variables. Let $\mathcal{A}^{(\nu)}_{\mathrm{opt}}$ be the optimal acceptance region for $\boldsymbol{\delta}^{\nu}$ Let $\mathcal{E}_{\alpha^{(\nu)}} \in \mathcal{P}_{\nu}$ be the set of types such that

$$\mathcal{E}_{\alpha^{(\nu)}} = \left\{ \mathbf{x} \in \mathcal{P}_{\nu} : \mathsf{D}_{\mathbf{x}} \notin \mathcal{A}^{(\nu)}_{\mathrm{opt}} \right\}$$

when $\mathbf{x} \leftarrow \mathsf{D}_{X_0^{\nu}}$. Similarly,

$$\mathcal{E}_{\beta^{(\nu)}} = \left\{ \mathbf{x} \in \mathcal{P}_{\nu} : \mathsf{D}_{\mathbf{x}} \in \mathcal{A}^{(\nu)}_{\mathrm{opt}} \right\}$$

when $\mathbf{x} \leftarrow \mathsf{D}_{X_1^\nu}$. Then,

$$\begin{aligned}
\Pr_{X_0^\nu}[\mathcal{E}_{\alpha^{(\nu)}}] &= \sum_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu} \Pr_{X_0^\nu}[\mathcal{T}(\mathsf{D}_X)] \\
&\leq \sum_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu} 2^{-\nu D(\mathsf{D}_X || X_0)} \\
&\leq \sum_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu} \max_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu} 2^{-\nu D(\mathsf{D}_X || X_0)} \\
&= \sum_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu} 2^{-\nu \min_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu} D(\mathsf{D}_X || X_0)} \\
&\leq \sum_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu} 2^{-\nu \min_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}}} D(\mathsf{D}_X || X_0)} \\
&= \sum_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu} 2^{-\nu D(\mathsf{D}_{X^*} || X_0)} \\
&\leq (\nu+1) \cdot 2^{-\nu D(\mathsf{D}_{X^*} || X_0)}
\end{aligned}$$

where the last inequality comes from

$$|\mathcal{P}_\nu| = \binom{\nu + |\mathcal{X}| - 1}{|\mathcal{X}| - 1}$$

The computation for upper bounding $\Pr_{X_1^\nu}[\mathcal{E}_{\beta^{(\nu)}}]$ are similar. For the lower bound, we need a set $\mathcal{E}_{\alpha^{(\nu)}}$ such that for all large ν, we can find a distribution in $\mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu$ which is close to D_{X^*}. As $\mathcal{E}_{\alpha^{(\nu)}}$ is the closure of its interior (thus the interior must be non-empty), then since $\bigcup_\nu \mathcal{P}_\nu$ is dense in the set of all distributions, it follows that $\mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu$ is non-empty for all $\nu \geq \nu_0$ for some ν_0. We can then find a sequence of distributions D_{X_ν} such that $\mathsf{D}_{X_\nu} \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu$ and $D(\mathsf{D}_{X_\nu} || \mathsf{D}_{X_0^\nu}) \to D(\mathsf{D}_{X^*} || _{X_0^\nu})$. For each $\nu \geq \nu_0$,

$$\begin{aligned}
\Pr_{X_0^\nu}[\mathcal{E}_{\alpha^{(\nu)}}] &= \sum_{\mathsf{D}_X \in \mathcal{E}_{\alpha^{(\nu)}} \cap \mathcal{P}_\nu} \Pr_{X_0^\nu}[\mathcal{T}(\mathsf{D}_X)] \\
&\geq \Pr_{X_0^\nu}[\mathcal{T}(\mathsf{D}_X)] \\
&\geq \frac{2^{-\nu D(\mathsf{D}_{X_\nu} || \mathsf{D}_{X_0})}}{\nu+1}
\end{aligned}$$

Consequently,

$$\begin{aligned}
\liminf \frac{1}{\nu} \Pr_{X_0^\nu}[\mathcal{E}_{\alpha^{(\nu)}}] &\geq \liminf \left(-\frac{\log(\nu+1)}{\nu} - D(\mathsf{D}_{X_\nu} || X_0) \right) \\
&= -D(\mathsf{D}_{X^*} || \mathsf{D}_{X_0})
\end{aligned}$$

The computations are similar for lower bounding $\Pr_{X_1^\nu}[\mathcal{E}_{\beta^{(\nu)}}]$ Combining the upper bounds derived before and this lower bound, and the computations of App. B yields Th. 3.3.2. □

Note that in our case, we can express the Chernoff information between D_0 and D_1 as

$$\gamma = C(\mathsf{D}_0, \mathsf{D}_1) = -\min_{0 \leq \lambda \leq 1} 2^{-\lambda} \cdot ((\frac{1}{2} + \varepsilon)^{\lambda} + (\frac{1}{2} - \varepsilon)^{1-\lambda})$$

is maximal for

$$\lambda = \frac{\log\left(-\frac{\log(1-2\varepsilon)}{\log(1+2\varepsilon)}\right)}{\log\left(\frac{1+2\varepsilon}{1-2\varepsilon}\right)}.$$

Thus, using a Taylor approximation, we finally get that the Chernoff exponent satisfies

$$\gamma = C(\mathsf{D}_0, \mathsf{D}_1) = -\frac{\varepsilon^2}{2} + O(\varepsilon^4).$$

This clearly matches the results of Lem. 3.48 stating that the advantage of an optimal distinguisher becomes significant when $\nu \approx \varepsilon^{-2}$. Finally, we note that the derivation of Chernoff's bound is still valid for *any* discrete probability distribution pairs; consequently, they are not extremely tight when applied in our statistical scenario.

Tighter Bounds

We now prove tighter bounds on the best advantage of any iterated distinguisher of order 1 using the same techniques than in the proof of Th. 3.2.5. First of all, let

$$z = \frac{k + \frac{1}{2} - np}{\sqrt{np(1-p)}};$$

the following notation will then used later:

$$\Phi^{(n,p)}_{\mathsf{down}}(k) = \Phi(z) - \frac{(1-2p)(z^2-1)\phi(z)}{6\sqrt{np(1-p)}} - \frac{0.056}{\sqrt{np(1-p)}} \tag{3.50}$$

and

$$\Phi^{(n,p)}_{\mathsf{up}}(k) = \Phi(z) - \frac{(1-2p)(z^2-1)\phi(z)}{6\sqrt{np(1-p)}} + \frac{0.056}{\sqrt{np(1-p)}} \tag{3.51}$$

represents the lower and the upper bound of Raff's theorem Th. 3.2.4, respectively. Let us furthermore denote by

$$\tau = \frac{\log(1-2\varepsilon)}{\log(1-2\varepsilon) - \log(1+2\varepsilon)} \tag{3.52}$$

the optimal threshold defined by Lem. 3.3.2. Let U denote a random variable modeling the number of times the unknown source outputs 1. Then, we have, according to Lem. 3.3.1 and Lem. 3.3.2,

$$\mathrm{BestAdv}^{\nu}(\mathsf{D}_0, \mathsf{D}_1) = 1 - (\alpha + \beta)$$

where

$$\alpha = \Pr_{\mathsf{D}_0}[U > \lceil \tau\nu \rceil] \text{ and } \beta = \Pr_{\mathsf{D}_1}[U < \lfloor \tau\nu \rfloor].$$

Using the notation defined in Eq. (3.50) and in Eq. (3.51), we get

$$1 - (\alpha + \beta) \geq \Phi_{\mathsf{down}}^{(\nu,\frac{1}{2})}(\lfloor \tau\nu \rfloor) - \Phi_{\mathsf{up}}^{(\nu,\frac{1}{2}+\varepsilon)}(\lfloor \tau\nu \rfloor).$$

Similarly, the following lower bound may be derived:

$$1 - (\alpha + \beta) \leq \Phi_{\mathsf{up}}^{(\nu,\frac{1}{2})}(\lfloor \tau\nu \rfloor) - \Phi_{\mathsf{down}}^{(\nu,\frac{1}{2}+\varepsilon)}(\lfloor \tau\nu \rfloor).$$

We have thus proven the following theorem.

Theorem 3.3.3. *Let* D_0 *be the uniform distribution on* $\{0,1\}$ *and* D_1 *be a probability distribution defined as* $\Pr_{\mathsf{D}_1}[X = 0] = 1 - \Pr_{\mathsf{D}_1}[X = 1] = \frac{1}{2} + \varepsilon$. *Then, for an optimal computationally unbounded iterated distinguisher* $\boldsymbol{\delta}^{\nu}$ *of order 1 limited to* ν *queries,*

$$\mathrm{BestAdv}^{\nu}(\mathsf{D}_0, \mathsf{D}_1) \geq \Phi_{\mathsf{down}}^{(\nu,\frac{1}{2})}(\lfloor \tau\nu \rfloor) - \Phi_{\mathsf{up}}^{(\nu,\frac{1}{2}+\varepsilon)}(\lfloor \tau\nu \rfloor)$$

and

$$\mathrm{BestAdv}^{\nu}(\mathsf{D}_0, \mathsf{D}_1) \leq \Phi_{\mathsf{up}}^{(\nu,\frac{1}{2})}(\lfloor \tau\nu \rfloor) - \Phi_{\mathsf{down}}^{(\nu,\frac{1}{2}+\varepsilon)}(\lfloor \tau\nu \rfloor),$$

where τ, Φ_{down}, *and* Φ_{up} *are defined in Eq.* (3.52), *Eq.* (3.50), *and Eq.* (3.51), *respectively.*

Without loss of generality, let us assume that $\varepsilon > 0$ is small and fixed. For $\nu \to +\infty$, we have $\tau \approx \frac{1}{2} + \frac{\varepsilon}{2}$, and it is then easy to verify that

$$\lim_{\nu\to+\infty} \Phi_{\mathsf{down}}^{(\nu,\frac{1}{2})}(\lfloor \tau\nu \rfloor) = \lim_{\nu\to+\infty} \Phi_{\mathsf{up}}^{(\nu,\frac{1}{2})}(\lfloor \tau\nu \rfloor) = 1$$

as well as

$$\lim_{\nu\to+\infty} \Phi_{\mathsf{up}}^{(\nu,\frac{1}{2}+\varepsilon)}(\lfloor \tau\nu \rfloor) = \lim_{\nu\to+\infty} \Phi_{\mathsf{down}}^{(\nu,\frac{1}{2})}(\lfloor \tau\nu \rfloor) = 0.$$

Thus, we have accordingly

$$\lim_{\nu\to+\infty} \mathrm{BestAdv}^{\nu}(\mathsf{D}_0, \mathsf{D}_1) = 1.$$

3.3.3 Optimal Linear Distinguishers

A *linear distinguisher* $\boldsymbol{\delta}_{\mathrm{lin}}^{\nu}$ is a (possibly computationally unbounded) Turing machine which can play with an oracle $\boldsymbol{\Omega}$ (see Alg. 3.7); it is actually a subclass of iterated distinguishers of order 1 which uses a linear approximation as $\mathsf{test}(.)$ function. Let us now consider the following scenario. We have a permutation C on $\{0,1\}^n$ for which

$$\Pr[\mathbf{a} \cdot x \oplus \mathbf{b} \cdot \mathsf{C}(x) = 0] = \frac{1}{2} + \varepsilon$$

1: **Input**: An oracle $\mathbf{\Omega}$ implementing an unknown permutation U on $\mathcal{B}$, a complexity ν, a linear approximation $(\mathbf{a}, \mathbf{b})$, an acceptance function accept.
2: **Output**: A decision bit.
3: Initialize a counter u to 0.
4: **for** $i = 1$ to ν **do**
5: Pick uniformly at random x and query $\mathsf{U}(x)$ to the oracle $\mathbf{\Omega}$.
6: **if** $\mathbf{a} \cdot x \oplus \mathbf{b} \cdot \mathsf{U}(x) = 0$ **then**
7: Increment u.
8: **end if**
9: **end for**
10: Output $\mathsf{accept}(u)$.

Algorithm 3.7: Classical modelization of a linear distinguisher $\boldsymbol{\delta}^{\nu}_{\mathrm{lin}}$.

for half of the keys, and

$$\Pr[\mathbf{a} \cdot x \oplus \mathbf{b} \cdot \mathsf{C}(x) = 0] = \frac{1}{2} - \varepsilon$$

for the other half, with $\varepsilon > 0$. This corresponds to an idealized version of a linear cryptanalysis where Ass. 3.2.1 holds. We can rewrite the above expression as

$$\Pr[\mathbf{a} \cdot x \oplus \mathbf{b} \cdot \mathsf{C}(x) = 0] = \frac{1}{2} + (-1)^{\kappa}\varepsilon$$

for a fixed value $\kappa \in \{0, 1\}$ depending on the key. We bound now the advantage of an optimal linear distinguisher $\boldsymbol{\delta}^{\nu}_{\mathrm{lin}}$ *knowing* κ between C and a uniformly distributed binary iid source D_0 using Th. 3.3.3, which gives us

$$\mathrm{Adv}_{\boldsymbol{\delta}^{\nu}_{\mathrm{lin}}}(\mathsf{C}, \mathsf{D}_0) \leq \Phi^{(\nu, \frac{1}{2})}_{\mathsf{up}}(\lfloor \tau\nu \rfloor) - \Phi^{(\nu, \frac{1}{2}+\varepsilon)}_{\mathsf{down}}(\lfloor \tau\nu \rfloor)$$

where τ, Φ_{down}, and Φ_{up} are defined in Eq. (3.52), Eq. (3.50), and Eq. (3.51), respectively. According to Vaudenay [320], the advantage of an optimal linear distinguisher aiming at distinguishing C^* from a uniformly distributed binary iid source is bounded by

$$\mathrm{Adv}_{\boldsymbol{\delta}^{\nu}_{\mathrm{lin}}}(\mathsf{C}^*, \mathsf{D}_0) \leq 3\sqrt[3]{\frac{\nu}{2^n - 1}} \tag{3.53}$$

where n is the block size of the permutations. Then, using the triangle inequality, we note that

$$\mathrm{Adv}_{\boldsymbol{\delta}^{\nu}_{\mathrm{lin}}}(\mathsf{C}^*, \mathsf{C}) \leq \mathrm{Adv}_{\boldsymbol{\delta}^{\nu}_{\mathrm{lin}}}(\mathsf{C}^*, \mathsf{D}_0) + \mathrm{Adv}_{\boldsymbol{\delta}^{\nu}_{\mathrm{lin}}}(\mathsf{C}, \mathsf{D}_0).$$

Then, we are in position to prove the following result.

Corollary 3.3.1. *Let* C *be a permutation such that*

$$\Pr[\mathbf{a} \cdot x \oplus \mathbf{b} \cdot \mathsf{C}(x) = 0] = \frac{1}{2} + (-1)^{\kappa}\varepsilon$$

for $\kappa \in \{0, 1\}$ *where* $\kappa = 0$ *for half of the keys. Then, the advantage of any linear distinguisher* $\boldsymbol{\delta}^{\nu}_{\text{lin}}$ *limited to* ν *queries between* C *and* C^* *is upper bounded by*

$$\text{Adv}_{\boldsymbol{\delta}^{\nu}_{\text{lin}}}(\mathsf{C}^*, \mathsf{C}) \leq \Phi^{(\nu,\frac{1}{2})}_{\mathsf{up}}(\lfloor \tau\nu \rfloor) - \Phi^{(\nu,\frac{1}{2}+\varepsilon)}_{\mathsf{down}}(\lfloor \tau\nu \rfloor) + 3\sqrt[3]{\frac{\nu}{2^n - 1}} \tag{3.54}$$

where n *is the block size of the permutations and where* τ, Φ_{down}, *and* Φ_{up} *are defined in Eq.* (3.52), *Eq.* (3.50), *and Eq.* (3.51), *respectively.*

Proof. These distinguishers are a particular case of more powerful ones which are given the value of κ as input (for C) or a uniformly distributed random bit (for C^*). Using the triangular inequality, Th. 3.3.3, and Eq. (3.53), we obtain the claimed result. □

We would like to remark that the condition that $\boldsymbol{\delta}^{\nu}_{\text{lin}}$ knows the value κ is not practically annoying, since it would be possible to build an optimal linear distinguisher based on the acceptance function defined by

$$\mathsf{accept}(u) = 1 \iff \frac{\frac{1}{2}\left(\frac{1}{2}+\varepsilon\right)^{u}\left(\frac{1}{2}-\varepsilon\right)^{\nu-u} + \frac{1}{2}\left(\frac{1}{2}-\varepsilon\right)^{u}\left(\frac{1}{2}+\varepsilon\right)^{\nu-u}}{\frac{1}{2^{\nu}}} \geq 1$$

However, in this case, the interval of the u values for which $\mathsf{accept}(u) = 1$ can no more be expressed analytically[24].

3.3.4 Optimal Differential Distinguishers

Similarly, we can briefly study *differential distinguishers* with help of the same statistical tools. A differential distinguisher $\boldsymbol{\delta}^{\nu}_{\text{diff}}$ is a (possibly computationally unbounded) Turing machine which is able to submit chosen pairs of plaintext to an oracle $\boldsymbol{\Omega}$ implementing with probability π_0 a permutation C^* drawn uniformly at random from the set of all permutations on m-bit strings, or with probability π_1, a fixed permutation C (see Alg. 3.8, which is the definition of a differential distinguisher given by Vaudenay [320]). Although the cryptanalytical settings are quite different ($\boldsymbol{\delta}^{\nu}_{\text{diff}}$ can indeed submit *chosen* queries), the distinguishing process is in a statistical point of view very similar to linear distinguishers.

If we look closely at Alg. 3.8, we note that, although the complexity ν is given in advance as input and is (implicitly) fixed, the effective number of

[24] Computing this interval is however possible if we solve this expression numerically.

1: **Input**: An oracle $\mathbf{\Omega}$ implementing an unknown permutation U on $\mathcal{B}$, a complexity ν, a differential (a, b).
2: **Output**: A decision bit.
3: **for** $i = 1$ to ν **do**
4: Pick uniformly at random x and query $\mathsf{U}(x)$ and $\mathsf{U}(x+a)$ to the oracle $\mathbf{\Omega}$.
5: **if** $\mathsf{U}(x + a) = \mathsf{U}(x) + b$ **then**
6: Output "1" and stop.
7: **end if**
8: **end for**
9: Output "0".

Algorithm 3.8: Classical modelization of a differential distinguisher $\boldsymbol{\delta}^{\nu}_{\text{diff}}$.

queries to the oracle is merely a random variable. In other words, depending on the situation, $\boldsymbol{\delta}^{\nu}_{\text{diff}}$ may decide to explicitely ignore some information. In fact, we can see the class of distinguishers submitting a random number of queries to the oracle as a generalization of the class of distinguishers submitting a fixed number of queries. We will study this generalization, called *sequential distinguishers*, in a cryptanalytical context in §3.3.7. In order to better understand the statistical decision process behind differential distinguishers, we in give in Alg. 3.9 an "unorthodox" modelization, denoted $\boldsymbol{\delta}'_{\text{diff}}$, which is very similar to linear distinguishers.

1: **Input**: An oracle $\mathbf{\Omega}$ implementing an unknown permutation U on $\mathcal{B}$, a complexity ν, a differential (a, b), an acceptance function accept.
2: **Output**: A decision bit.
3: Initialize a counter u to zero.
4: **for** $i = 1$ to ν **do**
5: Pick uniformly at random x and query $\mathsf{U}(x)$ and $\mathsf{U}(x+a)$ to the oracle $\mathbf{\Omega}$.
6: **if** $\mathsf{U}(x + a) = \mathsf{U}(x) + b$ **then**
7: Increment u.
8: **end if**
9: **end for**
10: Output $\mathsf{accept}(u)$.

Algorithm 3.9: Unorthodox modelization $\boldsymbol{\delta}'_{\text{diff}}$ of a differential distinguisher.

As outlined in §2.3.3, page 40, differential cryptanalysis depends on the quantity $\mathrm{DP}^{\mathsf{U}}(a, b) = \Pr_X[\mathsf{U}(X + a) = \mathsf{U}(X) + b]$ for a uniformly distributed plaintexts X. We assume now the following statistical scenario: the proba-

bility that the counter u is incremented in Alg. 3.9 is equal[25] to $\frac{1}{2^m-1}$ in the case if $\mathsf{U} = \mathsf{C}^*$, where m is the block-size of the permutation, and to $\frac{1+\varepsilon}{2^m-1}$ if $\mathsf{U} = \mathsf{C}$, for $0 < \varepsilon \leq 2^m - 2$. Applying the optimal Bayesian rule defined in Eq. (3.43), the optimal acceptance function will make $\boldsymbol{\delta}'_{\text{diff}}$ output 1 if

$$\binom{\nu}{u}\left(\frac{1+\varepsilon}{2^m-1}\right)^u\left(1-\frac{1+\varepsilon}{2^m-1}\right)^{\nu-u} \geq \binom{\nu}{u}\left(\frac{1}{2^m-1}\right)^u\left(\frac{2^m-2}{2^m-1}\right)^{\nu-u},$$

where u is the final value of the counter. Reworking algebraically the above expression, we get the following result.

Lemma 3.3.4. *Let* D_0 *be a probability distribution on* $\{0,1\}$ *defined as* $\Pr_{\mathsf{D}_0}[X=0] = 1 - \Pr_{\mathsf{D}_0}[X=1] = \frac{1}{2^m-1}$ *and* D_1 *be a probability distribution defined as* $\Pr_{\mathsf{D}_1}[X=0] = 1 - \Pr_{\mathsf{D}_1}[X=1] = \frac{1+\varepsilon}{2^m-1}$ *with* $0 < \varepsilon \leq 2^m - 2$. *Let* $\boldsymbol{\delta}^\nu$ *be a differential distinguisher limited to* ν *queries implementing an acceptance function* accept *defined by*

$$\mathsf{accept}(u) = 1 \iff u \geq \nu \cdot \frac{\log(2^m-2) - \log(2^m-2-\varepsilon)}{\log((2^m-2)(1+\varepsilon)) - \log(2^m-2-\varepsilon)} \quad (3.55)$$

where $0 \leq u \leq \nu$ *is the number of times that the source outputs 0. Then* $\boldsymbol{\delta}^\nu$ *has a maximal advantage for distinguishing* D_0 *from* D_1.

Note that, for small ε, Eq. (3.55) may be approximated by

$$\mathsf{accept}(u) = 1 \iff u \geq \nu \cdot \left(\frac{1}{2^m-1} + \frac{2^{m-1}-1}{(2^m-2)(2^m-1)} \cdot \varepsilon\right).$$

Finally, if we come back to Alg. 3.8, we note that $\boldsymbol{\delta}_{\text{diff}}$ as defined in Alg. 3.8 is an optimal differential distinguisher submitting n queries to the oracle if and only if Eq. (3.55) is satisfied for all $u \in \mathbb{N}$ with $1 < u \leq \nu$ and for all $0 < \varepsilon \leq 2^m - 2$. Actually, it is not difficult to artificially build a situation where Alg. 3.8 is not optimal: it is sufficient to take a characteristic (a, b) with $\mathrm{DP}^{\mathsf{C}}(a, b)$ true with *very high probability*. In this case, it is not sufficient for $\boldsymbol{\delta}^\nu_{\text{diff}}$ to wait for only *one* differential event and to stop, since if it is unique during the ν samples, it would have been better to output 0. However, if we have a look at Eq. (3.55), we can note that Alg. 3.8 captures well real-world situations, where exploited differential probabilities are usually only slightly larger than ideal ones.

As outlined page 43, we frequently encounter the concept of *signal-to-noise ratio* which was used by Biham and Shamir in the papers defining the differential cryptanalysis [31–33]; it is defined as being the ratio of probability of the right (sub-)key being suggested by a right pair and the probability of a random (sub-)key being suggested by a random pair, given the initial

[25] It is well-known [320] that the expectation over the key space of $\mathrm{DP}^{\mathsf{C}^*}(a, b)$ is equal to $\frac{1}{2^m-1}$, where m is the block-size of the permutation.

difference. By empirical evidence, they suggested that when this ratio is around 1-2, about 20-40 right pairs are sufficient for a successful attack, and when this ratio is higher, even 3-4 right pairs are enough; clearly, this is a (implicitly defined) likelihood-ratio test, which turns out to be optimal in terms of error probabilities. Finally, coming back to the quotation of Nakahara on the same page 43, and at the light of our considerations, it results naturally that there is no better strategy than flipping a coin in the case where the signal-to-noise ratio is equal to 1.

3.3.5 Generalized Linear Distinguishers

As outlined several times in the previous sections, the (idealized) statistical core of linear distinguishers (and therefore, of linear cryptanalysis) consists in sampling a biased binary source generating iid bits. Soon after the publication of [202], several attempts in generalizing linear cryptanalysis have been published. For instance, Kaliski and Robshaw [153] demonstrate how it is possible to combine *several* statistical independent linear approximations involving the same key bits. On their side, Harpes, Kramer and Massey [125] replace the linear approximation by so-called *input-output sums*, i.e. balanced binary-valued functions; they prove the actual effectiveness of such a generalization by exhibiting a block cipher secure against conventional linear cryptanalysis but vulnerable to their attack. Practical examples are the attack of Knudsen and Robshaw [167] against LOKI 91 and the one of Shimoyama and Kaneko [297] against DES; both attacks are based on non-linear probabilistic biased approximations. However, all these attacks sample binary probability spaces.

Another direction to generalize linear cryptanalysis was opened by Vaudenay in [312]: in this paper, he defines another kind of attack against DES, named χ^2-attack, and he demonstrates that one can obtain an attack only slightly less powerful than a linear cryptanalysis, but without the need to describe mathematically what happens *in the core* of the block cipher. Vaudenay's attack is, as its name suggests it, based on a χ^2 statistical test, and interestingly, it can sample probability spaces whose cardinality is larger than two.

Another noteworthy work is the one of Harpes and Massey [126]: in this paper, they generalize the results of [125] by considering *partitions pairs* of the input and output spaces. Let $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ and $\mathcal{Y} = \{\mathcal{Y}_1, \ldots, \mathcal{Y}_n\}$ be partitions of the input and output sets, respectively, where $\mathcal{X}_i$ and $\mathcal{Y}_i$ are called "blocks". Then, the pair $(\mathcal{X}, \mathcal{Y})$ is called a *partition-pair* if all blocks of $\mathcal{X}$ (resp. $\mathcal{Y}$) contain the same number of plaintexts (resp. ciphertexts). A *partitioning cryptanalysis* exploits the fact that the probabilities

$$\Pr\left[(X, \mathsf{f}_k(X)) \in (\mathcal{X}, \mathcal{Y})\right]$$

may not be uniformly distributed for a block cipher f for a fixed key k. In

order to characterize the non-uniformity of a sample distribution, Harpes and Massey consider mainly two "measures" between a discrete probability distribution and the uniform distribution, named *peak imbalance* I_{p} and *squared Euclidean imbalance*, I_{E} defined as follows

$$\mathsf{I}_{\mathrm{p}}(X) = \frac{m}{m-1}\left(\max_{0\leq i<m} \Pr[X=i] - \frac{1}{m}\right) \tag{3.56}$$

and

$$\mathsf{I}_{\mathrm{E}}(X) = \frac{m}{m-1}\sum_{i=0}^{m-1}\left(\Pr[X=i] - \frac{1}{m}\right)^2, \tag{3.57}$$

where X is a random variable distributed according to the sampling distribution and m is the cardinality of the probability space. Given a set of (sub-)key candidates, the attack choose the right (sub-)key as being the one maximizing the chosen bias measure. They report to have observed on toy examples that I_{E} seems to perform better than I_{p}. Harpes and Massey results were completed later by Jakobsen and Harpes [140, 141], where they develop useful bounds to estimate the resistance of block ciphers to partitioning cryptanalysis, with the help of spectral techniques; these bounds are relative to the squared Euclidean imbalance only, but this choice is not motivated in a formal way. To the best of our knowledge, the first practical example of partitioning cryptanalysis breaking a block cipher is the attack known as "stochastic cryptanalysis" [225] proposed by Minier and Gilbert against Crypton [186, 187].

Finally, the NESSIE [247] effort resulted in a few papers investigating the power of linear (or non-linear) approximations based on different algebraic structures, like $\mathbb{Z}_4$. For instance, Parker [259] shows how to approximate constituent functions of an S-box by *any* linear function over *any* weighted alphabet. However, Parker observes that it is not straightforward to piece these generalized linear approximations together. In [303], Standaert et al. take advantage of approximations in $\mathbb{Z}_4$ by *recombining* the values in order to reduce the problem to the well-known binary case; they still obtain more interesting biases comparatively to a classical linear cryptanalysis.

In this part, we address the statistical problems encountered in these works in the same manner than done for classical linear cryptanalysis, and we show that our approach easily allows to derive optimal distinguishers and to compute their complexities in terms of necessary number of samples; for this purpose, we briefly describe some results obtained in collaboration with Thomas Baignères and Serge Vaudenay; we refer to [14] for the proofs.

Let us consider a random source generating a sequence of iid discrete random variables $\boldsymbol{Z}^{\nu} = (Z_1, \ldots, Z_{\nu})$ distributed according either to D_0 or according to D_1, where both probability distributions are defined over a finite set $\mathcal{Z}$; note that we do not require that the cardinality of $\mathcal{Z}$ is equal to 2. Here, we will refer to D_0 has being an "ideal" distribution. Let us now

apply the optimal Bayesian rule defined in Eq. (3.43) to this scenario. Let us denote by $\boldsymbol{z}^\nu = (z_1, \ldots, z_\nu)$ a sample vector obtained from the unknown source. We denote by

$$\mathsf{llr}(\boldsymbol{z}^\nu) = \sum_{a \in \mathcal{Z} : N(a|\boldsymbol{z}^\nu) > 0} N(a|\boldsymbol{z}^\nu) \log \frac{\Pr_{\mathsf{D}_0}[a]}{\Pr_{\mathsf{D}_1}[a]}$$

the corresponding log-likelihood ratio. It is then straightforward to see that the optimal acceptance function is defined by

$$\mathsf{accept}(\boldsymbol{z}^\nu) = 1 \iff \mathsf{llr}(\boldsymbol{z}^\nu) \leq 0$$

where $N(a|\boldsymbol{z}^\nu)$ denotes the number of occurences of the symbol a in the sample $\boldsymbol{z}^\nu \in \mathcal{Z}^\nu$, with the convention that $\log \frac{0}{p} = -\infty$ and $\log \frac{p}{0} = +\infty$. The following result is then a direct application of the central limit theorem (see Th. A.2.1). It makes heavy use of the Kullback-Leibler distance between discrete probability distributions (see Def. B.1.1, page 264).

Lemma 3.3.5. *Let $Z_1, Z_2, \ldots$ be a sequence of iid random variables of distribution D and let D_0 and D_1 be two discrete probability distributions sharing the same support. Then,*

$$\Pr\left[\frac{\mathsf{llr}(\boldsymbol{Z}^\nu) - \nu\mu}{\sigma\sqrt{\nu}} < t\right] \xrightarrow{n\to\infty} \Phi(t),$$

where $\mu = \mu_j$ with $\mu_0 = D(\mathsf{D}_0 \parallel \mathsf{D}_1) \geq 0$ and $\mu_1 = -D(\mathsf{D}_1 \parallel \mathsf{D}_0) \leq 0$, and that σ^2 is

$$\sigma_j{}^2 = \sum_{z \in \mathcal{Z}} \Pr_{\mathsf{D}_j}[z] \left(\log \frac{\Pr_{\mathsf{D}_0}[z]}{\Pr_{\mathsf{D}_1}[z]}\right)^2 - \mu_j^2$$

when $\mathsf{D} = \mathsf{D}_j$ for $j \in \{0, 1\}$.

We now assume that D_0 and D_1 are "close" to each other; this is a frequently encountered situation during a cryptanalysis.

Assumption 3.3.1. *Let D_0 and D_1 be two discrete probability distributions sharing the same support. Then,*

$$\forall z \in \mathcal{Z} \qquad \Pr_{\mathsf{D}_0}[z] = \pi_z \text{ and } \Pr_{\mathsf{D}_1}[z] = \pi_z + \varepsilon_z \text{ with } |\varepsilon_z| \ll \pi_z.$$

Under Ass. 3.3.1, and using a Taylor series argument, we get

$$\mu_0 \approx \mu_1 \approx \frac{1}{2} \sum_{z \in \mathcal{Z}} \frac{\varepsilon_z^2}{\pi_z} \text{ and } \sigma_0^2 \approx \sigma_1^2 \approx \sum_{z \in \mathcal{Z}} \frac{\varepsilon_z^2}{\pi_z}.$$

The following heuristic result gives an estimation of the required number of samples an optimal distinguisher needs, as well as its implied error probability, for distinguishing two close probability distributions D_0 and D_1.

Theorem 3.3.4 (Baignères [15]). *Let $Z_1, \ldots, Z_\nu$ be iid random variables over the set $\mathcal{Z}$ of distribution D, D_0 and D_1 be two discrete probability distributions sharing the same support which are close to each other, and n be the number of samples of an optimal distinguisher between $\mathsf{D} = \mathsf{D}_0$ or $\mathsf{D} = \mathsf{D}_1$. Let d be a real number such that*

$$\nu = \frac{d}{\sum_{z \in \mathcal{Z}} \frac{\varepsilon_z^2}{\pi_z}} \approx \frac{d}{2D(\mathsf{D}_0 \parallel \mathsf{D}_1)}$$

(where $\pi_z = \Pr_{\mathsf{D}_0}[z]$ and $\pi_z + \varepsilon_z = \Pr_{\mathsf{D}_1}[z]$). Then, the overall probability of error is $\pi_{\mathrm{e}} \approx \Phi(-\sqrt{d}/2)$.

Let us now assume that D_0 is the uniform distribution. When D_1 is a distribution whose support is $\mathcal{Z}$ itself and which is close to D_0, Th. 3.3.4 can be rewritten with

$$\nu = \frac{d}{|\mathcal{Z}| \sum_{z \in \mathcal{Z}} \varepsilon_z^2}$$

This shows that the "distinguishability" can be measured by means of the Euclidean distance between D_0 and D_1. In the very specific case where $\mathcal{Z} = \{0, 1\}$, we have $\varepsilon_0 = -\varepsilon_1 = \varepsilon$ and one can see that n is proportional to ε^{-2}, which is a well-known result due to Matsui (see §3.2). We now recall what appears to be the natural measure of the bias of a distribution, considering the needed number of samples and Ass. 3.3.1.

Definition 3.3.2 (Squared Euclidean Imbalance). *Let $\varepsilon_z = \Pr_{\mathsf{D}_1}[z] - \frac{1}{|\mathcal{Z}|}$. The* Squared Euclidean Imbalance (SEI) $\Delta(\mathsf{D}_1)$ *of a distribution D_1 of support $\mathcal{Z}$ from the uniform distribution is defined by*

$$\Delta(\mathsf{D}_1) = |\mathcal{Z}| \sum_{z \in \mathcal{Z}} \varepsilon_z^2.$$

Although the appellation "SEI" coincides with the one of [125], note that the definitions slightly differ. It is well-known (for instance see [121, 159]) that a χ^2 cryptanalysis needs $O(1/\Delta(\mathsf{D}_1))$ queries to succeed, which is by no means worse, up to a *constant term*, than an optimal distinguisher. As recalled in §3.1.2, a χ^2 statistical test is asymptotically equivalent to a generalized likelihood-ratio test developed for a multinomial distribution; although such tests are not optimal in general, they usually perform reasonably well. The above results confirm this fact: a cryptanalyst will not loose any *essential* information in the case she can describe *only one* of the two distributions, but the precise knowledge of *both* distributions allows to derive an optimal attack. In other words, when it is impossible to derive both probability distributions, or when an attack involves many different distributions and only one is known, the best practical alternative to an

optimal distinguisher seems to be a χ^2 attack, as proposed in [312]. This fact corroborates the intuition stipulating that χ^2 attacks are useful when one does not know precisely what happens in the attacked block cipher.

We now assume that random variables are bitstrings, so that $\mathcal{Z} = \{0,1\}^\ell$. According to the notations of Ass. 3.3.1, let D_1 be the probability distribution defined by the set $\{\varepsilon_z\}_{z\in\mathcal{Z}}$, D_0 being the uniform distribution on $\mathcal{Z}$. We define the Fourier transform of D_1 at point $u \in \mathcal{Z}$ as

$$\widehat{\varepsilon}_u = \sum_{z\in\mathcal{Z}} (-1)^{u\cdot z}\ \varepsilon_z\ . \tag{3.58}$$

We note that the involution property of the Fourier transform leads to

$$\varepsilon_z = \frac{1}{2^\ell} \sum_{u\in\mathcal{Z}} (-1)^{u\cdot z}\ \widehat{\varepsilon}_u\ . \tag{3.59}$$

The next lemma can be compared to Parseval's Theorem.

Lemma 3.3.6. *In the case where D_0 is the uniform distribution over $\mathcal{Z} = \{0,1\}^\ell$, the SEI and the Fourier coefficients are related by*

$$\Delta(\mathsf{D}_1) = \sum_{u\in\mathcal{Z}} \widehat{\varepsilon}_u^2.$$

We now recall the definition of the linear probability of a Boolean random variable B:

$$\mathrm{LP}(B) = (\Pr[B=0] - \Pr[B=1])^2 = (2\Pr[B=0]-1)^2 = \left(\mathrm{E}\left[(-1)^B\right]\right)^2.$$

Lemma 3.3.7. *Let $\mathcal{Z} = \{0,1\}^\ell$. If $Z \in \mathcal{Z}$ is a random variable distributed according to D_1, the SEI and the linear probability are related by*

$$\Delta(\mathsf{D}_1)\ =\ \sum_{w\in\mathcal{Z}\setminus\{0\}} \mathrm{LP}(w\cdot Z).$$

Corollary 3.3.2. *Let Z be a random variable over $\mathcal{Z} = \{0,1\}^\ell$ of distribution D_1 and let $\mathrm{LP}^Z_{\max}$ be the maximum of $\mathrm{LP}(w\cdot Z)$ over $w \in \mathcal{Z}\setminus\{0\}$. We have*

$$\Delta(\mathsf{D}_1)\ \le\ \left(2^\ell - 1\right)\ \mathrm{LP}^Z_{\max}.$$

Interpreting Th. 3.3.4 and Cor. 3.3.2 together, we note that the sample complexity of the best distinguisher between two distributions of random bit strings can decrease with a factor up to 2^ℓ when compared to the best linear distinguisher. It is interesting to note that there are cases where this bound is tight. For example if D_1 is such that $\Pr_{\mathsf{D}_1}[z]$ is $\frac{1}{2^\ell} + \left(1-\frac{1}{2^\ell}\right)\gamma$ if $z=0$, and $\frac{1}{2^\ell} - \frac{1}{2^\ell}\gamma$ otherwise (where γ is a positive constant), it can be

shown that $\mathrm{LP}(w \cdot Z) = \gamma^2$ for all $w \neq 0$. Hence $\Delta(\mathsf{D}_1) = (2^\ell - 1)\gamma^2$ and $\mathrm{LP}_{\max} = \gamma^2$.

So far, we have considered the scenario where an optimal distinguishers is fed with two random variables following two distinct distributions in a set $\mathcal{Z} = \{0,1\}^\ell$ where ℓ should not be too large from an implementation point of view. If we try to distinguish two random variables distributed in some set $\{0,1\}^{\ell'}$ of large cardinality (e.g. where $\ell' = 128$), we will not be able to implement the best distinguisher of as the memory requirement would be too high (since we must keep a counter for each possible outcome z). Instead, we can reduce the source space to a smaller space $\mathcal{Z} = \{0,1\}^\ell$ by means of a *projection*[26] $\mathsf{h} : \{0,1\}^{\ell'} \to \mathcal{Z}$ defining, for a random variable $S \in \{0,1\}^{\ell'}$ of distribution $\widetilde{\mathsf{D}}$, a random variable $Z = \mathsf{h}(S)$ of distribution D. Here we consider that h is a balanced function and that $\widetilde{\mathsf{D}}_0$ is a uniform distribution, so that D_0 is the uniform distribution as well. This is a typical construction in a real-life block cipher cryptanalysis, where the block length is quite large. Now, even though we know which distinguisher is the best to use in order to distinguish D_0 from D_1, it is still not clear how the projection h has to be chosen. Probably the most classical example arises when $\ell = 1$ and $\mathsf{h}(S) = a \cdot S$ for some non-zero $a \in \{0,1\}^\ell$. We then talk about a *linear distinguisher* (as discussed in the previous sections). In this case, we note that $\Delta(\mathsf{D}_1) = \mathrm{LP}(a \cdot S) \leq \mathrm{LP}^S_{\max}$. Modern ciphers protect themselves against that type of distinguisher by bounding the value of $\mathrm{LP}^S_{\max}$. A natural extension of the previous scheme would be to consider any linear projection onto wider spaces, e.g. to consider $\mathsf{h}(S) \in \mathcal{Z} = \{0,1\}^\ell$ (where $\ell > 1$ is still small) such that h is GF(2)-linear. We then talk about an *extended linear distinguisher*. It seems natural to wonder about the complexity gap between linear cryptanalysis and this extension. The following result proves that if a cipher provably resists classical linear cryptanalysis, it is (to some extent) protected against extended linear cryptanalysis.

Theorem 3.3.5. *Let S be a random variable over $\{0,1\}^{\ell'}$. Whenever the source space is reduced by a projection $\mathsf{h} : \{0,1\}^{\ell'} \to \{0,1\}^\ell$ in a* GF(2)*-linear way, we have $\Delta(\mathsf{h}(S)) \leq (2^\ell - 1)\mathrm{LP}^S_{\max}$.*

A classical example of a linear space reduction arises when considering *concatenation* of several projections. For example, denoting $\mathsf{D}_1^{(i)} = \mathsf{h}^{(i)}(\widetilde{\mathsf{D}}_1)$ for $i \in \{1,\ldots,\ell\}$ where $\mathsf{h}^{(i)} : \{0,1\}^{\ell'} \to \{0,1\}$ is linear, we consider $\mathsf{h}(S) = \big(\mathsf{h}^{(1)}(S),\ldots,\mathsf{h}^{(n)}(S)\big)$. This corresponds to the works of Kaliski and Robshaw [153] (where different linear characteristics involving *identical* key bits are merged) and to the situation discussed in §3.2.4 (where different linear characteristics involving *different* key bits are merged). In the latter situation, if no assumption is made about the dependency among the $\Delta(\mathsf{D}_1^{(i)})$'s,

[26] We borrow this appellation from Vaudenay [312]; the same expression is used within Wagner's unified view of block cipher cryptanalysis [323] as well.

Th. 3.3.5 tells us $\Delta(\mathsf{D}_1^{(1)} \times \cdots \times \mathsf{D}_1^{(\ell)}) \leq (2^\ell - 1)\mathrm{LP}^S_{\max}$. The following proposition tells us what happens in general when the $\mathsf{D}_1^{(i)}$'s are independent but do not necessarily come from a linear projection nor a Boolean projection.

Theorem 3.3.6. *Consider the case where* $\mathsf{D}_1 = \mathsf{D}_1^{(1)} \times \cdots \times \mathsf{D}_1^{(\ell)}$. *If* $\mathsf{D}_1^{(1)}, \ldots, \mathsf{D}_1^{(\ell)}$ *are* independent *distributions, then*

$$\Delta(\mathsf{D}_1) + 1 = \prod_{i=1}^{\ell} \left(\Delta(\mathsf{D}_1^{(i)}) + 1\right).$$

Therefore, $\Delta(\mathsf{D}_1)$ *can be approximated by the sum of the* $\Delta(\mathsf{D}_1^{(i)})$*'s.*

This result tells us that merging ℓ independent biases should only be considered when their respective amplitudes are within the same order of magnitude. In the light of the preceeding discussion, the cryptanalyst may wonder if it is possible to find a distinguisher with a high advantage even though the value of $\mathrm{LP}^S_{\max}$ is very small. We refer the reader to [14], where an example for which it is indeed the case is provided.

Generalized linear distinguishers have recently been used in [189,190] by Lu and Vaudenay to derive very efficient distinguishing attacks against E0, a stream cipher used in the Bluetooth technology [43].

3.3.6 Aggregate Distinguishers

Optimal distinguishers can be implemented in an efficient way in most situations, i.e. when the cardinality of the underlying probability spaces keeps "reasonably" small. These algorithms, however, are hardly practical if the underlying probability distributions share a support whose cardinality is very large, since they need in this case too large amounts of memory, as we will explain it later. In the following, we briefly describe the construction of suboptimal distinguishing algorithms which may however still be rather efficient in terms of the number of necessary samples and which need practical (and fixed in advance) number of memory cells.

As outlined previously, a natural "measure" of bias is the *squared Euclidean imbalance* $\Delta(\mathsf{D}_0, \mathsf{D}_1)$, between a distribution D_0 and a close distribution D_1 defined by

$$\Delta(\mathsf{D}_0, \mathsf{D}_1) = \sum_{x \in \mathcal{X}} \frac{\varepsilon_x^2}{\pi_x} \tag{3.60}$$

since $\Delta(\mathsf{D}_0, \mathsf{D}_1)$ is, according to Th. 3.3.4, directly linked to the number of sample required for an optimal distinguisher to distinguish both probability distributions with a good success probability. In order to be able to implement an optimal distinguisher, the evaluation of the following likelihood-ratio is required:

$$\mathsf{llr}(\boldsymbol{z}^\nu) = \sum_{x \in \mathcal{X}} \nu_x(\boldsymbol{z}^\nu) \log \frac{\Pr_{\mathsf{D}_0}[x]}{\Pr_{\mathsf{D}_1}[x]}. \tag{3.61}$$

where $\nu_x(\boldsymbol{z}^\nu)$ denotes the number of times the symbol x appears in $\boldsymbol{z}^\nu$. Let us now consider the situation where one knows how to compute efficiently the probabilities $\pi_x = \Pr_{\mathsf{D}_0}[x]$ and $\pi_x + \varepsilon_x = \Pr_{\mathsf{D}_1}[x]$ of a given element x, but where the cardinality of $\mathcal{X}$ is very large. In this case, we can sequentially[27] evaluate the sum of Eq. (3.61) and we do not need to store the value of

$$\log \frac{\Pr_{\mathsf{D}_0}[x]}{\Pr_{\mathsf{D}_1}[x]} \tag{3.62}$$

for each x. Thus, the computational complexity of evaluating Eq. (3.61) is ν times as costly as the (average) time required to evaluate Eq. (3.62), and the memory complexity is negligible.

The situation becomes slightly more complicated in these case where it not possible to evaluate Eq. (3.62) on-the-fly. This can have several reasons: either the evaluation of Eq. (3.62) is too costly, or values were obtained using some heuristic approximations, and they have to be stored, for instance. In this case, the amount of memory needed to manage the counters, and thus to compute Eq. (3.62), may be too large to be possible in practice. For instance, it would be, at the time of writing, quite costly to implement an optimal distinguisher between probability distributions defined on a support $\mathcal{X}$ with $|\mathcal{X}| \gg 2^{32}$. Our goal is thus to construct *sub-optimal* distinguishers taking still a reasonable amount of samples and which do not require too much memory.

In order to get some intuition into our problem, let us consider the following scenario: D_0 is the uniform probability distribution on a set $\mathcal{X}$, and D_1 is a probability distribution defined on the same set $\mathcal{X}$ such that for $\mathcal{X}' \subset \mathcal{X}$, $\mathcal{X}'' \subset \mathcal{X}$, $\mathcal{X}' \cap \mathcal{X}'' = \emptyset$, $\omega = |\mathcal{X}'| = |\mathcal{X}''| \ll \mathcal{X}$, and

$$\Pr_{\mathsf{D}_1}[x] = \frac{1}{|\mathcal{X}|} + \varepsilon \text{ for all } x \in \mathcal{X}', \quad \Pr_{\mathsf{D}_1}[x] = \frac{1}{|\mathcal{X}|} - \varepsilon \text{ for all } x \in \mathcal{X}'',$$

for some $\varepsilon > 0$, and $\Pr_{\mathsf{D}_1}[x] = \frac{1}{|\mathcal{X}|}$ for all $x \notin \mathcal{X}' \cup \mathcal{X}''$. Clearly, the "bias" of the distribution D_1 is concentrated on $\mathcal{X}' \cup \mathcal{X}''$. Let us now define a probability distribution D_1' derived from D_1 as follows: we aggregate all $x \in \mathcal{X}'$ to a single element x', and we assign it the probability $\omega(\frac{1}{|\mathcal{X}|} + \varepsilon)$; similarly, we aggregate all $x \in \mathcal{X}''$ to a single element x'', and we assign it the probability $\omega(\frac{1}{|\mathcal{X}|} - \varepsilon)$; finally, we aggregate all $x \notin \mathcal{X}' \cup \mathcal{X}''$ to a single element x^* and we assign it the probability $1 - \frac{2\omega}{|\mathcal{X}|}$. Similarly, we build a new probability distribution out of D_0 by assigning the probability $\frac{\omega}{|\mathcal{X}|}$ to x', the probability $\frac{\omega}{|\mathcal{X}|}$ to x'', and $1 - \frac{2\omega}{|\mathcal{X}|}$ to x^*. The intuition behind this construction is the following: one concentrates the distinguishing efforts on biased elements of D_1, since other elements do not give any information. Clearly, an optimal

[27] Thus, we do not even need to store $\boldsymbol{z}^\nu$ nor to manage counters $\nu_x(\boldsymbol{z}^\nu)$ for each possible x.

distinguisher aiming at distinguishing D_0 from D_1 will need, given a fixed overall success probability, exactly the same amount of samples than for distinguishing D'_0 from D'_1 (i.e. $(2|\mathcal{X}|\omega\varepsilon^2)^{-1}$) but interestingly, the underlying probability distributions are far simpler. If we cannot easily compute on-the-fly the values of Eq. (3.62), then we can restrict ourselves to manage 2ω (instead of $|\mathcal{X}|$) memory cells (i.e. by storing in a table for an element $x \in \mathcal{X}' \cup \mathcal{X}''$ the value $\kappa \in \{-1, 1\}$ of its probability $\frac{1}{|\mathcal{X}|} + \kappa\varepsilon$, the elements not present in the table being assigned the probability $\frac{1}{|\mathcal{X}|}$). Thus, by defining a distinguisher interpreting D_0 and D_1 as D'_0 and D'_1, respectively, one can decrease the memory needs without any lost of distinguishing power.

This extreme example leads naturally to the concept of *aggregate distinguisher*. Let us assume now that we have enough memory to manage ω cells. Informally, we can proceed as follows: we derive two probability distributions D'_0 and D'_1 from D_0 and D_1, respectively, such that D'_0 and D'_1 share a support $\mathcal{Y}$ with $\omega = |\mathcal{Y}| \ll |\mathcal{X}|$ and such that the distinguishing power loss is minimized; furthermore, we need to define an explicite surjection $\mu : \mathcal{X} \to \mathcal{Y}$ which maps each element of $\mathcal{X}$ to an element of $\mathcal{Y}$. The natural idea consists then in *aggregating* elements with low biases to form new elements of the derived distributions.

Let us assume now that we would like to aggregate two elements $x, y \in \mathcal{X}$ with $x \neq y$ to a single one named $x||y \in \mathcal{Y}$. The loss of contribution to the sum of Eq. (3.60) can be expressed as

$$\frac{\varepsilon_x^2}{\pi_x} + \frac{\varepsilon_y^2}{\pi_y} - \frac{(\varepsilon_x + \varepsilon_y)^2}{\pi_x + \pi_y} = \frac{(\pi_y\varepsilon_x - \pi_x\varepsilon_y)^2}{\pi_x\pi_y(\pi_x + \pi_y)}.$$

This quantity must obviously be minimized in order to get the best possible aggregate in terms of $\Delta(\mathsf{D}'_0, \mathsf{D}'_1)$. These considerations naturally lead to Alg. 3.10, which can be viewed as a derivation of an optimal encoding procedure for Huffman codes [135] and whose optimality follows from the preceeding considerations by induction.

Theorem 3.3.7. *Let $\omega > 1$ be a fixed integer; let D_0 and D_1 be two discrete probability distributions sharing the same support $\mathcal{X}$. Then, Alg. 3.10 minimizes*

$$\Delta(\mathsf{D}_0, \mathsf{D}_1) - \Delta(\mathsf{D}'_0, \mathsf{D}'_1)$$

where D'_0 and D'_1 are the two probability distributions defined by $\mathcal{S}$ and μ.

Finally, we must note a quite annoying property of Alg. 3.10 which could render it useless in certain situations: namely, for each "aggregate", we must store somewhere the corresponding part of the mapping μ. Unfortunately, the memory required to store the mapping may be (in certain cases) as large as the memory required to store the whole information about the probability distribution under scrutiny. However, when applied to probability distributions having a small number of elements with a highly biased probability,

Input: D_0, D_1, and ω.
Output: D'_0, D'_1 such that $|\mathsf{D}'_0| = |\mathsf{D}'_1| = \omega$ and $\Delta(\mathsf{D}_0, \mathsf{D}_1) - \Delta(\mathsf{D}'_0, \mathsf{D}'_1)$ is minimal, and a surjective mapping $\mu : \mathcal{X} \to \mathcal{Y}$.
Let $\mathcal{S} = \mathcal{X}$.
while $|\mathcal{S}| > \omega$ **do**
    Let $x, y \in \mathcal{S}$ with $x \neq y$ such that $\frac{(\pi_y \varepsilon_x - \pi_x \varepsilon_y)^2}{\pi_x \pi_y (\pi_x + \pi_y)}$ is minimal.
    Remove x and y from $\mathcal{S}$, replace them with the element labelled $x||y$, and assign it the probability $\pi_x + \pi_y + \varepsilon_x + \varepsilon_y$. Add furthermore $x \mapsto x||y$ and $y \mapsto x||y$ to the mapping μ.
end while

Algorithm 3.10: Derivation of Optimal Aggregates

then aggregate distinguishers are definitely useful. We may even think about situations where one can explicitely find sub-optimal distinguishers which do not require much memory to store the mapping μ, but it remains an open problem to us at the time of writing.

3.3.7 Sequential Distinguishers

Input: An oracle $\mathbf{\Omega}$ implementing an unknown permutation U, a complexity ν, acceptance functions $\mathsf{accept}_i, 1 \leq i \leq \nu$, and rejection functions $\mathsf{reject}_i, 1 \leq i \leq \nu - 1$.
Output: A decision bit.
$i \leftarrow 1$
repeat
    Select non-adaptively a message x_i and get $y_i = \mathsf{U}(x_i)$.
    if $\mathsf{accept}(y_1, \ldots, y_i) = 1$ **then**
        Output "1" and stop.
    else if $\mathsf{reject}(y_1, \ldots, y_i) = 0$ **then**
        Output "0" and stop.
    end if
    $i \leftarrow i + 1$
until $i = \nu - 1$
Select non-adaptively a message x_ν and get $y_\nu = \mathsf{U}(x_\nu)$.
if $\mathsf{accept}(y_1, \ldots, y_\nu) = 1$ **then**
    Output "1".
else
    Output "0".
end if

Algorithm 3.11: A ν-limited sequential non-adaptive distinguisher

If we look at Alg. 3.8, we can observe that, although the complexity ν

is given in advance as input and is (implicitly) *fixed*, the effective number of queries to the oracle is merely a random variable. In other words, $\boldsymbol{\delta}_{\text{diff}}$ does not make use of all the information that it could exploit. In fact, we can see the class of distinguishers submitting a *random* number of queries to the oracle as a generalization of the class of distinguishers submitting a *fixed* number of queries. We will call this generalization *sequential distinguishers*. In this part, we formalize the concepts of *sequential non-adaptive distinguisher (SNAD)* and of *ν-limited sequential non-adaptive distinguisher (ν-limited SNAD)*. These kinds of distinguishers use sequential sampling procedures as their statistical core. The only example of an advanced attack based on such a sequential procedure we are aware of is an attempt of Davies and Murphy described in the appendix of [83] to decrease the complexity of their non-surjetcive attack against DES.

In the Luby-Rackoff model, a non-adaptive adversary (which may be modelized by an ν-limited adaptive distinguisher as described in Algorithm 3.4) is an infinitely powerful Turing machine which has access to an oracle $\boldsymbol{\Omega}$. It aims at distinguishing a cipher C from the "ideal cipher" C^* by querying $\boldsymbol{\Omega}$, and with a limited number ν of inputs. The attacker must finally take a decision; usually, one is interested in measuring the ability (i.e. the advantage as defined in Eq. (3.42)) to distinguish C from C^* for a given, *fixed* amount ν of queries. Clearly, in this model, we are interested in maximizing the advantage under the constraint defined by the number of allowed queries. In the real life, a cryptanalyst proceeds usually in an inverse manner: given a fixed success probability (i.e. a fixed advantage), she may look for minimizing the amount of queries to $\boldsymbol{\Omega}$, since such queries are typically costly to obtain in practice. With this model in head, we can now define a *ν-limited sequential non-adaptive distinguisher* (see Alg. 3.11), which actually turns out to be more efficient in terms of the average number of oracle queries than Alg. 3.5 given a fixed advantage. In fact, such a distinguisher implements an adaptive decision process. Namely, after having received the i-th response from the oracle, the distinguisher compares the i responses it has at disposal towards an acceptance set $\mathcal{A}_i$ (defined by the Boolean function accept_i in Alg. 3.5) and a rejection set $\mathcal{R}_i$ (defined by the Boolean function reject_i), which depends on the number of queries and on the (fixed in advance) advantage, and can then take *three* different decisions: either it decides to output "0" and to stop, or to output "1" and to stop, or to query one more question to the oracle and to repeat the decision process, until it has queried d questions. Note that $\mathcal{A}_i \subseteq \mathcal{B}^i$ and $\mathcal{R}_i \subseteq \mathcal{B}^i$ must be disjoint sets for all $1 \leq i < \nu$ and that $\mathcal{A}_\nu \cup \overline{\mathcal{A}_\nu} = \mathcal{B}^\nu$, which means that the functions accept_i and reject_i must define consistent decisions. In statistics, this process is known as a *sequential decision procedure*[28]. We finally note that

[28] Historically, in very general terms, the concept of sequential hypothesis testing came into existence simultaneously in the United States and Great Britain in response to de-

Alg. 3.8 can be interpreted as a sequential differential distinguisher which does not take explicitely into account intermediate decision functions, since it always outputs "1" as soon as it observes a "differential event".

Sequential Statistical Inference

We describe now formally the *sequential decision procedure* behind Alg. 3.11. Let $\mathcal{D}$ be the set of possible decisions.

Definition 3.3.3 (Sequential decision procedure). *Let $X_1, X_2, \ldots$ be random variables observed sequentially. A* sequential decision procedure *consists in:*

1. *a* stopping rule σ_i *which specifies whether a decision must be taken without taking any further observation at step i. If at least one observation is taken, this rule specifies for every set of observed values $(x_1, \ldots, x_i)$, with $i \geq 1$, whether to stop sampling and take a decision out of $\mathcal{D}$ or to take another observation x_{i+1}.*

2. *a* decision rule δ_i *which specifies the decision to be taken. If $i \geq 1$ observations have been taken, then one takes an action $\delta_\nu(x_1, \ldots, x_\nu) \in \mathcal{D}$. Once a decision has been taken, the sampling process is stopped.*

If we consider Alg. 3.11 at the light of Def. 3.3.3, $\mathcal{D} = \{0, 1\}$,

$$\sigma_i(x_1, \ldots, x_i) = \begin{cases} \text{Continue sampling if} & \mathsf{accept}_i(x_1, \ldots, x_i) = 0 \text{ and} \\ & \mathsf{reject}_i(x_1, \ldots, x_i) = 1 \\ \text{Stop sampling if} & \mathsf{accept}_i(x_1, \ldots, x_i) = 1 \text{ or} \\ & \mathsf{reject}_i(x_1, \ldots, x_i) = 0 \end{cases}$$

and

$$\delta_i(x_1, \ldots, x_\nu) = \begin{cases} 0 \text{ if } \mathsf{reject}(x_1, \ldots, x_i) = 1 \\ 1 \text{ if } \mathsf{accept}(x_1, \ldots, x_i) = 1 \end{cases}.$$

We are now interested in applying sequential decision procedures in the context of binary hypothesis tests and of the Neyman-Pearson paradigm. We have seen that Lem. 3.1.1 allows us to define precisely the shape of the optimal acceptance regions in the case of binary hypothesis tests (for a simple hypothesis versus a simple alternative). Theoretically, if we would be able to compute the exact joint probability distribution of the oracle's responses when it implements both ciphers, we would be able to compute the optimal acceptance region $\mathcal{A}$ for any ν-limited distinguisher. Practically, one should notice that it seems considerably easier to compute joint probability distributions when the distinguisher is *non-adaptive*, since one can use

mands for more efficient sampling inspection procedures during World War II, and all these developments were summarized by their principal architect, Wald, in [324]. We refer to [299] for an excellent treatment of this subject.

some (maybe heuristic) statistical independence assumptions. A *sequential likelihood-ratio test* uses exactly the same process to define *two* sets of acceptance regions, denoted $\mathcal{A}_i$ and $\mathcal{R}_i$, respectively. So, it is always possible to define a sequential test when one has a classical test at disposal. In few words, a sequential test has three alternatives once it has received a response from the oracle: either it can conclude for one of both hypotheses, or it can decide to query more samples. In its simpler definition, a sequential ratio test has the possibility to query *as many samples as it is needed to take a decision*, given a fixed error probability. The *expected* number of queries required to reach one of the two possible decision turns out to be less than it would need in order to make the same decision on the basis of a single *fixed-size* sample set. Of course it may happen that the sequential procedure will take more queries than the fixed-size one, but sequential sampling is a definitely economical procedure. To fit into our framework, we may interpret Alg. 3.11 as a *truncated sequential test*, i.e. one fixes an upper-bound d on the number of queries; it is still clear that such a sequential procedure cannot be worse than a fixed-size sampling procedure.

We state now some well-known definitions and results about sequential hypothesis tests.

Definition 3.3.4 (Sequential Likelihood-Ratio Test). *To test* $\boldsymbol{X} \leftarrow \mathsf{D}_0$ *against* $\boldsymbol{X} \leftarrow \mathsf{D}_1$*, define two constants* $\tau_{\text{up}} > \tau_{\text{down}} > 0$ *depending on* α *and* β*, and define the likelihood ratio*

$$\mathsf{lr}(\boldsymbol{x}) = \frac{\mathsf{f}_{\boldsymbol{X}_1}(\boldsymbol{x})}{\mathsf{f}_{\boldsymbol{X}_0}(\boldsymbol{x})}$$

The decision function at i*-th step is*

$$\boldsymbol{\delta}_{\text{opt}} = \begin{cases} 1 & (i.e \text{ accept } \boldsymbol{X} \leftarrow \mathsf{D}_1) \quad \text{if } \mathsf{lr}(\mathbf{x}^{(i)}) \geq \tau_{\text{up}} \\ 0 & (i.e. \text{ accept } \boldsymbol{X} \leftarrow \mathsf{D}_0) \quad \text{if } \mathsf{lr}(\boldsymbol{x}^{(i)}) \leq \tau_{\text{down}} \\ \emptyset & \text{query another sample} \quad \text{otherwise} \end{cases} \tag{3.63}$$

When the observations are *independent and identically distributed*, then sequential likelihood-ratio tests have a nice property, as stated by Th. 3.3.8. We refer the reader to the book of Siegmund [299] for an excellent treatment of sequential procedures and for the proof of the three following theorems.

Theorem 3.3.8. *For testing a simple hypothesis against a simple alternative with independent, identically distributed observations, a sequential probability ratio test is optimal in the sense of* minimizing the expected sample size *among all tests having no larger error probabilities.*

The following results relate error probabilities α and β to τ_{up} and τ_{down}, and give an approximation of the expected number of samples.

Theorem 3.3.9. *Let us consider a sequential likelihood-ratio test with stopping bounds τ_{up} and τ_{down}, with $\tau_{\text{up}} > \tau_{\text{down}}$ and error probabilities $0 < \alpha < 1$ and $0 < \beta < 1$, then*

$$\tau_{\text{down}} \geq \frac{\beta}{1-\alpha} \quad \text{and} \quad \tau_{\text{up}} \leq \frac{1-\beta}{\alpha}$$

The approximation $\tau_{\text{down}} = \frac{\beta}{1-\alpha}$ and $\tau_{\text{up}} = \frac{1-\beta}{\alpha}$ is known as "Wald's approximation". The following theorem gives some credit to this approximation.

Theorem 3.3.10. *Let us assume we select for given $\alpha, \beta \in]0,1[$, where $\alpha + \beta \leq 1$, the stopping bounds $\tau'_{\text{down}} = \frac{\beta}{1-\alpha}$ and $\tau'_{\text{up}} = \frac{1-\beta}{\alpha}$. Then it holds that the sequential likelihood-ratio test with stopping bounds τ'_{down} and τ'_{up} has error probabilities α' and β' where*

$$\alpha' \leq \frac{\alpha}{1-\beta}, \quad \beta' \leq \frac{\beta}{1-\alpha} \quad \text{and} \quad \alpha' + \beta' \leq \alpha + \beta$$

Let us denote by N the random variable modelizing the number of samples queried to $\mathbf{\Omega}$ before Alg. 3.11 stops and outputs a decision bit. By taking into account Wald's approximation, we can easily compute approximations of the expected number of queries:

$$\mathrm{E}_{\boldsymbol{X}_0}[N] \approx \frac{\alpha \log\left(\frac{1-\beta}{\alpha}\right) + (1-\alpha)\log\left(\frac{\beta}{1-\alpha}\right)}{\mathrm{E}_{X_0}[\log(\mathsf{f}_{X_1}(x)) - \log(\mathsf{f}_{X_0}(x))]}$$

where $\mathrm{E}_{\boldsymbol{X}_i}[.]$ denotes that the expectation is taken over D_i and

$$\mathrm{E}_{\boldsymbol{X}_1}[N] \approx \frac{(1-\beta)\log\left(\frac{1-\beta}{\alpha}\right) + \beta\log\left(\frac{\beta}{1-\alpha}\right)}{\mathrm{E}_{X_1}[\log(\mathsf{f}_{X_1}(x)) - \log(\mathsf{f}_{X_0}(x))]}.$$

Application of Sequential Distinguishers

We now illustrate the usefulness of sequential distinguishers in a cryptanalytic context using two examples. The first one consists in turning Matsui's First Algorithm Alg. 3.1 in a sequential procedure, while the second example is the optimization of Canvel *et al.*timing attack against SSL [57].

A Toy-Example on DES In order to illustrate advantages of sequential linear distinguishers, we have implemented a linear cryptanalysis of DES reduced to five rounds which uses a sequential distinguisher for deciding the parity of the linear approximation, i.e. the parity of the sum of involved key bits. Using a static test (i.e. using Matsui's First Algorithm Alg. 3.1), we need roughly 2800 known plaintext-ciphertext pairs in order to get a success probability of 97 %. Using a sequential strategy and for the same success

	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5
Pr[Alg. 3.1 succ.]	0.9689	0.9687	0.9684	0.9686	0.9688
Pr[Seq. dist. succ.]	0.9686	0.9684	0.9683	0.9682	0.9684
Av. number of queries	1218.7	1218.7	1218.3	1219.1	1218.8

Figure 3.8: Sequential Linear Cryptanalysis of 5-Rounds DES.

probability, only 1218 samples were necessary on average. We describe now both the static and the sequential decision rules.

Let S_ν denote the number of times that Matsui's best linear characteristic [202] on 5-rounds DES evaluates to 0, where n is the number of known plaintext-ciphertext pairs at disposal. This linear approximation holds with probability $\frac{1}{2} + 0.01907$. According to Th. 3.2.6, the optimal static decision rule is then defined to make Alg. 3.1 output the message "key parity = 0" if and only if

$$S_\nu \geq \frac{\nu}{2},$$

and to output the message "key parity = 1" if and only if

$$S_\nu < \frac{\nu}{2}$$

With 2800 known pairs at disposal, the static rule is successful in about 97% of the cases.

Let us now focus on defining a sequential rule to perform the same task. For $\alpha = \beta = 0.025$, Wald's approximation gives $\tau_{\text{up}} = 48$ and $\tau_{\text{down}} = \frac{1}{48}$. Let furthermore $\varepsilon = 0.01907$. Then the sequential rule is then defined as follows: in case

$$S_\nu \leq \frac{\nu}{2} - \frac{\log \tau_{\text{up}}}{2 \log \left(\frac{1+2\varepsilon}{1-2\varepsilon} \right)} \tag{3.64}$$

then output the message "key parity = 1" and stop; in case

$$S_\nu \geq \frac{\nu}{2} + \frac{\log \tau_{\text{down}}}{2 \log \left(\frac{1-2\varepsilon}{1+2\varepsilon} \right)} \tag{3.65}$$

then output the message "key parity = 0" and stop; in case where *none* of the conditions of Eq. (3.64) and of Eq. (3.65) are statisfied, then query another sample.

We repeated this experiment 1'000'000 times for 5 different keys and got the results tabulated in Fig. 3.8. Basically, we need more than two times less samples when using a sequential rule than when using a static rule.

Optimization of the Timing Attack against SSL/TLS The SSL (Secure Socket Layer) and its most modern version, named TLS (Transport Layer Security), are certainly the most important protocols used to build secure communications over the Internet (see [39, 90] for their formal specifications). It consists basically in first negotiating a security parameters and a *cipher suite*, i.e. a set of cryptographic algorithms which the two communicating entities accept to use for the sequel of the communication, and second, in exchanging secret keys. Once this setup step is performed successfully, the communicating entities are able to exchange arbitrary messages which will typically be authenticated using a Message Authentication Code (MAC), and then encrypted using a block cipher. In case where the message concatenated with the MAC is not long enough to fit in a multiple of the block length of the cipher, then a padding operation is necessary which consists in adding enough (fixed and known) supplementary bytes. Recently, Vaudenay exhibited in [319] a side-channel attack based on the knowledge whether or not the padding removal operation during the decryption was successful or not. Provided this side information is available, this attack allows an adversary to decrypt any block of data without knowledge of the key, and is thus very powerful.

Unfortunately, such a side-channel is not (directly) available in the context of SSL/TLS, since the error messages are encrypted. However, in order to get access to this information anyway, Canvel *et al.* [57] observed that it is possible to obtain it using a timing attack (first proposed by Kocher in [170] against public-key cryptosystems). Namely, in order to check whether the padding is correct, the server only needs to perform simple operations on the very end of the ciphertext. If the padding is correct, the server further has to perform time-expensive cryptographic operations on the whole length of the ciphertext in order to check its authenticity (by re-computing a MAC), and this typically requires more time. Thus, if an adversary is able to observe this small time shift, it can finally get the needed information.

Provided the adversary is relatively close to the server (i.e. in terms of network distance), she can measure the elapsed time several times, and observe the sampling distribution in both cases. If the noise due to external conditions keeps relatively low, we can expect the adversary be able to distinguish the two distributions. Fig. 3.9 represents timing measurements obtained by Canvel *et al.* on a LAN where a simple firewall was present between the adversary and the server. They report to have modelled these sampling distributions by two normal distributions whose parameters where inferred from statistical measurements.

Motivated by our work, Canvel *et al.* report to have implemented both a static and a sequential distinguisher, and they observed that the latter was four to five time faster depending on the experimental conditions, rendering the attack very practical.

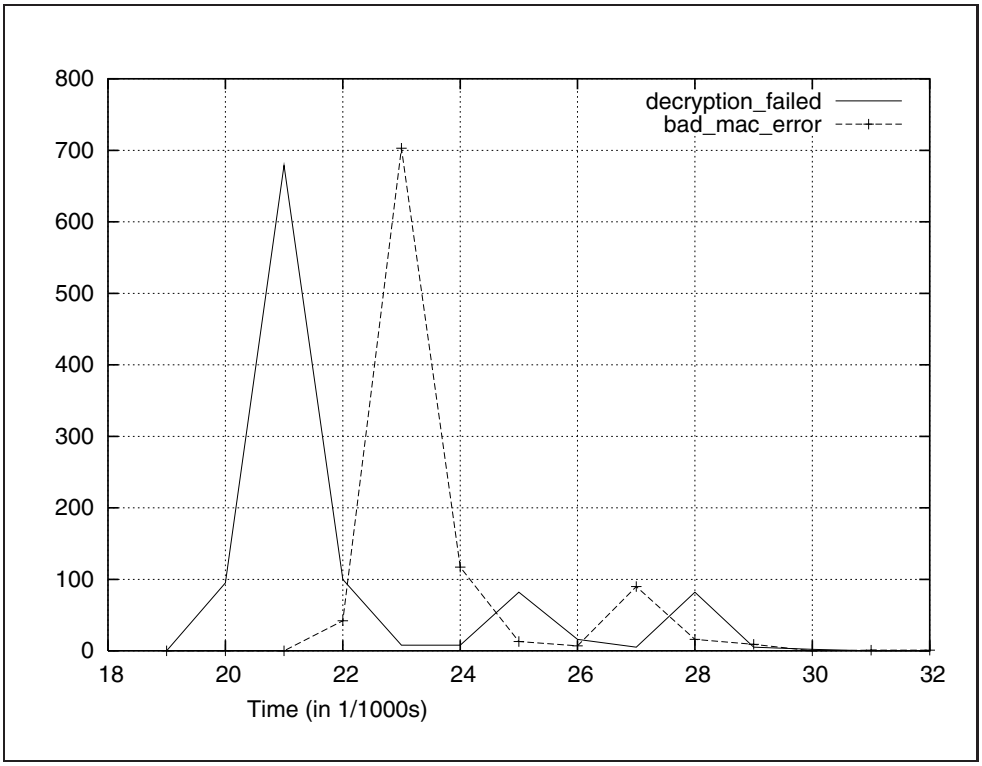

Figure 3.9: Timing measurement in Canvel *et al.* attack against SSL/TLS

3.3.8 Summary

In this section, we have shown that it is possible to interpret the Luby-Rackoff model of security as a statistical hypothesis problem. This has allowed us to derive the optimal shape of several types of distinguishers, to bound tightly their best advantage, and to improve some existing results under slightly weakened assumptions. Furthermore, we have presented some results about the generalization of linear distinguishers, which are allowed here to sample random sources of greater cardinality than 2. Then, we have discussed how to improve the practicability of optimal distinguishers in certain cases by applying the idea to derive more friendly probability distribution using the aggregation of elements; finally, we have shown that the concept of sequential sampling has interesting properties in the context of cryptanalysis.

3.4 New Linear-Like Attacks against IDEA

In this section, we present a sequence of new attacks against reduced-round versions of IDEA, up to 4 rounds (out of 8.5). These attacks are mainly based on the Biryukov-Demirci relation proposed by Nakahara *et al.* [240]. Some

of them, given a comparable computational complexity, reduce considerably the amount of necessary chosen plaintexts, while other attacks, given a comparable amount of chosen plaintexts, decrease favorably the computational complexity; none of our attacks need large amounts of memory. Furthermore, we show how to use some of these attacks in combination with other known attacks, which allows in certain cases to gain more key bits with a lesser complexity, or to avoid the use of both encryption and decryption oracles. We refer the reader to §2.2.2 for a description of the IDEA block cipher and for a discussion about the existing attacks against it. In the sequel, we will make use of the notations defined in Fig. 2.10, page 22: the i-th input word at round r is denoted $x_i^{(r)}$, the i-th output word is denoted $y_i^{(r)}$, and the i-th word after a half-round (i.e. after a key addition layer) is denoted $c_i^{(r)}$; finally, the left and right input words of the MA-box at round r are denoted $\alpha^{(r)}$ and $\beta^{(r)}$, respectively, while the output words are denoted $\gamma^{(r)}$ and $\delta^{(r)}$.

3.4.1 The Biryukov-Demirci Relation

A crucial observation on which our attacks is based is that there exists a linear-like expression holding with probability one on *any* number of rounds. Nakahara *et al.* [240] name it *Biryukov-Demirci relation*. It is actually a combination of two facts, one of these being the following observation by Demirci [84].

Lemma 3.4.1 (Demirci [84]). *For any number r of round(s) in the block cipher* IDEA*,*

$$\mathrm{lsb}\left(\gamma^{(r)} \oplus \delta^{(r)}\right) = \mathrm{lsb}\left(\alpha^{(r)} \odot k_5^{(r)}\right) \tag{3.66}$$

where $\mathrm{lsb}(a)$ *denotes the least significant (rightmost) bit of* a*.*

Using this theorem, one can easily set up a distinguisher using a few known triplets $(\alpha^{(r)}, \gamma^{(r)}, \delta^{(r)})$ which works as follows: for each possible value of $k_5^{(r)}$, check whether Eq. (3.66) hold for a certain number of known triplets; this allows to sieve wrong values of $k_5^{(r)}$ from the right one. Actually, one gets *two* candidates for $k_5^{(r)}$, as observed by Demirci: if $k_5^{(r)} \notin \{0, 1\}$, this distinguisher eliminates all keys except the correct one and a "conjugate" $2^{16} + 1 - k_5^{(r)}$. Otherwise, it eliminates all keys except 0 and 1.

The second (unpublished) observation[29] states that the two middle words in a block of data are only combined, either with subkeys or with internal cipher data, via group operations (namely $\oplus$ and $\boxplus$) which are GF(2)-linear when considering their least significant (rightmost) bit; this fact is valid across the full cipher (and is actually independent of the number of rounds).

[29] According to [240], this observation is due to Biryukov.

Combining this observation and Lem. 3.4.1, one easily obtain the *Biryukov-Demirci relation.*

Theorem 3.4.1 (Biryukov-Demirci relation). *For any number r of round(s) in the block cipher* IDEA*, the following expression is true with probability one:*

$$\mathrm{lsb}\left(\bigoplus_{i=1}^{n}\left(\gamma^{(i)}\oplus\delta^{(i)}\right)\oplus x_2^{(1)}\oplus x_3^{(1)}\oplus y_2^{(n+1)}\oplus y_3^{(n+1)}\right)= \mathrm{lsb}\left(\bigoplus_{j=1}^{n}\left(k_2^{(j)}\oplus k_3^{(j)}\right)\right)$$

Note that Th. 3.4.1 can easily be extended when a final half-round (key-addition layer) is present by adding the two relevant key bits.

3.4.2 Retrieving All Key Bits for 1.5 Rounds

The simplest attack described in [240] is built on top of the following expression holding with probability one; it is a straightforward application of Th. 3.4.1 to 1.5-rounds IDEA.

$$\mathrm{lsb}\left(x_2^{(1)}\oplus x_3^{(1)}\oplus c_2^{(2)}\oplus c_3^{(2)}\oplus k_2^{(1)}\oplus k_3^{(1)}\oplus k_2^{(2)}\oplus k_3^{(2)}\oplus k_5^{(1)}\odot\left(\left(x_1^{(1)}\odot k_1^{(1)}\right)\oplus\left(x_3^{(1)}\boxplus k_3^{(1)}\right)\right)\right)=0 \tag{3.67}$$

By taking into account the key-schedule algorithm and guessing key bits numbered (see Fig. 2.11) 0-15, 32-47, 64-79, which represent 48 unknown key bits, one can recover these right key bits and $\mathrm{lsb}\left(k_2^{(1)}\oplus k_2^{(2)}\right)$ with probability larger than 0.99 in roughly $3\cdot\frac{12}{30}\cdot 2^{48}\approx 2^{48.26}$ 1.5-rounds IDEA evaluations if 55 known plaintext-ciphertext pairs are available using Alg. 3.12. The complexity of this attack can be evaluated as follows: for each key candidate, one needs to evaluate Eq. (3.68) at least two times, three times with probability $\frac{1}{4}$, four times with probability $\frac{1}{8}$, and so on, which results in an average of three evaluations of Eq. (3.68); as in [240], we assume furthermore that a $\odot$ operation is equivalent to three $\oplus$ (or three $\boxplus$) operations: thus, one evaluation of Eq. (3.68) costs 12 simple operations while a full evaluation of 1.5-round IDEA costs 30 simple operations. Note that we may have adopted the strategy of [240], which consists in guessing $\mathrm{lsb}\left(k_2^{(1)}\oplus k_2^{(2)}\right)$ as well and evaluating Eq. (3.67). In this case, one would need one pair of known plaintext-ciphertext more to ensure the same success probability, and the complexity would have been equal to $2\cdot\frac{14}{30}\cdot 2^{49}\approx 2^{48.90}$, which is slightly worse.

1: **Input**: An oracle $\mathbf{\Omega}$ implementing encryption by 1.5-rounds IDEA under a fixed, unknown key.
2: Query the ciphertexts corresponding to 55 different, uniformly distributed plaintexts P_i to $\mathbf{\Omega}$.
3: **for** all possible subkey candidates $(k_1^{(1)}, k_3^{(1)}, k_5^{(1)})$ **do**
4: Check whether the expression

$$\begin{aligned} &\mathrm{lsb}\left(x_2^{(1)} \oplus x_3^{(1)} \oplus c_2^{(2)} \oplus c_3^{(2)} \oplus \right. \\ &\left. k_5^{(1)} \odot \left(\left(x_1^{(1)} \odot k_1^{(1)}\right) \oplus \left(x_3^{(1)} \boxplus k_3^{(1)}\right)\right)\right) \end{aligned} \tag{3.68}$$

gives the same bit for the two first pairs. If yes, take sequentially other pairs as long as Eq. (3.68) evaluates to a constant. If it holds for all 55 pairs, output "Key candidate".
5: **end for**

Algorithm 3.12: Attack breaking 1.5-round IDEA

We observe that it is actually possible to apply a common trick[30] to the Biryukov-Demirci relation and thus extend Nakahara *et al.* attack: we can apply the relation in *two directions*, namely in the encryption or in the decryption direction. When applied to the decryption direction, the distinguisher Eq. (3.67) becomes

$$\begin{aligned} &\mathrm{lsb}\left(x_2^{(1)} \oplus x_3^{(1)} \oplus c_2^{(2)} \oplus c_3^{(2)} \oplus k_2^{(1)} \oplus k_3^{(1)} \oplus k_2^{(2)} \oplus k_3^{(2)} \oplus \right. \\ &\left. k_5^{(1)} \odot \left(\left(c_1^{(2)} \odot k_1^{(2)}\right) \oplus \left(c_2^{(2)} \boxplus k_2^{(2)}\right)\right)\right) = 0 \end{aligned} \tag{3.69}$$

Although it would not be more interesting to use Eq. (3.69) as single distinguisher (since one should the same number of unknown key bits), one can use it after Eq. (3.67) to recover *all key bits* using roughly the same amount of computational effort. More precisely, once the key bits 0-15, 32-47, 64-79 are known, which actually fix $k_1^{(1)}$ and $k_5^{(1)}$, one can recover $k_1^{(2)}$ and $k_2^{(2)}$ (key bits numbered 96-127) in a $3 \cdot \frac{12}{30} \cdot 2^{32} \approx 2^{32.26}$ effort, derive the key bit 31, and search exhaustively for the remaining 47 unknown key bits. The overall complexity of this attack is approximately equal to $2^{48.26} + 2^{47} + 2^{32.26} \approx 2^{48.76}$ 1.5-round IDEA evaluations.

[30] This trick was proposed for the first time, to the best of our knowledge, by Matsui [203] in the linear cryptanalysis of DES.

3.4.3 A New Chosen-Plaintext Attack Breaking 2 Rounds

Let us consider the relation Eq. (3.67) on 2 rounds, and let us fix $x_1^{(1)}$ and $x_3^{(1)}$ to arbitrary constants[31]. Our attack proceeds as follows and assumes that the adversary is able to encrypt about 62 chosen plaintexts: as first step, encrypt 23 chosen plaintexts with fixed $x_1^{(1)}$ and $x_3^{(1)}$, and guess $k_5^{(2)}$. In a second step, guess $k_6^{(2)}$ and test Eq. (3.67) on the partially decrypted ciphertext, and determine these unknown key bits with help of Eq. (3.67) by eliminating the candidates which do not render this expression constant, since the expression

$$\mathrm{lsb}\left(k_5^{(1)} \odot \left(\left(x_1^{(1)} \odot k_1^{(1)}\right) \oplus \left(x_3^{(1)} \boxplus k_3^{(1)}\right)\right)\right)$$

provides an unknown, but *constant* bit to the cryptanalyst. This process gives us 4 candidates for the key bits 57-88 within a complexity of less than 2^{20} 2-rounds IDEA evaluations.

Once this process is achieved, one can use the attacks described in §3.4.2 to derive key bits 0-15 and 32-47 in a 2^{33} effort and key bits 96-127 in another 2^{33} effort with 39 additional chosen-plaintext. Hence, this attacks recovers all key bits (the 31 remaining ones with help of an exhaustive search) in a computational complexity approximately equal to 2^{34} 2-rounds IDEA evaluations. Thus, this attack compares quite favorably with Demirci's square-like attack [84] which requires roughly the same order of chosen-plaintexts and a 2^{64} computational effort to recover the whole key.

If a decryption oracle is available, instead of an encryption one, we can still mount a *chosen-ciphertext* attack based on the same properties. It would work as follows: fix $y_1^{(2)}$ and $y_3^{(2)}$ to an arbitrary constant, and guess $k_1^{(1)}$, $k_3^{(1)}$, and $k_5^{(1)}$ (which represent 48 unknown key bits numbered 0-15, 32-47, and 64-79). Once these 48 bits recovered, after a 2^{48} process (provided 55 chosen plaintexts are available), one can recover 16 more bits (i.e. the still unknown bits of $k_5^{(2)}$ and $k_6^{(2)}$ in a second step, and 32 more (numbered 96-127 and corresponding to subkeys $k_1^{(2)}$ and $k_2^{(2)}$) in a third step; finally, the remaining ones can be found with help of an exhaustive search.

[31] A similar technique was used by Knudsen and Mathiassen [166] to speed up by a small constant a linear cryptanalysis of DES.

3.4.4 A New Chosen-Plaintext Attack Breaking 2.5, 3, and 3.5 Rounds

If we apply the Demirci-Biryukov relation to 2.5-rounds IDEA, then one get the following expression:

$$\mathrm{lsb}\left(x_2^{(1)} \oplus x_3^{(1)} \oplus c_2^{(3)} \oplus c_3^{(3)} \oplus \bigoplus_{i=1}^{3}\left(k_2^{(i)} \oplus k_3^{(i)}\right)\right) \oplus$$

$$\mathrm{lsb}\left(k_5^{(1)} \odot \left(\left(x_1^{(1)} \odot k_1^{(1)}\right) \oplus \left(x_3^{(1)} \boxplus k_3^{(1)}\right)\right)\right) \oplus \tag{3.70}$$

$$\mathrm{lsb}\left(k_5^{(2)} \odot \left(\left(c_1^{(3)} \odot \overline{k_1}^{(3)}\right) \oplus \left(c_2^{(3)} \boxminus k_2^{(3)}\right)\right)\right) = 0 \tag{3.71}$$

where $\overline{k_1}^{(3)}$ denotes the inverse of $k_1^{(3)}$ relatively to the group operation $\odot$. If we use the same trick than for 2 rounds and fix $x_1^{(1)}$ and $x_3^{(1)}$, an adversary can recover $k_5^{(2)}$, $\overline{k_1}^{(3)}$ and $k_2^{(3)}$ (key bits 57-72 and 89-120) in a 2^{48} effort if 55 chosen-plaintexts are available (the success probability is then larger than 0.99). Once achieved, one can recover 39 key bits ($k_1^{(1)}$, $k_1^{(3)}$ and the remaining unknown bits of $k_1^{(5)}$) numbered 0-15, 32-47 and 73-79 with the same distinguisher where Eq. (3.71) is fixed and known. For this, we need 46 additional known plaintexts. The remaining 41 key bits can be recovered with an exhaustive search within negligible computational complexity. Note that in this case, the Demirci-Biryukov relation applied on the decryption operation results in the same distinguisher. To the best of our knowledge, it is the fastest attack on 2.5-rounds IDEA not involving any weak-key assumption.

If a decryption oracle is available, instead of an encryption one, it is possible to mount a similar (chosen-ciphertext) attack: fix $c_1^{(2)}$ and $c_2^{(2)}$ to an arbitrary constant, and guess $k_1^{(1)}$, $k_3^{(1)}$, and $k_5^{(1)}$. In a second step, guess the remaining unknown bits of $k_1^{(3)}$, $k_2^{(3)}$, and $k_5^{(2)}$; one can finalize the attack using an exhaustive search.

We can extend to 3 rounds the attack previously described in a straightforward way: actually, if we fix $x_1^{(1)}$ and $x_3^{(1)}$ and guess $k_5^{(2)}$, $\overline{k_1}^{(3)}$, $k_2^{(3)}$, $k_5^{(3)}$ and $k_6^{(3)}$ (which represent key bits numbered 50-81 and 89-120), one can recover 64 key bits in a 2^{64} process if 71 chosen-plaintext are available. Then, once $k_5^{(2)}$, $\overline{k_1}^{(3)}$, $k_2^{(3)}$, $k_5^{(3)}$ and $k_6^{(3)}$ are known, one can apply the attack on 2.5 rounds to derive 49 more bits (numbered 0-15, 32-47, 73-79 and 127) with negligible complexity and the remaining 15 bits can finally be searched exhaustively.

The chosen-ciphertext version of this attack is clearly less effective, since one has to guess at least 96 unknown key bits (corresponding to subkeys $k_5^{(3)}$, $k_6^{(3)}$, $k_1^{(3)}$, $k_2^{(3)}$, $k_5^{(2)}$, $k_1^{(1)}$, $k_3^{(1)}$, and $k_5^{(1)}$; the unknown key bits are numbered 0-15, 32-47, 50-81,and 89-120).

For attacking 3.5 rounds, one uses a new time the distinguisher described above, one fixes $x_1^{(1)}$ and $x_3^{(1)}$ and one guesses furthermore all the keys of the last half-round; the subkeys under consideration are then $k_5^{(2)}$, $k_1^{(3)}$, $k_2^{(3)}$, $k_5^{(3)}$, $k_6^{(3)}$, $k_1^{(4)}$, $k_2^{(4)}$, $k_3^{(4)}$ and $k_4^{(4)}$ (i.e. all the key bits but the interval 18-49, representing 96 key bits). The computational effort is approximately equal to 2^{97} if 103 chosen-plaintexts are available.

The same attack can be adapted for a decryption oracle, however resulting in a higher complexity: if 119 chosen-ciphertext are available to an attacker (where $c_1^{(4)}$ and $c_3^{(4)}$ are fixed to an arbitrary constant), then one can recover 112 key bits numbered 0-111 (corresponding to subkeys $k_5^{(2)}$, $k_1^{(2)}$, $k_3^{(2)}$, $k_5^{(1)}$, $k_6^{(1)}$, $k_1^{(1)}$, $k_2^{(1)}$, $k_3^{(1)}$, and $k_4^{(1)}$).

3.4.5 Time-Memory Tradeoffs

We show now that it is possible under certain circumstances to trade between and time complexities in the attacks of Nakahara *et al.* [240].

Let us consider 2.5-rounds IDEA, and let us assume that we make use of 55 known plaintext-ciphertext pairs. For all possible values of $k_1^{(1)}$, $k_1^{(3)}$, and $k_5^{(1)}$ (i.e. key bits numbered 0-15, 32-47, and 64-79), we can compute a guess for the value of the following expression.

$$\underbrace{k_2^{(1)} \oplus k_2^{(2)} \oplus k_3^{(2)} \oplus k_2^{(3)} \oplus k_3^{(3)}}_{\text{constant}} \oplus \quad \text{lsb}\left(\delta^{(2)} \oplus \gamma^{(2)}\right) \tag{3.72}$$

The sub-sum depending only of the key bits is unknown but constant. Let us store all these guesses in a large hash table made of 2^{48} 55-bit words, As second step of the attack, one guesses the key bits $k_5^{(2)}$, $k_1^{(3)}$, and $k_2^{(3)}$ (i.e. bits numbered 57-72 and 89-120): for all these guesses, and for the 55 ciphertexts, we can compute (by partially decrypting the ciphertexts) the value of $\text{lsb}(\delta^{(2)} \oplus \gamma^{(2)})$ and checking whether this value (or its complement) is stored in the table or not. With high probability, the right subkey candidate will be determined by one of the few expected matches. This attack hence requires two times $55 \cdot 2^{48} \approx 2^{54}$ partial encryptions/decryptions, and 2^{48} memory cells, while the remaining 41 unknown bits can be recovered with an exhaustive search within negligible complexity.

The attack can be extended to more rounds in the following way. Using the same approach than for the 2.5-round case, one compute a hash table containing, for all possible values of $k_1^{(1)}$, $k_1^{(3)}$, and $k_5^{(1)}$, a guess for

$$\underbrace{k_2^{(1)} \oplus k_2^{(2)} \oplus k_3^{(2)} \oplus k_2^{(3)} \oplus k_3^{(3)}}_{\text{constant}} \oplus \quad \text{lsb}\left(\delta^{(2)} \oplus \gamma^{(2)}\right) \oplus \text{lsb}\left(\delta^{(3)} \oplus \gamma^{(3)}\right)$$

for 71 known plaintext-ciphertext pairs. In a second step, by guessing $k_5^{(2)}$, $k_1^{(3)}$, $k_2^{(3)}$, $k_5^{(3)}$, and $k_5^{(3)}$ (i.e. key bits numbered 50-81 and 89-120), one can

recover a total of 96 key bits, the remaining 32 ones with help of an exhaustive search, in an approximate overall computational complexity of 2^{70} operations.

Finally, this attack can be extended to 3.5 rounds if we guess the additional unknown key bits of $k_1^{(4)}$, $k_2^{(4)}$, $k_3^{(4)}$, and $k_4^{(4)}$ (i.e. bits numbered 0-17 and 121-127). One needs in this case 103 known plaintext-ciphertext pairs, 2^{48} 103-bit words of memory, and a computational complexity of about 2^{103} operations.

The same attack strategy on 4 rounds would imply guessing all the key bits, thus it is less efficient than an exhaustive key search.

3.4.6 Combination with other Attacks

Interestingly, we note that our attacks can be used in parallel with other attacks to gain more key bits. For instance, the attack on 3-rounds IDEA of Demirci *et al.* described in [85] is able to recover the values of $k_2^{(1)}$, $k_4^{(1)}$, $k_5^{(2)}$, and $k_5^{(3)}$ (which represents 41 key bits) in a 2^{42} effort (after a 2^{64} precomputation). Then, to derive 32 other key bits, the authors assume that a *decryption oracle* is available. If it is not the case, one can still relax this condition by applying the attack described in §3.4.4 and recover 41 additional key bits, namely those numbered 73-81 and 89-120, within negligible computational complexity. Similar considerations apply if *only a decryption oracle* is available.

Another interesting combination of known attacks and the ones described in this paper is the following: in [84], Demirci describes a square-like distinguisher which, with help of two sets of 2^{32} chosen-plaintexts, allows to recover $k_5^{(3)}$ in about 2^{49} operations. If, in a second step, we plug the obtained value of $k_5^{(3)}$ into the attack described in §3.4.4, we can derive 48 other key bits numbered 66-81, and 89-120 in a 2^{49} computational effort in a second step, and finally the remaining bits within negligible time. This defines an attack which derives all key bits within 2^{50} operations if 2^{33} chosen-plaintexts are available. This represents a complexity decrease by a factor of about 2^{32}. Unfortunately, the same strategy does only improve marginally the attack against 3.5 (or more rounds): one can replace the final exhaustive search of the remaining 80-bit keys by our more efficient attack.

3.4.7 A New Square-Like Distinguisher

As observed for the first time by Nakahara *et al.* [238] and later by Demirci [84], square-like distinguishers can be used with success to attack IDEA. We present now such a distinguisher which is somewhat simpler to use than the ones available in the literature.

Lemma 3.4.2 (Square-Like Distinguisher on 2.5-Round IDEA). *Let 2^{16} different inputs of 2.5-round IDEA be defined as follows: $x_1^{(1)}$, $x_2^{(1)}$, and $x_3^{(1)}$ are fixed to arbitrary constants, and $X_4^{(1)}$ takes all possible values. Then the XOR of the 2^{16} values of the equation*

$$\begin{aligned} & x_2^{(1)} \oplus x_3^{(1)} \oplus c_2^{(1)} \oplus c_3^{(1)} \oplus \\ & k_2^{(1)} \oplus k_3^{(1)} \oplus k_2^{(2)} \oplus k_3^{(2)} \oplus k_2^{(3)} \oplus k_3^{(3)} \oplus \\ & \mathrm{lsb}\left(\gamma^{(1)} \oplus \delta^{(1)}\right) \oplus \mathrm{lsb}\left(\gamma^{(2)} \oplus \delta^{(2)}\right) \end{aligned} \tag{3.73}$$

is equal to 0 with probability one.

We can then use this distinguisher to attack reduced-round versions of IDEA. To attack 3 rounds, encrypt 39 different structures of 2^{16} chosen plaintexts according to Lem. 3.4.2. Then, for all possible values of $k_5^{(3)}$ and $k_6^{(3)}$ (i.e. bits numbered 50-81), partially decrypt the ciphertext for the 39 structures using the same iterative strategy as in Alg. 3.12. This attack recovers 32 key bits, and with a few more chosen plaintexts, we can apply the attack on 2.5-rounds described in §3.4.4 to recover all the keys bits. In summary, this attack requires less than 2^{22} chosen-plaintexts and a computational complexity of approximately 2^{50} operations.

On 3.5 rounds, we can attack the round keys $k_5^{(3)}$, $k_1^{(4)}$, and $k_2^{(4)}$ (i.e. 48 key bits numbered 50-65 and 82-113) in a similar fashion. In this case, we need 55 structures of 2^{16} chosen plaintexts (i.e. less than 2^{22} chosen plaintexts as well), and a computational complexity of approximately $3 \cdot 2^{16} \cdot 2^{48} \approx 2^{66}$ operations.

Finally, we can attack 4 rounds using the same strategy by guessing further key bits, i.e. those of $k_5^{(4)}$ and of $k_6^{(4)}$, which represents 80 unknown bits in total. Hence, we need about 87 structures of 2^{16} chosen plaintexts, which is less than 2^{23} chosen plaintexts, and a computational cost of about $3 \cdot 2^{16} \cdot 2^{80} \approx 2^{98}$ operations.

3.4.8 Summary

In this section, we have used the same kind of properties derived by Demirci [84] and Nakahara et al. [239] to derive a sequence of simple, yet efficient attacks against reduced-round versions of IDEA; the attacks against 2 and 2.5 rounds are the best known ones not involving any weak-key assumption, to the best of our knowledge. Some of them, given the same order of computational complexity, reduce the amount of necessary chosen plaintexts, while other attacks, given a comparable amount of chosen texts, decrease favorably the computational complexity; additionally, some tradeoffs between time and memory are presented, which lead to far less complex attacks using only known plaintext-ciphertext pairs. Furthermore, we showed how to

use some of these attacks in combination with other known attacks, which allows sometimes to gain more key bits with a lesser complexity, or to avoid the use of both encryption and decryption oracles. The more important attacks against this block cipher are tabulated in Fig. 3.10 (which is an updated version of Fig. 2.12), as well as their respective complexities.

Rounds	Data	Time	Attack type	Ref.	Note
2	2^{10} CP	2^{42}	differential	[219]	
2	62 CP	2^{34}	*linear-like*	*§3.4.3*	
2	23 CP	2^{64}	square-like	[84]	
2.5	2^{10} CP	2^{106}	differential	[219]	Memory: 2^{96}
2.5	2^{10} CP	2^{32}	differential	[77]	For one key out of 2^{77}
2.5	2^{18} CP	2^{58}	square	[238]	
2.5	2^{32} CP	2^{59}	square	[238]	
2.5	2^{48} CP	2^{79}	square	[238]	
2.5	2 CP	2^{37}	square	[238]	Under 2^{16} rel. keys
2.5	55 CP	2^{81}	square-like	[84]	
2.5	101 CP	2^{48}	*linear-like*	*§3.4.4*	
2.5	97 KP	2^{90}	linear-like	[239]	
2.5	55 KP	2^{54}	*linear-like*	*§3.4.5*	*Memory:* 2^{48}
3	2^{29} CP	2^{44}	differential-linear	[45]	
3	71 CP	2^{71}	square-like	[84]	
3	71 CP	2^{64}	*linear-like*	*§3.4.4*	
3	2^{33} CP	2^{64}	collision	[85]	Memory: 2^{64}
3	2^{33} CP	2^{50}	*linear-like + [84]*	*§3.4.6*	
3	2^{22} CP	2^{50}	*square-like*	*§3.4.7*	
3	71 KP	2^{70}	*linear-like*	*§3.4.5*	*Memory:* 2^{48}
3.5	2^{56} CP	2^{67}	truncated diff.	[45]	
3.5	$2^{38.5}$ CP	2^{53}	impossible diff.	[27]	Memory: 2^{48}
3.5	2^{34} CP	2^{82}	square-like	[84]	
3.5	2^{24} CP	2^{73}	collision	[85]	
3.5	2^{22} CP	2^{66}	*square-like*	*§3.4.7*	
3.5	103 CP	2^{103}	square-like	[84]	
3.5	103 CP	2^{97}	*linear-like*	*§3.4.4*	
3.5	119 KP	2^{112}	linear-like	[239]	
3.5	103 KP	2^{97}	*linear-like*	*§3.4.5*	*Memory:* 2^{48}
4	2^{37} CP	2^{70}	impossible diff.	[27]	Memory: 2^{48}
4	2^{34} CP	2^{114}	square-like	[84]	
4	2^{24} CP	2^{89}	collision	[85]	Memory: 2^{64}
4	2^{23} CP	2^{98}	*square-like*	*§3.4.7*	
4	121 KP	2^{114}	linear-like	[239]	
4.5	2^{64} CP	2^{112}	impossible diff.	[27]	
4.5	2^{24} CP	2^{121}	collision	[85]	Memory: 2^{64}
5	2^{24} CP	2^{126}	collision	[85]	Memory: 2^{64}

Figure 3.10: Attacks against IDEA(updated)

Chapter 4

Design of Block Ciphers

This chapter is devoted to the design of secure and efficient block ciphers. We first review generalities about the task of designing a block cipher. Then, we address specifically the problem of designing fast *diffusive* components in §4.2. Finally, we present in §4.3 the description of a new family of block ciphers, named FOX, and we discuss thoroughly its characteristics both in terms of security and of performance.

4.1 General Considerations

In his seminal paper [295], Shannon noticed that the design of a block cipher should include two fundamental principles: *confusion* and *diffusion*. The principle of confusion was interpreted by Lai [179] in his PhD thesis as:

> "*The dependence of the key on the plaintext and ciphertext should be so complex that it is useless for cryptanalysis.*"

while the principle of diffusion is defined as:

> "*For virtually every key, the encryption function should be such that there is no statistical dependence between simple structures in the plaintext and simple structures in the ciphertext and that there is no simple relation between different encryption functions.*"

In this part, we discuss several strategies and issues about the design of modern, fast and secure block ciphers, namely how to guarantee these two properties into a block cipher. We treat successively common skeletons of block ciphers in §4.1.1, non-linear components in §4.1.2, diffusive components in §4.1.3, and key-schedule algorithms in §4.1.4. Finally, we briefly describe in §4.1.6 typical target platforms and the constraints they induce during the design of a block cipher.

4.1.1 Skeletons

Most of the known algorithms share the same kind of high-level structure, or skeleton. We discuss now the most common skeletons encountered in the litterature and used in the practice. Finally, we discuss their respective advantages and drawbacks.

Feistel Schemes

The most widely used block cipher scheme is clearly the *Feistel scheme* [102] (see Def. 2.2.1 and Fig. 2.3 for a formal definition thereof and for an illustration, respectively). This popularity comes certainly from the fact that it is fairly easy to use any random round function f in order to make a permutation. In addition, encryption and decryption operations hardly need separate implementations. Another reason why the Feistel scheme is so popular is that it has been widely studied in a theoretical point of view. For instance, Luby and Rackoff proved in [191] that a Feistel network can generate a pseudorandom permutation if the underlying round functions are pseudorandom and if we use at least three rounds (see also the description of the Luby-Rackoff model of security in §2.4.2, page 53, which contains an extensive list of bibliographical references to the subject as well). A large number of block ciphers are Feistel ciphers: DES, the FEAL family, LOKI, Khufru, Khafre, Blowfish, Misty1, E2 and DFC, to name a few. Twofish uses a slightly modified Feistel structure, involving data-independent rotations in the branches, while Camellia is a Feistel cipher with an additional mixing layer every 6 rounds.

Generalized Feistel Schemes

In a common (balanced) Feistel scheme, half the bits operate on the other half. It is natural to consider the *unbalanced* case, where the f-function takes as input a different number of bits than its output. This approach has been proposed by Blaze and Schneier, in the design of MacGuffin, and then generalized in [285]. Fig. 4.1 illustrates the example of an unbalanced Feistel scheme with 4 branches. Three quarters of the data width form the f-function input in each round and the last quarter is combined with the f-function output. The high-level structures of CAST and Mars are a kind of unbalanced Feistel schemes.

It is furthermore possible to generalize the Feistel construction: for instance, we may consider a scheme where the f-function is not the same one (i.e. it is not only modulated by the subkeys) in each round. Since the internal properties of such a heterogenous scheme change from round to round, it may be much more complicated to find any kind of cryptanalytic property that propagates well through all rounds. However, implementing such a kind of schemes is costly and analyzing them is much more difficult.

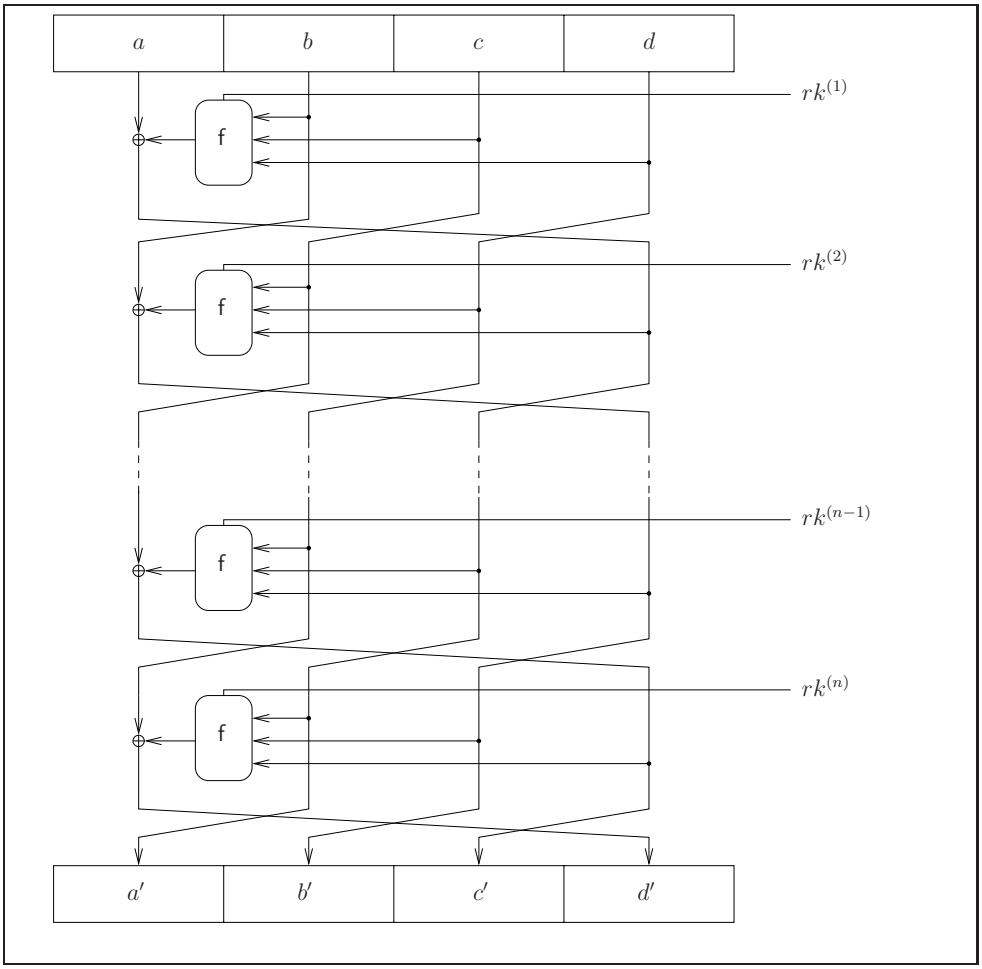

Figure 4.1: Generalized unbalanced Feistel network with 4 branches

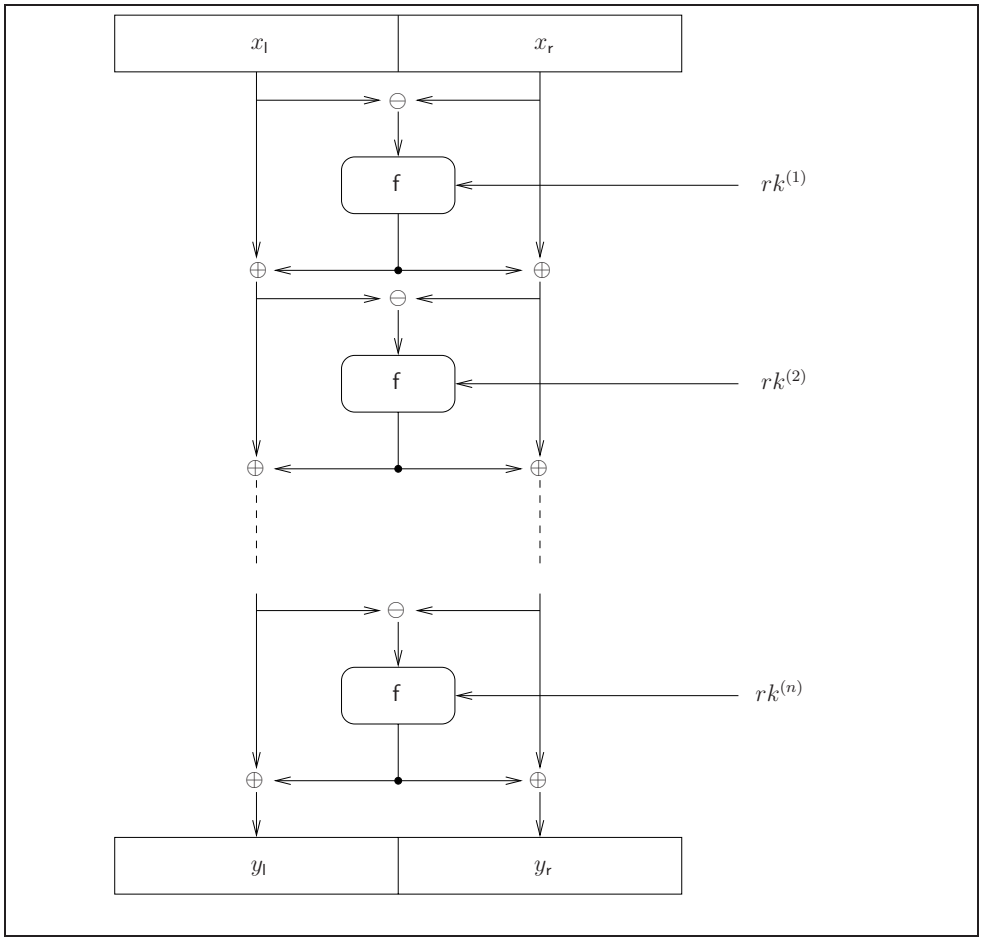

Figure 4.2: Lai-Massey Scheme

Lai-Massey Schemes

A far less-used skeleton is the so-called *Lai-Massey scheme*, which was introduced with PES and its successor IDEA. It is based on a group law and, like the Feistel scheme, it allows to build a permutation from any function (see Fig. 4.2). The two halves of the input are combined with help of the inverse group law (denoted $\ominus$ in Fig. 4.2), fed into the round function, and the corresponding output is combined with the two branches with help of the group law (denoted $\oplus$ in Fig. 4.2). We note however that without any modification, the Lai-Massey scheme is insecure, since there exists a differential characteristic which holds with probability one (see Eq. (4.5), page 197). This annoying property can however be avoided by plugging an orthormorphism, as we will discuss it in §4.3.4. Besides the ciphers PES, IDEA, and the MESH family, FOX, which will be described in §4.3, is also built on top of a (slightly modified) Lai-Massey scheme.

Substitution-Permutation Networks

The so-called *substitution-permutation network (SPN)* (see Fig. 4.3) applies the principles of confusion and diffusion in a rather straightforward way by alternating layers of substitution and layers of permutation. Although one can view Feistel networks and related schemes as SPNs, we won't refer to them with this notation, because each layer does not apply on the whole block size. There is usually two methods for combining the key material: either subkeys are XORed before the entry in a substitution layer, or the subkeys determine (key-dependent) S-boxes.

While SPNs' security has been extensively studied (from an information-theoretic point of view [300], towards differential and linear cryptanalysis [130–132], for instance), there is rather few available theoretical work on the *structural* security properties of SPNs. Recently, Biham published [22] a dedicated attack against an ASASA structure, i.e. against a 5-layers SPN where the permutations are affine transformations. Later, Biryukov and Shamir have presented [36] the first structural attack against a SASAS structure, with a surprisingly low complexity. Another work in the same vein is Moriai and Vaudenay's discussion of SPNs at the light of the decorrelation theory [231]. Several modern ciphers are SPNs. For instance, one can mention Shark, Square, Rijndael, or Serpent.

Other Constructions

Some modern ciphers are not built on top of one of the typical structures described below. For instance, RC5 has a kind of Feistel structure, but where the f-function operations, the key mixing and the combination of both branches are mixed. This structure has been extended to a kind of generalized Feistel network with four branches in RC6.

4.1.2 Non-Linear Components

The strength of most of block ciphers (more specifically their resistance against linear and differential cryptanalysis) is inevitably tied to the strength of their S-boxes, which is usually their sole non-linear component. In this part, we discuss the strength criteria of non-linear constructions together with strong S-boxes construction methods.

An n-bit to m-bit S-box defines simply a *substitution*, i.e. to each n-bit input is mapped a corresponding m-bit output value (which has not necessarily the same length than the input). S-boxes are responsible for bringing confusion in the data processing. This means that they should hide any mathematical relationship between the plaintext, the ciphertext and the key.

An S-box can be described by a lookup table of 2^n elements of m bits. Hence, for implementation reasons, it is desirable to keep the table as small

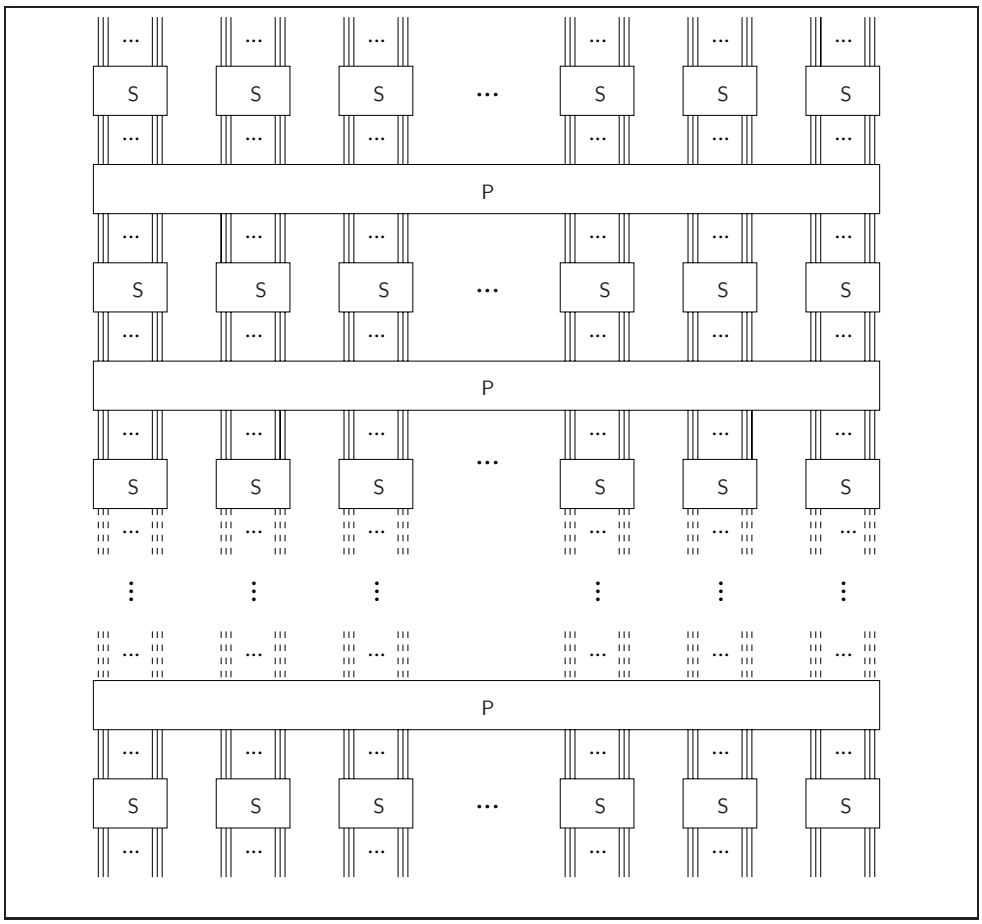

Figure 4.3: Substitution-Permutation Network (SPN)

as possible or to give a simple algorithmic description. For instance, in dedicated hardware implementations, S-boxes can be implemented as a straightforward lookup table stored in ROM. Since the area of such table grows exponentially with the length of the input, small boxes are desirable from this point of view. In software, an S-box is usually implemented as an array of constants that is indexed by the n-bit input. In general, S-boxes have a small portability; as a matter of fact, there is no benefit in using processors with a register length larger than the S-box input length. In certain cases, if the underlying processor has a register size which is several times as large as a S-box input, it is however possible to handle several S-boxes at the same time. Another way to implement S-boxes in an efficient way, given that the input size is not too large, is the *bitslice trick* [21] (used on our fast implementation of DES, see §3.2.5): one expresses each S-box's output bit as a Boolean expression of the input bits. This approach has been applied in the design of Serpent. However, finding optimal Boolean expressions of a S-box is not a trivial task [178] in the general case, and the results can be quite different on various architectures, provided the number and the size of logical operations at disposal, or the instruction cache size [216].

Non-Linearity Criteria

Obviously, there exist weak and strong S-boxes: to give a trivial illustration thereof, one can simply take the trivial substitution which consists in replacing each input by itself. Thus it seems natural to define strength criteria of S-boxes. We must however note that a weakness (or strength) measure is always related to a given attack, thus, new attacks often imply new strength criteria.

Perfect Nonlinear Functions The importance of measuring the non-linearity (see Def. 4.1.1) of cryptographic functions was first noticed by Pieprzyk and Finkelstein [266].

Definition 4.1.1 (Non-linearity of a Boolean function). *Let* f *and* g *be functions from* $\mathrm{GF}(2)^n$ *to* $\mathrm{GF}(2)$*. Let*

$$\delta_{\mathsf{H}}(\mathsf{f},\mathsf{g}) = \sum_{x} 1_{\mathsf{f}(x)\neq\mathsf{g}(x)}$$

be the Hamming distance between f *and* g*. Let* $\varphi_1,\ldots,\varphi_{2^n+1},\ldots,\varphi_{2^{n+1}}$ *be all affine functions* $\varphi_i : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$*. Then*

$$n_{\mathsf{f}} = \min_{i=1,\ldots,2^{n+1}} \delta_{\mathsf{H}}(\mathsf{f},\varphi_i)$$

is called the non-linearity *of* f*.*

In their seminal paper [220] initiating the study of Boolean functions in a cryptographic context, Meier and Staffelbach consider the distance to linear and affine functions as a non-linearity criterion. They define *perfect non-linear functions* (see Def. 4.1.2); furthermore, it turns out that perfect non-linear functions correspond to certain functions known in combinatorics, the so-called *bent functions*.

Definition 4.1.2 (Perfect non-linear Boolean functions). *A Boolean function* $\mathsf{f} : \mathrm{GF}(2^n) \rightarrow \mathrm{GF}(2)$ *is called* perfect non-linear *if for every non-zero vector* $a \in \mathrm{GF}(2)^n$ *the values* $\mathsf{f}(x \oplus a) = \mathsf{f}(x)$ *are equal for exactly half of the arguments* $x \in \mathrm{GF}(2)^n$.

An equivalent way to define perfect non-linear Boolean functions consists in saying that they have a non-linearity degree[1] equal to $2^{n-1} + 2^{\frac{n}{2}-1}$.

Strict Avalanche Criterion Based on earlier ideas of Feistel [102] and of Kam and Davida [154], Webster and Tavares have proposed the *strict avalanche criterion* (SAC) in [326] (see Def. 4.1.3). This concept is based on the concept of *completeness* of a cryptographic function and on the concept of *avalanche effect*. A cryptographic function is said to be *complete* if each ciphertext bit depends on all the plaintext bits, while a function is said to exhibit the avalanche effect if an average of one half of the output bits change whenever a single input bit is complemented.

Definition 4.1.3 (Strict avalanche criterion). *Let* x *and* x' *be two* n*-bit binary vectors such that* x *and* x' *differ only in bit* $i, 1 \leq i \leq n$. *Let* $v = y \oplus y'$, *where* $y = \mathsf{f}(x), y' = \mathsf{f}(x')$ *and* $\mathsf{f}(.)$ *is the Boolean function under consideration.* f *is said to meet the* strict avalanche criterion *if the probability that each bit of* v *is equal to 1 is equal to* $\frac{1}{2}$ *over the set of all vectors* x *and* x' *for all* i.

This notion has then been generalized and linked to the Walsh-transform by Forré [107] and functions satisfying higher order SAC criteria have been studied, for instance, by Cusick [74].

Links between Nonlinearity Criteria The links between non-linearity criteria have been the subject of several efforts after the discovery of the linear and differential cryptanalysis (see §2.3 for a description of these attacks). In order to study linear and differential properties of functions, useful measures are the linear (see Def. 2.3.9) and differential probabilities (see Def. 2.3.2). We summarize in the following list known results on Boolean functions and links between non-linearity criteria. We refer to f as a function mapping $\{0,1\}^p$ to $\{0,1\}^q$.

[1] Note that perfect non-linear Boolean functions exist only for n even.

- $\mathrm{DP}^{\mathsf{f}}_{\max} \geq 2^{-q}$. The equality holds for perfect non-linear functions.
- $\mathrm{DP}^{\mathsf{f}}_{\max} \geq 2^{1-p}$. The equality holds for *almost perfect non-linear functions* [252].
- $\mathrm{LP}^{\mathsf{f}}_{\max} \geq 2^{-p}$. The equality holds for *bent functions* [280].
- $\mathrm{LP}^{\mathsf{f}}_{\max} \geq 2^{1-p}\left(1 + \frac{(2^{q-p}-1)(2^{p-1}-1)}{2^q-1}\right)$. The equality holds for *almost-bent functions* [58].
- Bent functions and perfect non-linear functions are equivalent [220] and exist if and only if $p \geq 2q$ and p is even [249].
- Almost-bent functions exist if and only if $p = q$ and p is odd and are almost-perfect non-linear functions (the reciprocal is not true) [58].
- Almost-perfect non-linearity is only possible if $(p, q) = (2, 1)$ or $q \geq p$.

S-Boxes Construction Methods

Since S-boxes play such a key role in the security of block ciphers, their selection method is a very important task during the design of a block cipher. It is possible to identify three different strategies: random choice, random choice followed by filtering, and algebraic constructions. The two first strategies are heuristic methods, while the last one is clearly more mathematical.

Random Choice A possible strategy in selecting S-boxes is to choose them completely at random. While this approach can be insecure for small S-boxes (for instance, Biham and Shamir noticed [33] that replacing S-boxes of DES by random S-boxes yielded ciphers that were far weaker towards differential cryptanalysis than the original algorithm), theoretical works by O'Connor [253–255] and later by Youssef and Tavares [333–335] have shown that large random S-boxes are in average very resistant to linear and differential cryptanalysis. A way to choose random S-boxes is to make them key-dependent. As example, this approach has been used in the algorithms IDEA, RC5, RC6, Blowfish or Twofish.

Random Choice Followed by Filtering Another way to select good S-boxes is to generate random ones and to check if they have the desired properties until a good one is found. The following list enumerates common test criteria:

- Upper bound on every linear characteristic's probability (according to Def. 2.3.11).
- Upper bound on every differential characteristic's probability (according to Def. 2.3.3).

- Lower bound on the non-linearity of the S-box (see Def. 4.1.1).
- Upper bound on the single-bit correlation

$$\left| \Pr_x [\mathsf{S}(x)_j = x_i] - \frac{1}{2} \right|$$

  where x_i denotes the i-th bit of x.
- SAC fullfiled (see Def. 4.1.3)
- Upper bound on the number of gates in a hardware implementation

Although it can be a very heavy and computation-intensive process, this approach has been used in a variety of block ciphers, including DES, Serpent, Mars, Crypton, or CAST, for instance.

Algebraic Methods In order to bring non-linearity in a block cipher, one can use algebraic methods, like mixing non-isomorphic operations (XOR and addition modulo 2^{32} for 32-bit vectors, for instance), or by using algebraic operations known to offer good non-linearity properties. Thus, one can see these S-boxes as data-dependent. We give now a (naturally non-exhaustive) list of common algebraic constructions:

- Mixing of addition in $\mathrm{GF}(2)^n$ and in $\mathbb{Z}_n$ used in the design of FEAL, for instance.
- Power function in $\mathrm{GF}(2^n)$ as in LOKI, or Shark.
- Combination of an inverse function $x \mapsto \frac{1}{x}$ in $\mathrm{GF}(2^n)$ and an affine transformation over some other incompatible algebraic structure as in Square, Rijndael, Camellia, and many others.
- Combination of a power function $x \mapsto x^e$ in $\mathrm{GF}(2^n)$ and an affine transformation over $\mathbb{Z}_n$ (E2, Misty1). Note that this generalizes the previous case (where $e = 2^n - 2$).

It remains however quite unclear at this time whether a pure algebraic S-boxes construction method is really as secure as expected (see the discussion in §4.3.1).

4.1.3 Diffusive Components

The purpose of a diffusive construction is to provide an avalanche effect, both in the context of differential and linear approximations. In the linear context, this means that there should be no correlations between linear combinations of a small set of inputs and linear combinations of a small set of outputs. In the differential context, small input changes should cause

large output changes, and conversely, to produce a small a output change, a large input change should be necessary. We discuss in this chapter several theoretical concepts which are used to build diffusive constructions in the practice.

Multipermutations

The concept of *multipermutation*[2] was first introduced by Schnorr and Vaudenay in [287]. This concept is simply a formalization of perfect diffusion.

Definition 4.1.4 (Multipermutation). *An* (r, n)-multipermutation *over an alphabet* $\mathcal{A}$ *is a function* f *from* $\mathcal{A}^r$ *to* $\mathcal{A}^n$ *such that two different* $(r+n)$*-tuples of the form* $(x, \mathsf{f}(x))$ *cannot collide in any* r *positions.*

An equivalent definition says that the set of all (r, n)-tuples of the form $(x, \mathsf{f}(x))$ is an error correcting code with minimal distance $n + 1$, which is the maximal possible. Some multipermutations are equivalent to Latin squares. For instance, a $(2, 1)$-multipermutation is nothing else as a *common Latin square*, i.e. a $k \times k$ matrix having elements over a finite set $\mathcal{A}$ of $k = \#\mathcal{A}$ elements such that all elements are represented in each column and in each row. A $(2, n)$-multipermutation is equivalent to a set of n two-wise *orthogonal Latin squares*[3].

The design of multipermutations over a finite alphabet is a very difficult problem, as the design of two-wise orthogonal Latin squares is a well-known difficult one. If we allow f to be a linear function[4], then one can use a MDS code, which is the topic of a next section.

Branch Number Concept

A way to characterize the avalanche effect of a diffusive construction is its *branch number* [76], which is a concept proposed by Daemen in his PhD thesis. In the following, we denote by $\delta_{\mathsf{W}}(x)$ the Hamming weight of a vector having components in a finite field $\mathrm{GF}(2^n)$, i.e. the number of components different from zero.

Definition 4.1.5 (Branch number). *Let* $\theta : \mathrm{GF}(2^n)^m \to \mathrm{GF}(2^n)^m$ *be a mapping. Then*

$$\mathfrak{B}(\theta) = \min_{x \neq 0} \{\delta_{\mathsf{W}}(x) + \delta_{\mathsf{W}}(\theta(x))\}$$

is called the branch number $\mathfrak{B}(\theta)$ *of* θ*.*

[2]The term "bipermutation" was suggested by Massey to mean a $(2, 2)$-multipermutation, which eventually led to the one of "multipermutation".

[3]Two Latin square $\mathbf{A}$ and $\mathbf{B}$ are orthogonal if the mapping $(i, j) \mapsto (\mathbf{A}_{i,j}, \mathbf{B}_{i,j})$ gets all possible couples.

[4]An example of *non-linear* multipermutation is the one used in CS Cipher.

The branch number of a diffusive construction gives a measure for the worst case diffusion: it thus is a lower bound for the number of active S-boxes in two consecutive rounds of a linear or a differential characteristic. Since a cryptanalysis will always exploit the worst case, this is a good measure for the diffusion property.

We note that the Hamming weight $\delta_{\mathsf{W}}(x) \leq m$ for a vector x and for any choice of θ. If $\delta_{\mathsf{W}}(x) = 1$, this implies that $\mathfrak{B}(\theta) \leq m+1$. This leads to the definition of an optimal mapping

Definition 4.1.6 (Optimal mapping). *A mapping*

$$\theta : \mathrm{GF}\,(2^n)^m \rightarrow \mathrm{GF}\,(2^n)^m$$

is called optimal *if* $\mathfrak{B}(\theta) = m + 1$.

As a matter of fact, this optimal case corresponds to the multipermutation definition. A popular implementation of optimal linear invertible mappings are the MDS codes, which will be discussed afterwards. Several modern block ciphers use the concept of branch number in their design: this is the case of Shark, Square or Rijndael, for instance.

MDS Codes

MDS codes have rapidly become a very popular tool in block cipher design; Vaudenay [311] has first proposed them as an implementation of a multipermutation. Indeed, they are elegant ways to implement multipermutations and they are optimal invertible linear mappings, in the sense of Daemen's branch concept number. We first give the formal definition of a linear code.

Definition 4.1.7 (Linear code). *A* $[n, k, d]$-linear code $\mathcal{C}$ *of length* n*, dimension* k *and minimal distance* d *is a* k*-dimensional subspace of the vector space of* n*-tuples over a given field, where the distance* d *between two codewords is defined to be the number of elements in which they differ.*

A well-known result of coding theory is the Singleton bound, which is an upper bound on the minimal distance of any linear code.

Theorem 4.1.1 (Singleton's bound). *For any* $[n, k, d]$*-linear code, the minimal distance* d *is upper bounded by*

$$d \leq n - k + 1$$

Codes with a minimal distance $d = n - k + 1$ are called *maximal distance separable (MDS)* codes. A well-known example of MDS-codes are the Reed-Solomon codes. A linear code $\mathcal{C}$ can be represented by a *generation matrix* $\mathbf{G}_{k \times n}$. This matrix has dimension $k \times n$, and is always of full rank. $\mathcal{C}$ is formed by the subspace of dimension k that is spanned by the rows of G

$$\mathcal{C} = \left\{ \mathbf{G} \cdot x \mid x \in \mathrm{GF}\,(2^m)^k \right\}$$

As the generation matrix of a code is not unique, one can always write

$$\mathbf{G}_{\mathrm{e}} = \mathbf{T} \cdot \mathbf{G} = \left[\mathbf{I}_{k\times k}\mathbf{B}_{k\times(n-k)}\right]$$

where $\mathbf{I}_{k\times k}$ is the $k \times k$ identity matrix and $\mathbf{T}$ is a full-rank matrix; this form is called the *echelon-form matrix* of $\mathbf{G}$. The following theorem [275] (which is nothing but a rewrite of Vaudenay results in the branch number terminology) ensures that a MDS-code has optimal diffusive properties

Theorem 4.1.2. *Let $\mathcal{C}$ be a $[2n, n, n+1]$-code over* GF (2^n)*. Let $\mathbf{G}_{\mathrm{e}}$ be the generator matrix in echelon form $\mathbf{G}_{\mathrm{e}} = \left[\mathbf{I}_{k\times k}\mathbf{B}_{k\times(n-k)}\right]$. Then $\mathcal{C}$ defines an optimal linear invertible mapping θ:*

$$\theta : \left\{ \begin{array}{rcl} \mathrm{GF}(2^m)^n & \rightarrow & \mathrm{GF}(2^m)^n \\ \mathbf{x} & \mapsto & \mathbf{B} \times \mathbf{x} \end{array} \right.$$

Finally, we give an alternate and useful criterion (see [197, page 321] for a proof thereof) on matrices which allows to test for their optimality in terms of diffusion.

Theorem 4.1.3. *A matrix is an MDS matrix if and only if every sub-matrix is non-singular.*

MDS codes have become an increasing popularity: Shark, Rijndael, Twofish, Square are examples of ciphers using them as diffusive element. In §4.2, we will present a formal approach in designing efficient MDS matrices, as well as new constructions.

Pseudo-Hadamard Transform

The *Pseudo-Hadamard Transform* (see Fig. 4.4) was proposed by Massey in the cipher SAFER K-64. This is an unorthodox linear transform that allows the cipher rapidly to achieve the desired diffusion. The box denoted 2-PHT is defined as follows, where x and x' denote the input bytes and y and y' the output ones.

$$\text{2-PHT} : \left\{ \begin{array}{rcll} y & = & 2x + x' & (\mathrm{mod}\ 256) \\ y' & = & x + x' & (\mathrm{mod}\ 256) \end{array} \right.$$

The arithmetic is usual byte arithmetic, i.e. performed modulo 256. Between levels of the linear layer, a *decimation-by-2 permutation*, which is familiar from the FFT and the ordinary discrete Hadamard transform, is applied.

4.1.4 Key-Schedule Algorithms

The *key schedule* algorithm is responsible for deriving subkey material out of the key to the data randomizing part of the block cipher. We discuss in this part various issues around the design of key schedule algorithms.

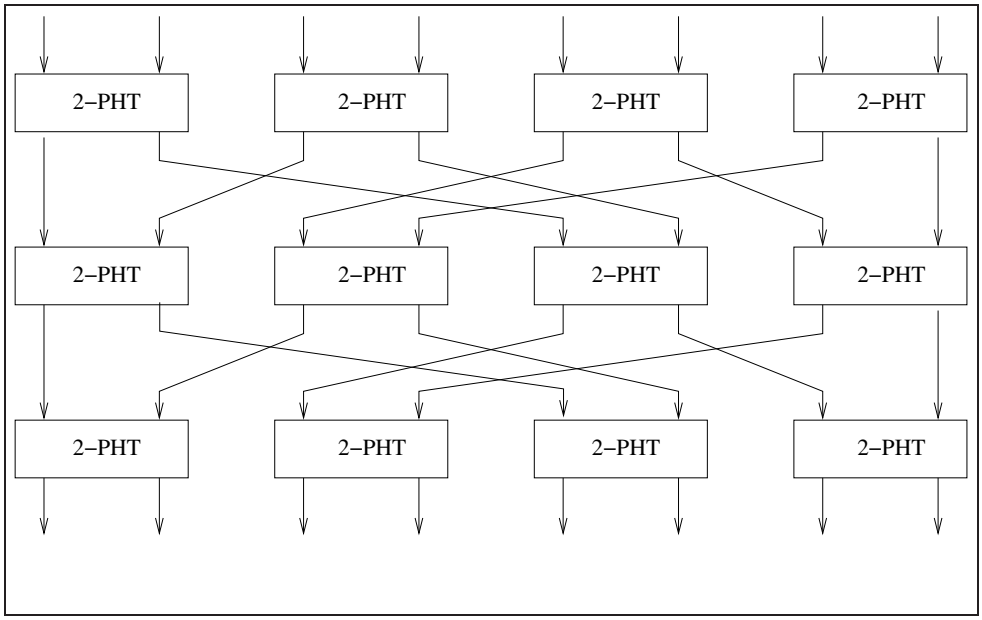

Figure 4.4: Pseudo-Hadamard Transform

Although block cipher design theory seems now to have several well-established strategies (which we discuss in §4.1.5) at disposal, the descriptions of the key schedule algorithms found in the academic literature are astonishingly full of empiric receipts and it is not common to get well-founded scientific arguments on a specific key scheduling algorithm. The only usually needed functional property of a key schedule is that it is possible to compute it in both directions, i.e. one can produce a sequence $rk^{(0)}, rk^{(1)}, \ldots, rk^{(n)}$ of subkeys from the key k as well as the reverse sequence $rk^{(n)}, rk^{(n-1)}, \ldots, rk^{(0)}$. This property is important for the efficiency of the decryption process.

The key schedule can play an important role for eliminating symmetry in a block cipher. For instance, it is quite common that a round transformation treats all bytes of the "current state" in a very similar manner. This symmetry can be removed by having constants in the key schedule. Furthermore, the round transformation is often the same for all rounds. This equality can be removed by having round-dependent round constants in the key-schedule.

Two major tendencies in the key-schedule design can be distinguished. A goal of older key schedule algorithms is to spread the key entropy in the subkeys in a "smooth" manner. A supplementary goal of more modern key schedule algorithms is to bring some one-wayness, in the sense that getting a subkey does not compromise most of the other subkeys. We discuss now these two strategies.

Diffusive Key Schedules

Older block ciphers have often a quite simple key schedule algorithm. One very illustrative example of this fact is DES. The key schedule algorithm is responsible to produce 16 subkeys of 48 bits from the 56-bit key. The key is diffused in the subkeys by a subtle (but completely linear) key bits selection process which uses rotations and a "compressive permutation". Thus, every key bit is used in approximately 14 of the 16 subkeys, although not all bits are used exactly the same number of times.

Other nice examples of simple key schedules are the one of LOKI, which uses a Feistel scheme with 12-bit and 13-bit rotations as f-function and the one of IDEA, which employs rotations as well.

Unfortunately, using such simple, linear algorithms to derive subkey bits from the key can expose the cipher to annoying, although not dramatic, properties, like weak and semi-weak keys, complementation properties, or even related-keys attacks (see §2.3 for more details about these attacks). For instance, all the attacks against IDEA described in §3.4 do not work it the key-schedule algorithm of IDEA is replaced by a stronger version.

Modern Key Schedules

In order to eliminate these potential annoying properties, block ciphers designers are using non-linear key schedules. An early example is FEAL, which uses a kind of Feistel scheme combined with the (slightly modified) f-function of its data randomizing part. However, it is worth to note that designers of FEAL assign to the key-schedule as sole task "*the generation of* different *extended keys from the 64-bit secret key*", without any further motivation.

Many attack models assume that the subkeys are statistically independent and uniformly distributed. In order to allow to prove some security level, one can see the key schedule algorithm as a pseudo-random generator which uses the key as seed value. For instance, this approach was used in the design of DFC in a way which was not successful in the original version (as demonstrated by Coppersmith which exhibited a large class of weak keys); this was later fixed in the design of DFCv2.

An interesting observation is that many attacks on iterated ciphers first recover (part of) a round key. This knowledge is subsequently used to recover other round keys. To make these attacks less efficient, one can generate the round keys by processing the key with a preimage resistant function. This idea has been used in the key schedule algorithm of Blowfish and CAST, or in Shark. It is worth to note that this is not a general rule: Square specifies its key schedule algorithm as being an "*iterative invertible transformation*".

In order to protect block ciphers against exhaustive search attacks, it has been proposed to render the key schedule process computationally intensive. In the real world, a key schedule algorithm execution is often bound with

other time consuming operations, like public-key operations, or network protocol executions. Thus, as long as the key schedule is precomputed in memory and the data randomizing part of the block cipher is fast, this speed loss is not a real disadvantage in terms of performance. However, this argument holds only if enough memory is available, which is typically not the case in a smartcard environment, for instance. This concept has been applied in Blowfish in an intentional way, but most of modern block ciphers have a key schedule algorithm which is more time consuming than the data randomizing part.

When choosing the non-linear parts of the key schedule algorithm, block cipher designers reuse very often the same (or very slightly) modified components as in the data randomizing part. This is obviously a real advantage when implementing the cipher in hardware.

4.1.5 Block Ciphers Design Paradigms

In this part, we discuss several paradigms used in the past to design block ciphers, as well as their (dis-) advantages.

Empirical Security

A first possibility for a block cipher designer is to choose the *empirical security* strategy. The principle is very simple: one designs the algorithm such that it fullfils a set of practical design criteria (speed, platform, key size, implementation complexity) but without any mathematical justification towards the real security level of the cipher. So the cipher is considered to be secure by its designer(s) (and hopefully by the users) as long as nobody is able to break it faster than an exhaustive key search.

An example of (early) empirical security design is FEAL. In the first proposal [296], in 1987, the designers propose a block cipher built on a Feistel scheme dedicated to be faster than DES and to be an alternative thereof. This cipher, called FEAL-4, has 4 rounds, a block and a key size of 64 bits. We quote here, for historical purposes, the only statement found in the paper regarding the cipher's security:

> *"*FEAL *working with no parity in a key block is safe from all-key attack because it is controlled by a 64-bit key, which is more secure than the 56-bit* DES *key. Regarding ciphertext randomization,* FEAL *is considered safe because the randomization indices are closer to the theoretical values than those of* DES*."*

As FEAL-4 was broken by den Boer [86] a few months later, the number of rounds was increased to 8 and called FEAL − 8. In 1990, Biham and Shamir proposed the differential cryptanalysis [31] and showed that it is possible to break FEAL with a differential cryptanalysis up to 31 rounds. In the

same time, other researchers have proposed attacks against several variants of FEAL: Chassé and Gilbert [113] against 8 rounds in 1990, Tardy-Corfdir and Gilbert [308] against 4 and 6 rounds in 1991, Matsui and Yamagishi [206] against 4, 6, and 8 rounds in 1992. In response, Miyaguchi [227] proposed the so-called FEAL−n cipher family, where n is the number of rounds, together with FEALNX, a 128-bit key version of the cipher. The current suggested number of rounds is 32. Other studies regarding the resistance of the FEAL cipher family towards modern cryptanalysis have been presented. For instance, Moriai, Aoki and Ohta prove in [229] that FEAL-32 is secure against linear cryptanalysis.

We think that empirical security arguments can no longer be considered as serious for modern block ciphers. Although FEAL with 32 rounds is secure against modern forms of cryptanalysis, no real credibility is given by its design against future cryptanalysis. Since the first publication regarding FEAL, several new cryptanalysis methods and more systematic ways of building secure block ciphers have been proposed.

Heuristic Security

By combining elements having known good properties, like for instance multipermutations for diffusion and highly non-linear functions for confusion purposes, one can build block ciphers which have a very good security level. For instance, CS Cipher is such a cipher. By heuristic security, one means that some properties like resistance against various attacks can be proven for an algorithm by using heuristic assumptions, i.e. assumptions that are very likely to hold. In the case of CS Cipher, Vaudenay [315] proves, that under a certain assumption, this algorithm is secure[5] against linear, differential, and truncated differentials cryptanalysis. The involved assumption is that all round keys are uniformly distributed and independent; this is quite intuitive, given that the key schedule is not badly flawed. Furthermore, a certain confidence towards future cryptanalysis methods is given by the underlying cryptographic primitives.

Provable Security

As outlined in §2.4.3, it is very difficult to really prove security in a general way for a symmetric block cipher; many older constructions have empiric or heuristic security foundations, which bring more or less confidence. However, recent works have proposed interesting theoretical constructions in order to "prove" certain aspects of security in a block cipher. We discuss them in this section.

[5] More exactly, the proofs bound the probability of the best differential and linear characteristics.

Recursive Schemes Several theoretical results regarding the resistance of a block cipher against linear and differential cryptanalysis have been presented. The following theorem, proven[6] by Nyberg and Knudsen [252] for which concerns the differential cryptanalysis and by Nyberg [250] in the case of a linear cryptanalysis, gives an upper bound on $\mathrm{LP}^{\Psi}_{\max}$ and $\mathrm{DP}^{\Psi}_{\max}$ in the context of a Feistel scheme.

Theorem 4.1.4. *Assume that each round function* f_i *in a Feistel scheme* Ψ *is bijective and that*

$$\begin{cases} \mathrm{LP}^{\mathsf{f}_i}_{\max} & \leq & p \\ \mathrm{DP}^{\mathsf{f}_i}_{\max} & \leq & p \end{cases}$$

If the entire function Ψ *has at least four rounds, then*

$$\begin{cases} \mathrm{LP}^{\Psi}_{\max} & \leq & 2p^2 \\ \mathrm{DP}^{\Psi}_{\max} & \leq & 2p^2 \end{cases}$$

An example of cipher using this theorem is Misty1. In the paper describing it, Matsui shows how to "amplify" the results stated by this theorem with help of a recursive Feistel scheme, i.e. a Feistel scheme whose round function is a Feistel scheme, etc. Hence, by using a three level recursion, Matsui shows that the upper bound is p^4 instead of p^2.

Another example of a modern and recursive design is DEAL. In order to build a 128-bit block cipher, Knudsen simply takes a well-known and well studied 64-bit algorithm, DES, and uses it as a round function in a Feistel scheme. This constrction has a level of recursion equal to two. The main advantages of this approach are a good backward compatibility with existing hardware and software implementations and a good confidence in its overall security level, while serious drawbacks are of course the resulting speed performances.

Decorrelation As described in the previous sections, the problem of security analysis of a block cipher is often solved heuristically by giving evidences that known types of cryptanalysis (like linear or differential cryptanalysis) cannot work. Approaches have seldom been proposed in order to formally deal with security in a general way, i.e. against (still) unknown forms of attack. Vaudenay's decorrelation theory (see §2.4.3 and [320]) proposes the Peanut construction (over which DFC is based) which is basically an r-round Feistel scheme with *decorrelation modules* as a round function. If these decorrelation modules achieve a d-wise decorrelation bias of ε, by using the multiplicative properties and the triangular inequality with a truly random 3-round Feistel construction, one obtains from the Luby-Rackoff Theorem [191] that

$$\mathrm{Dec}_d(F) \leq \left(2d^2 2^{-\frac{m}{2}} + \varepsilon\right)^{\left\lfloor \frac{r}{3} \right\rfloor}$$

[6] The primitive results have been slightly improved by Aoki and Ohta in [10].

where m is the block length in bits. A drawback is that decorrelation with large a d requires very long keys; this explains why one uses $d = 2$ (like in DFC). The security may look very limited but one can still prove security against reasonable generalizations of differential and linear cryptanalysis. One can even prove the resistance against *any* statistical attack which uses simple events on $(x, \Psi(x))$ pairs.

4.1.6 Target Platforms

Block ciphers are typically implemented on very different target platforms, depending on their usage. We briefly discuss in this part the constraints imposed by these different kinds of platforms on the design of fast block ciphers. We successively treat hardware platforms, low-cost microprocessors, and high-end microprocessors.

Hardware

Clearly, the most efficient way to implement a block cipher consists in designing a dedicated electronic circuit, instead of programming it using a general-purpose microprocessor. According to Standaert [302], there exist two main approaches: the first one consists in using a purpose-built hardwired technology, e.g. an Application Specific Integrated Circuit (ASIC) to perform the execution of a block cipher in hardware while the second method consists in using so-called "reconfigurable devices", like Field Programmable Gate Arrays (FPGAs).

ASICs are specifically designed to perform a given computation, and are thus extremely fast and efficient. However, the circuit cannot be modified after its production. FPGAs present similar properties, but a portion of the circuit is dedicated to the reconfigurability of the device instead of the application. This implies a decrease of performances which keeps however very reasonable compared to the benefits of the reconfigurability.

It is worth noticing that ASICs don't define real constraints on the design of block ciphers by themselves, since any cipher which can be implemented in software can be implemented by an ASIC. However, an interesting feature of FPGAs is that they usually can compute very efficiently *any* mapping on 4-bit values. Namely, FPGA can be viewed as a large set of programmable blocks whose logic and routing are user-programmable, where each logic block usually contains a 4-bit arithmetic and logic unit.

Low-Cost Microprocessors

By low-cost microprocessors, we mean processors that can typically be found on low-cost smartcards, or on low-cost embedded devices. In this section, we list the requirements dictated by smartcards regarding the implementation of a block cipher.

- *Low RAM usage*: the resource representing the most important bottleneck in a block cipher implementation on a smartcard is of course the RAM usage. The amount of efficiently usable RAM available on a smartcard is typically in the order of 256 bytes. It is a bit larger on an ARM architecture, but as this type of smart card is devoted to contain more than a simple encryption routine, it would be a good thing not to use a tool large part of the available RAM.

- *ROM usage*: ROM is not as scarce as RAM on a smartcard, so the code size can be greater than the RAM usage. However, it is reasonable not to have a minimal code size (instructions + possible tables) greater than 1024 bytes.

- *Speed*: Obviously, the encryption/decryption (with key schedule) operation should last a minimal time.

- *Complex operations*: It is extremely costly to implement multiplications of operands greater than 8 bits.

As smartcards are one of the preferred targets for modern block ciphers, we give here more insight into two common categories of available smartcards: the 8051 and the ARM architectures. The first one is a typical cheap micro-controller developed by Intel Corp. [136] which is used in many smartcards; it is a low-cost device, typical of everyday life usage such as Pay-TV applications. The second one, developed by ARM Ltd. [12] is supposed to be representative of a powerful smartcard, aimed at providing several advanced services, like security, for instance.

8051 Architecture The 8051 architecture forms one of the most-used basic, low-cost microprocessor used in the smartcards. We can sketch its main characteristics as follows. Typically, a 8051 microcontroller implements an 8-bit Harvard architecture. Depending on versions, performance peeks vary between 1 (at 12 MHz) and 2.5 millions (at 30 MHz) 8-bits instructions per second, but we note that the standard clock frequency for a smartcard is fixed to 3.57 MHz. Usually, 32 to 2304 bytes of RAM are available. However, the RAM is organized in pages of 256 bytes, and accessing data located on another page is a rather costly operation. An 8051 is usually provided with 6 KB to 32 KB of ROM, and 8 general-purpose 8-bit registers are available.

An 8051 possesses a CISC architecture, i.e. it makes use of a complex, variable-sized set of instructions; some of them allow interesting spare of time in the context of a block cipher implementation. Furthermore, it is an accumulator-based architecture which means that data processing instructions specify only one operand; the other one must be stored in the accumulator before the instruction call, and the result will also be placed

in the accumulator. Several addressing modes are supported: for many instructions, operating on memory is as fast as on registers.

The 8051 has a byte-to-byte multiplier, i.e. an instruction allowing to multiply two 8-bits values yielding a 16-bits result. We note however that this instruction may not always be implemented in hardware, which can result in varying execution time among different versions of the chip. Finally, the 8051 has a rotation instruction; it is however limited to rotate an 8-bit value by one bit position (to the left or to the right).

ARM Architecture ARM (for Advanced RISC Machine) is an architecture developped by ARM Ltd. [12]; the ARM architecture is a typical example which can be found on more sophisticated smartcards, for instance.

Its main characteristics are the following: ARM microprocessors are built on a 32-bit, 3-stages pipelined von Neumann architecture, with 3-address instructions: every data processing instruction specifies both the operands and the destination place for the results. 16 registers are available, some of them having a preassigned function (like the program counter, or the stack pointer). It is a LOAD-STORE-architecture, i.e. data processing instructions always operate either on direct values or on registers. Every instruction can be conditionally executed according the the state of flags, and there is a barrel shifter, which means that the second operand in a data processing instruction can optionaly be shifted or rotated before being processed. This architecture can furthermore multiply two 32-bit values, yielding the 32 low-order bits of the product (some more advanced ARM processors have the possibility to get the full 64-bit result). Finally, there is typically a few kilobytes of RAM at disposal.

32/64-bit Microprocessors

This category of microprocessors is the one found in common computers. Although some constructors are, at the time of writing, shipping machine built on top of 64-bit processors, 32-bit architectures should remain a standard platform for a few years. For a long time, 64-bit RISC CPUs were not available on low-cost, desktop computers used by everybody. They were reserved to mainframes and to very expensive work stations, sold by Hewlett-Packard, Compaq and other companies. This situation is changing as most chip manufacturers are shipping 64-bit microprocessors and one can expect a slow increase of such CPUs in a larger public.

Typically, a 64-bit RISC CPU offer many more registers than a CISC architecture, while the latter can usually do complex operations (like bit rotation) faster. Another advantage of RISC architectures is that it is always possible to get the fastest possible implementation with a high-level language, while it is not the case in a CISC architecture: an optimized version of a program must frequently be written in assembly code.

Processor	cache size [kB]	Note
Alpha 21164	8	(data)
Alpha 21264	64	(data)
AMD Athlon XP	128	(data + code)
AMD Athlon MP	128	(data + code)
AMD Opteron	64	(data)
Intel Pentium III	16	(data)
Intel Pentium IV	8/16 (Prescott)	(data)
Intel Xeon	8	(data)
Intel Itanium	16	(data)
Intel Itanium2	16	(data)
PowerPC G4	32	(data + code)
PowerPC G5	32	(data)
UltraSparc II	16	(data)
UltraSparc III	64	(data)

Figure 4.5: Amount of L1 cache for various high-end microprocessors

We figured out that the main bottleneck of these architectures regarding the design of fast block ciphers is the amount of the fastest memory (typically named L1 cache) at disposal, as fetching memory cells which are not present at this level of the memory hierarchy results in very costly (in terms of clock cycles) cache misses. This implies an upper-bound on the amount of precomputed data necessary to execute an optimized implementation. Fig. 4.5 lists these figures for some popular 32/64-bit microprocessors.

4.2 Fast Diffusive Components

Although linear perfect diffusion primitives (i.e. MDS encoding matrices or linear multipermutations), are nowadays widely used in block ciphers, very little systematic work has been published so far, to the best of our knowledge, on how to find "efficient" ones. In this part, we present a first attempt to systematically address this problem by considering software implementations of such components on various platforms. Interestingly, this opens several combinatorial problems which we investigate, and finally, we propose a sequence of new efficient constructions of 4×4 and of 8×8 MDS matrices which can be used e.g. in block ciphers. We refer the reader to §4.1.3 for an introduction on perfect diffusive components.

It is actually very difficult to mathematically define what is an "optimal" matrix in terms of the efficiency of its implementation on, since there exist a large number of criteria which may be taken into account; furthermore, these criteria are usually very dependent of the target platform. In this

part, we have purposedly chosen to treat the problem of constructing MDS matrices whose implementation is very efficient on most low-cost platforms, like 8051, or ARM architectures (see §4.1.6 for a technical overview of these platforms). For this, we isolate a few criteria which look important, and we derive several optimality results based on these criteria.

Let $\mathfrak{K} = (\mathcal{K}, +, \times)$ be a field of characteristic 2. Typically, we have $\mathfrak{K} = \mathrm{GF}(256)$. We consider linear multipermutations defined from $\mathcal{K}^p$ to $\mathcal{K}^q$ and we denote by $\mathbf{M}$ a $q \times p$ matrix whose elements lie in $\mathcal{K}$. Let furthermore $\mathbf{M}_{i,j}$ denote the matrix element on row i and column j, where $1 \leq i \leq q$ and $1 \leq j \leq p$. Let us furthermore denote vectors taking elements in $\mathcal{K}$ by $\mathbf{x}, \mathbf{y}, \ldots$, and the i-th component of the vector $\mathbf{x}$ by $\mathbf{x}_i$. We interpret a linear multipermation by the matrix multiplication $\mathbf{x} \mapsto \mathbf{M} \times \mathbf{x}$ which can simply be rewritten as

$$\mathbf{x}_i = \sum_{j=1}^{p} \mathbf{M}_{i,j}\mathbf{x}_j \qquad \text{for } i = 1, \ldots, q.$$

Here, we note that the addition in $\mathfrak{K}$ can be efficiently implemented using a XOR operation, denoted $\oplus$, which is available on virtually all microprocessor architectures. We describe now how to implement in an efficient way such mappings, on the one hand, on 32/64-bit architectures which dipose of large amounts of very fast memory, and on the other hand, on 8-bit low-cost architectures which do *not* have large amounts of memory at disposal.

4.2.1 Performances of Linear Multipermutations

Modern 32/64-bit microprocessors typically have relatively large quantities of very fast memory, also called *cache memory*. By "very fast", we mean the fastest accessible cache memory, usually named L1 cache[7]. Fig. 4.5, page 160, lists typical high-end microprocessors and the quantity of L1 cache they have at disposal.

When enough fast memory is available, a well-known and quite simple implementation strategy (described in [81], for instance) can be applied. Namely, the columns of $\mathbf{M}$ can be partitioned into several sub-columns whose size correspond to the microprocessor word size or less. Let us denote by w the size of the words in terms of elements of $\mathcal{K}$. Then, all possible multiplications can be precomputed and put in a lookup table. In other words, we consider the matrix $\mathbf{M}$ as a block matrix of type $\lceil \frac{q}{w} \rceil \times p$, the output vector $\mathbf{y}$ as a block vector of $\lceil \frac{q}{w} \rceil$ elements, and where every block are vectors of w elements of $\mathcal{K}$, except the blocks in the last row which may be smaller if w does not divide q. For instance, let us consider the example of a 32-bit

[7]Note that we focus here on L1 cache, since accessing data which are not present at this level of cache hierarchy produces penalties of a large number of clock cycles on most of microprocessors. However, it has been shown by Lipmaa [188] that in certain precise cases, one can take profit of choosing an alternate implementation strategy exploiting higher levels of the cache hierarchy.

architecture, where $\mathfrak{K} = \mathrm{GF}(2^8)$, and $p = q = 4$ (as in AES or in FOX64, for instance). This fixes $w = 4$. We define four tables T^j, with $1 \leq j \leq 4$ of 256 4-words vectors such that

$$\mathsf{T}^j : u \mapsto \begin{pmatrix} \mathbf{M}_{1,j} \cdot u \\ \mathbf{M}_{2,j} \cdot u \\ \mathbf{M}_{3,j} \cdot u \\ \mathbf{M}_{4,j} \cdot u \end{pmatrix} \qquad \text{for } u \in \mathcal{K}$$

Then the evaluation of $\mathbf{y} = \mathbf{M} \times \mathbf{x}$ reduces to the computation of

$$\mathbf{y} = \mathsf{T}^1(\mathbf{x}_1) \oplus \mathsf{T}^2(\mathbf{x}_2) \oplus \mathsf{T}^3(\mathbf{x}_3) \oplus \mathsf{T}^4(\mathbf{x}_4)$$

which represents three XORs and four table lookups. The memory needs are, in this example, four tables of 256 32-bit values each, i.e. 4096 bytes; this clearly fits in the L1 cache of modern high-end microprocessors. An 8×8 matrix multiplication over the same field $\mathfrak{K}$ would involve 7 XORs, 8 table lookups, and 8 tables of 256 64-bit values, i.e. 16384 bytes in total. Under this approach, performances only depend on the external characteristics of $\mathbf{M}$ and of the microprocessor, i.e. p, q, w, and $\#\mathcal{K}$, but they do not depend on the internal structure of $\mathbf{M}$. Interestingly, one can furthermore combine this strategy with a possible simultaneous evaluation of 8-bit substitution boxes by directly taking them into account in the precomputed tables (see §4.3.5 for detailed explanations about an example of this strategy of implementation).

The situation becomes more problematic for low-cost 8-bit architectures, since one cannot afford to waste too much memory for storing precomputed data: thus, the matrix multiplication must be computed on-the-fly, and in this case, the internal structure of $\mathbf{M}$ has an impact on the performances. Obviously, it is necessary to evaluate in the order of $p \cdot q$ operations in $\mathfrak{K}$ as no element can be equal to the additive neutral element of $\mathfrak{K}$ (otherwise, it would be a contradiction to Th. 4.1.3).

A first solution would consist of expressing each element x of $\mathcal{K}$ as $x = g^i$, where g is a generator of the multiplicative group of $\mathfrak{K}$, and to store the precomputed mappings $x \mapsto i$ and $i \mapsto x$ (this costs 512 bytes of memory if $\mathfrak{K} = \mathrm{GF}(2^8)$). Thus, any multiplication in $\mathfrak{K}$ can be implemented with help of three table lookups and an addition modulo 255. A table lookup can even be saved as the multiplications are actually between a variable operand and a *constant* one, thus the logarithm of the constant terms can be hard-coded. However, this approach remains quite costly.

Some multiplications tables are quite simple though: for instance, the multiplication by 1 (the multiplicative neutral element of $\mathfrak{K}$) is trivial and do not cost anything. From this point, we will adopt the following approach: all multiplication tables by $\mathbf{M}_{i,j}$ such that $\mathbf{M}_{i,j} \neq 1$ are pre-computed. Thus, the three following properties of matrices are playing an important role.

Definition 4.2.1 (Number of occurences of 1). *Let $\mathcal{K}^*$ be a set including a distinguished element denoted 1. Let $\mathbf{M}$ be a $q \times p$ matrix with $\mathbf{M}_{i,j} \in \mathcal{K}^*$ for all i and j. Then $\varkappa(\mathbf{M})$ is defined to be the number of entries (i,j) of $\mathbf{M}$ such that $\mathbf{M}_{i,j} = 1$ and is called the* number of occurences *of 1:*

$$\varkappa(\mathbf{M}) = \#\{\mathbf{M}_{i,j} : \mathbf{M}_{i,j} = 1\}.$$

Definition 4.2.2 (Number of (non-trivial) entries). *Let $\mathcal{K}^*$ be a set including a distinguished element denoted 1. Let $\mathbf{M}$ be a $q \times p$ matrix with $\mathbf{M}_{i,j} \in \mathcal{K}^*$ for all i and j. Then $\vartheta(\mathbf{M})$ is defined to be the number of different entries (i,j) of $\mathbf{M}$ and is called the* number of entries *of $\mathbf{M}$:*

$$\vartheta(\mathbf{M}) = \#\{\mathbf{M}_{i,j} : 1 \leq i \leq q \text{ and } 1 \leq j \leq p\}.$$

Furthermore, if 1 occurs in $\mathbf{M}$, we define $\vartheta_1(\mathbf{M}) = \vartheta(\mathbf{M}) - 1$ as the number of non-trivial entries *of $\mathbf{M}$. Otherwise, $\vartheta_1(\mathbf{M}) = \vartheta(\mathbf{M})$.*

Under this approach, one needs to store pre-computed tables having a total size equal to

$$\vartheta_1(\mathbf{M}) \times \#\mathcal{K}$$

to be able to implement a matrix multiplication, and the total time complexity may be expressed as $q(p-1)$ XORs and

$$\vartheta_1(\mathbf{M}_{1,.}) + \ldots + \vartheta_1(\mathbf{M}_{q,.})$$

table lookups, where $\mathbf{M}_{i,.}$ denotes the i-th column of $\mathbf{M}$. Clearly, the quantity $\varkappa(\mathbf{M})$ is linked to the data complexity of the implementation while $\vartheta_1(\mathbf{M})$ is related to its time complexity. Thus, a natural goal consists in finding matrices or skeletons of matrices minimizing $\vartheta_1(.)$ and maximizing $\varkappa(.)$.

A further possibility to trade some multiplication tables against some computation units consists in filling $\mathbf{M}$ with "efficient" entries, i.e. elements of $\mathcal{K}$ different of 1 which can be efficiently multiplied with any operand. For instance, if $\mathfrak{K} = \mathrm{GF}(2^8)$, one can interpret the elements of the field by polynomials $a_0 + a_1 x + \ldots + a_7 x^7$ of degree at most seven with coefficients a_i in $\mathrm{GF}(2)$, and represent these polynomials as 8-bit strings $a_7 \cdots a_1 a_0$. In this case, the multiplication by x can simply be implemented by a logical left shift by one bit and a conditional XOR with a constant when a carry bit is set. Note that one should be especially careful about timing attacks; we give an example of such a constant-time implementation on an 8051 architecture in §4.3.5.

Similary, the division by x can be implemented with the help of a logical right shift and a conditional XOR with a constant. Furthermore, if a matrix $\mathbf{M}$ includes to elements x and $x+1$, then we can omit the multiplication

table by $x+1$, since one can emulate it by one multiplication by x followed with an XOR.

Another example is the multiplication by constants which are even power of a coefficient for which we make use of a precomputed table. For instance, if have the coefficients x and x^2 in a matrix, we do not need a precomputed table for the multiplication by x^2 since it can be emulated by two lookups in the table storing the results of a multiplication by x.

Finally, we can also algebraically optimize implementations by playing with temporary variables storing intermediate results during the computations.

4.2.2 Bi-Regular Arrays

We concentrate now on finding MDS matrices with high $\varkappa$ and low ϑ coefficients. The following definition introduces *bi-regular arrays* (and their converse, namely *bi-singular arrays*) which are useful objects to build MDS matrices. Clearly, a bi-singular array cannot be an MDS matrix since it contains a 2×2 singule sub-determinant. Hence, being bi-regular is a necessary (but not sufficient) condition for being an MDS matrix.

Our approach for constructing MDS matrices with high $\varkappa$ and low ϑ coefficients is first to construct a bi-regular array, second to assign elements to some non-zero field values until we get an MDS matrix. We can e.g. look at random values until it succeeds or concentrate on efficient GF elements, and using a brute-force search.

Definition 4.2.3 (Bi-Regular Arrays). *Let $\mathcal{K}^*$ be a set including a distinguished element denoted 1. We say that a 2×2 array with entries in $\mathcal{K}^*$ is* bi-regular *if at least one row and one column have the property of having fferent entries; a $q \times p$ array with entries in $\mathcal{K}^*$ is* bi-regular *if all 2×2 sub-arrays are bi-regular.*

Definition 4.2.4 (Bi-Singular Arrays). *An array which is not bi-regular is called* bi-singular*.*

We state now two theorems summarizing the optimal values of both $\varkappa$ and ϑ_1 for $1 \leq p, q \leq 8$ such that it is still possible to build bi-regular arrays. We refer to [152] for their proof.

Theorem 4.2.1. *The maximal possible value of $\varkappa(\mathbf{M})$ such that it is possible to find a $q \times p$ bi-regular array $\mathbf{M}$ for $1 \leq p, q \leq 8$ is given in Fig. 4.6.*

Theorem 4.2.2. *The minimal possible value of $\vartheta_1(\mathbf{M})$ such that it is possible to find a $q \times p$ bi-regular array $\mathbf{M}$ for $1 \leq p, q \leq 8$ is given in Fig. 4.7.*

	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
3	4	6	7	8	9	10	11
4	5	7	9	10	12	13	14
5	6	8	10	12	13	14	17
6	7	9	12	13	16	18	19
7	8	10	13	14	18	21	22
8	9	11	14	17	19	22	24

Figure 4.6: Maximal possible value for $\varkappa(\mathbf{M})$ for a bi-regular array $\mathbf{M}$

	2	3	4	5	6	7	8
2	2	2	2	3	3	3	3
3	2	2	3	3	3	3	4
4	2	3	3	3	4	4	4
5	3	3	3	3	4	4	4
6	3	3	4	4	4	4	5
7	3	3	4	4	4	4	5
8	3	4	4	4	5	5	5

Figure 4.7: Minimal possible value for $\vartheta_1(\mathbf{M})$ for a bi-regular array $\mathbf{M}$

4.2.3 New Constructions

We study now constructions with $p = q = 4$ over the field $\mathfrak{K} = \mathrm{GF}(256)$. Elements are represented as polynomials of degree at most 7 over GF(2), the $a_0 + a_1 x + \cdots + a_7 x^7$ polynomial being represented by the bitstring $a_7 \cdots a_1 a_0$. Formally, x represents a root of an irreducible polynomial over GF(2) of degree 8.

The AES Matrix

Here is the MDS matrix which is the core diffusive component of the AES [81, 245]:

$$\mathbf{M}_{\mathsf{AES}} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \tag{4.1}$$

Multiplication by x results in a logical shift and a conditional XOR, while the multiplication by $x+1$ requires an additional XOR . In this case, $\vartheta(\mathbf{M}_{\mathsf{AES}}) = 3$ is optimal according to our criteria and Th. 4.2.2, but $\varkappa(\mathbf{M}_{\mathsf{AES}}) = 8$ is not. As described in [81], a multiplication by (4.1) can be implemented (in a pseudo-C notation) using 15 XORs, 4 table lookups and 3 temporary

```
t = a[0] ^ a[1] ^ a[2] ^ a[3]; /* a is the input vector */
u = a[0];
v = a[0] ^ a[1]; v = time[v]; a[0] = a[0] ^ v ^ t;
v = a[1] ^ a[2]; v = time[v]; a[1] = a[1] ^ v ^ t;
v = a[2] ^ a[3]; v = time[v]; a[2] = a[2] ^ v ^ t;
v = a[3] ^ u;    v = time[v]; a[3] = a[3] ^ v ^ t;
```

Figure 4.8: Multiplication by $\mathbf{M}_{\mathsf{AES}}$

variables with a single 256-byte table, namely the multiplication by x (see Fig. 4.8).

As a side remark, we note that AES also requires to implement the inverse MDS matrix for the decryption operation. Unfortunately, this inverse matrix (described in Eq. (4.2)) is not as implementation-friendly as Eq. (4.1), since it requires to implement the multiplication by four different coefficients:

$$\mathbf{M}_{\mathsf{AES}}^{-1} = \begin{pmatrix} x^3+x^2+x & x^3+x+1 & x^3+x^2+1 & x^3+1 \\ x^3+1 & x^3+x^2+x & x^3+x+1 & x^3+x^2+1 \\ x^3+x^2+1 & x^3+1 & x^3+x^2+x & x^3+x+1 \\ x^3+x+1 & x^3+x^2+1 & x^3+1 & x^3+x^2+x \end{pmatrix} \quad (4.2)$$

In this case, we need either considerably more accesses to the table storing all possible results of an element multiplied by x, or we need additional tables. Another possibility is to consider Eq. (4.2) as the product of $\mathbf{M}_{\mathsf{AES}}$ with another matrix, namely

$$\mathbf{M}_{\mathsf{AES}}^{-1} = \mathbf{M}_{\mathsf{AES}} \times \begin{pmatrix} x^2+1 & 0 & x^2 & 0 \\ 0 & x^2+1 & 0 & x^2 \\ x^2 & 0 & x^2+1 & 0 \\ 0 & x^2 & 0 & x^2+1 \end{pmatrix},$$

and accordingly, to implement *two* matrix multiplications, one of which being required by the encryption operation, and the other being reasonably simple to implement.

An Efficient 4×4 Matrix

As we have seen, $\varkappa^{4,4} = 9$ and $\vartheta^{4,4} = 3$ and we can hit both optimal criteria ($\mathbf{M}_4$ in Eq. (4.3)); let us furthermore consider a second matrix $\mathbf{M}'_4$, which is a permuted version of $\mathbf{M}_4$.

$$\mathbf{M}_4 = \begin{pmatrix} a & 1 & 1 & 1 \\ 1 & 1 & b & a \\ 1 & a & 1 & b \\ 1 & b & a & 1 \end{pmatrix} \qquad \mathbf{M}'_4 = \begin{pmatrix} a & 1 & 1 & 1 \\ 1 & a & 1 & b \\ 1 & b & a & 1 \\ 1 & 1 & b & a \end{pmatrix} \quad (4.3)$$

```
u    = a[0] ^ a[1] ^ a[2] ^ a[3]; /* a is the input vector */
a[0] = u ^ timeap1[a[0]];     v    = timeap1[a[1]];
a[2] = timeap1[a[2]];         a[3] = timeap1[a[3]];
a[1] = u ^ v ^ timebp1[a[3]]; a[3] = u ^ a[3] ^ timebp1[a[2]];
a[2] = u ^ a[2] ^ timebp1[v];
```

Figure 4.9: Multiplication by $\mathbf{M}'_4$

One can easily verify that necessary conditions for $\mathbf{M}'_4$ being an MDS matrix are: 0, 1, a and b must be pairwise different, $a \neq b^2$, $a \neq b+1$, and $a^2 \neq b$. If we make use of two multiplication tables (namely, by $a+1$ and by $b+1$), we can implement a multiplication by $\mathbf{M}'_4$ as illustrated by Fig. 4.9. This implementation needs 10 XORs, 2 temporary variables, 7 table lookups in two 256-byte tables. This allows us to decrease the overall number of temporary variables and of operations (at the cost of a supplementary precomputed table), if the XOR operations and table lookups generate identical costs. A permuted version of $\mathbf{M}_4$ is used in the design of FOX64; note that as this latter does make use of a self-inverting high-level scheme, we do not require that the inverse of $\mathbf{M}_4$ to be efficiently implementable.

Efficient 8×8 Matrices

Here, we give explicit constructions with $p = q = 8$ over $\mathfrak{K} = \mathrm{GF}(256)$.

Circulating-Like Matrix By using the same circulating-like construction of the AES matrix, but with $p = q = 8$, we obtain $\varkappa = 21$ and $\vartheta = 7$ which are not optimal figures.

$$\mathbf{M}_8 = \begin{pmatrix} f & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & a & b & c & d & e & f \\ 1 & f & 1 & a & b & c & d & e \\ 1 & e & f & 1 & a & b & c & d \\ 1 & d & e & f & 1 & a & b & c \\ 1 & c & d & e & f & 1 & a & b \\ 1 & b & c & d & e & f & 1 & a \\ 1 & a & b & c & d & e & f & 1 \end{pmatrix}$$

Many different possibilities for filling the coefficients exist; we give here as illustration two different examples. For GF(256) represented by the irreducible polynomial $x^8 + x^4 + x^3 + x^2 + 1$ over GF(2), a possible combination is given by $a = x+1$, $b = x^3+1$, $c = x^3+x^2$, $d = x$, $e = x^2$ and $f = x^4$. Note that we need a single precomputed table, namely the multiplication by x. If we can afford two precomputed multiplication tables (by x and by x^{-1}, in this case), when using $x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1$ as field representation,

```
y[0] = x[0] ^ x[1] ^ x[2] ^ x[3] ^ x[4] ^ x[5] ^ x[6] ^
       xtime[x[7]];
y[1] = x[1] ^ x[0] ^ x[7] ^ xtime[x[1] ^ x[3] ^ xtime[4]] ^
       xm1time[x[2] ^ x[5] ^ xm1time[x[2] ^ x[6]];
y[2] = x[0] ^ x[6] ^ x[7] ^ xtime[x[0] ^ x[2] ^ xtime[3]] ^
       xm1time[x[1] ^ x[4] ^ xm1time[x[1] ^ x[5]];
y[3] = x[6] ^ x[5] ^ x[7] ^ xtime[x[6] ^ x[1] ^ xtime[2]] ^
       xm1time[x[0] ^ x[3] ^ xm1time[x[0] ^ x[4]];
y[4] = x[5] ^ x[4] ^ x[7] ^ xtime[x[5] ^ x[0] ^ xtime[1]] ^
       xm1time[x[6] ^ x[2] ^ xm1time[x[6] ^ x[3]];
y[5] = x[4] ^ x[3] ^ x[7] ^ xtime[x[4] ^ x[6] ^ xtime[0]] ^
       xm1time[x[5] ^ x[1] ^ xm1time[x[5] ^ x[2]];
y[6] = x[3] ^ x[2] ^ x[7] ^ xtime[x[3] ^ x[5] ^ xtime[6]] ^
       xm1time[x[4] ^ x[0] ^ xm1time[x[4] ^ x[1]];
y[7] = x[2] ^ x[1] ^ x[7] ^ xtime[x[2] ^ x[4] ^ xtime[5]] ^
       xm1time[x[3] ^ x[6] ^ xm1time[x[3] ^ x[0]];
```

Figure 4.10: Multiplication by $\mathbf{M}_8$

another possible combination is $a = x + 1$, $b = x^{-1} + x^{-2}$, $c = x$, $d = x^2$, $e = x^{-1}$ and $f = x^{-2}$. An implementation using 29 table lookups, 71 XORs is given in Fig. 4.10.

Matrix with Rectangle Patterns We obtain $\varkappa = 15$ and $\vartheta = 5$ so this is optimal for which concerns the number of different coefficients with the following matrix.

$$\mathbf{M}_{\mathsf{rect}} = \begin{pmatrix} b & a & c & b & d & c & 1 & d \\ b & c & a & d & b & 1 & c & 1 \\ c & b & d & a & 1 & b & 1 & c \\ c & d & b & 1 & a & 1 & b & d \\ d & c & 1 & b & 1 & a & d & b \\ d & 1 & c & 1 & b & d & a & c \\ 1 & d & 1 & c & d & b & c & a \\ 1 & 1 & d & d & c & c & b & b \end{pmatrix}$$

Representing GF(256) with $x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1$ as irreducible polynomial, a possible combination is given by $a = x^{-3} + x^{-1}$, $b = x^{-2} + x^{-1} + 1$, $c = x^4 + x$ and $d = x$. With $x^8 + x^4 + x^3 + x^2 + 1$ as irreducible polynomial, a valid combination is $a = x + 1$, $b = x^4 + 1$, $c = x^4 + x$ and $d = x$. Using these coefficients, we are able to implement this matrix multiplication with the same amount of table lookups (i.e. 16), 54 XORs instead of 56 and two less temporary variables than the matrix used by the designers of Khazad (as described in [18]), for instance (see Fig. 4.11). Here,

```
t0 = x[0]^x[1]; t1 = x[0]^x[2];
t2 = x[3]^x[5]; t3 = x[1]^x[4];
t4 = x[2]^x[4]; t5 = x[5]^x[7];
t6 = x[3]^x[6]; t7 = x[4]^x[6];
r1 = t1^t5;     r2 = t2^t4;
r3 = t3^t6;     r4 = t2^t6;
y[0] = t0^t6^xtime[t3^t5^x[2]]^x4time[t1^t2];
y[1] = r1^x[4]^xtime[t6^x[1]^x[2]]^x4time[t0^t7];
y[2] = r4^t3^xtime[r1^t2]^x4time[t0^t5];
y[3] = r2^x[6]^xtime[t0^x[4]^x[7]]^x4time[t1^x[6]];
y[4] = r2^x[7]^xtime[t1^x[5]^ x[6]]^x4time[x[2]^x[3]^x[7]];
y[5] = r3^xtime[r1^x[7]]^x4time[t4^x[7]];
y[6] = r1^xtime[r3^x[7]]^x4time[r4];
y[7] = t0^x[6]^x[7]^xtime[r2]^x4time[t5^t7];
```

Figure 4.11: Multiplication by $\mathbf{M}_{\mathsf{rect}}$

we use two precomputed tables, namely `xtime`[.] (multiplication by x) and `x4time`[.] (multiplication by x^4). We might do even better by dedicated optimizations.

4.2.4 Open Problems

MDS matrices are a well-known way to build linear multipermations, i.e. optimal diffusion components which can be used as building blocks of cryptographic primitives, like block ciphers and hash functions. Although their implementation is quite straightforward on 32/64-bit architectures, which have large data L1 caches and thus allow to store large precomputed tables, we need to evaluate the matrix multiplication on-the-fly on low-cost 8-bit architectures, and we can afford only a very limited amount of precomputed data.

In this part, we have studied MDS matrices under the angle of efficiency, chosen an implementation strategy, defined mathematical criteria according to this strategy; furthermore, we give new constructions for efficient 4×4 and 8×8 matrices over GF(256).

We would like to stress out that, although not completely arbitrary, our choice of implementation strategy may not be optimal in certain scenarii, or under certain precise application-driven constraints. Thus, future potential investigations may go in the direction of hardware implementations of linear multipermutations, for instance, which we do not cover. Furthermore, we may extend our mathematical considerations criteria to the specific case of SPNs; such matrices must have inverses which are also efficient, for fast decryption operations.

4.3 The FOX Family of Block Ciphers

In this section, we describe the design of a new family of block ciphers, named FOX, developed for the company *MediaCrypt AG* [218]. The main goals of this design, besides a very high security level, are a large implementation flexibility on various platforms as well as high performances. The high-level structure is based on a Lai-Massey scheme, while the round functions are substitution-permutation networks. In addition, we propose a new design of strong and efficient key-schedule algorithms.

The section is organized as follows: in §4.3.1, we motivate the need for a new design, in §4.3.2, we give a formal description of the block ciphers, then we describe the rationales behind our design in §4.3.3; the security foundations are developed in §4.3.4, where we recall some Luby-Rackoff-like results about the Lai-Massey scheme, we analyse the security of FOX towards linear and differential cryptanalysis and we discuss some issues related to other attacks. Finally, we discuss several implementations aspects in §4.3.5, where some results about the performances of the ciphers are given.

4.3.1 Motivation

Inevitably, a burning question before designing a new block cipher consists in knowing whether it is really useful or not ! First of all, a very large number of algorithms have been published since the beginning of the 80's (see for instance Fig. 2.2, page 11), and among this list, a reasonable fraction thereof is still considered as both sufficiently fast and secure for any practical use. Furthermore, the AES [3] and NESSIE [247] efforts, among others, have resulted in a number of new proposals of block ciphers, in parallel with serious advances in the cryptanalysis field.

However, it is noteworthy that there exists a clear trend in direction of lightweight and fast key-schedule algorithms, as well as substitution boxes based on purely algebraic constructions. In a parallel way, we observe that, on the one hand, several of the last published attacks against block ciphers take often advantage of exploiting "simple" key-schedule algorithms (nice illustrations thereof are certainly Muller's attack [233] against Khazad and Phan's impossible differential cryptanalysis of 7-rounds AES [264]), and, on the other hand, algebraic S-boxes are helpful to Courtois-Pieprzyk algebraic attacks [70], and lead to puzzling properties as shown e.g. by [16, 104, 228, 235] (see §2.2.3, page 25, as well). Finally, despite of the standardization effors, there still exists in the business world some demand of proprietary algorithms being publicly reviewed.

Our first goal is to offer a serious and secure alternative to block ciphers following present trends; we have explicitely chosen to *avoid a lightweight key-schedule and a pure algebraic construction for the S-boxes*. Our second goal is to reach the highest possible flexibility, being in terms of round

Name	Block size (in bits)	Key size (in bits)	Rounds number
FOX64	64	128	16
FOX128	128	256	16
FOX64/k/r	64	k	r
FOX128/k/r	128	k	r

Figure 4.12: Members of the FOX family

number, key size, block size and in terms of implementation issues. For instance, we feel that it is still useful to propose a 64-bit block size flavour for back-compatibility reasons. Our last motivation was to design a family of block ciphers which compares favourably with the performances of the fastest block ciphers on hardware, 8-bit, 32-bit, and 64-bit architectures.

Finally, we have adopted the following security objectives for FOX: the mapping $(x, k) \mapsto \mathsf{e}_k(x)$ should be indistinguishable from a uniformly distributed random function, the key schedule should output pseudo-random sequences of subkeys, and the best attacks applying to FOX should be the generic ones (i.e. exhaustive key search, time-memory tradeoffs, dictionnary attacks, ...).

4.3.2 Description

The family consists in two main block cipher designs, the first one having a 64-bit blocksize and the other one a 128-bit blocksize. Each design allows a *variable number of rounds* and a *variable key size* up to 256 bits. The different members of the FOX family are listed in Fig. 4.12. The following conditions *must* hold in the case of FOX64/k/r and FOX128/k/r: the number of rounds r must satisfy $12 \leq r \leq 255$, while the key length k must satisfy $0 \leq k \leq 256$, with k multiple of 8.

Notations and Representation of GF $\left(2^8\right)$ We first describe some generic conventions used for the description of the FOX family. A variable x written with the suffix "$_{(n)}$" (*i.e.* $x_{(n)}$) indicates that x has a length of n bits. For instance, $y_{(1)}$ is a single-bit variable and $f_{(64)}$ is a 64-bit value. The suffix will be omitted if the context is clear. A variable x written with the suffix "$_{[a...n]}$" (*i.e.* $x_{[a...b]}$) indicates the bit subset of the variable x beginning at position a (inclusive) and ending at position b (inclusive). Indexed variables are denoted as follows: x_i is a variable x indexed by i. A variable x indexed by i with a length of ℓ bits is denoted $x_{i(\ell)}$. A C-like notation is used for indexing which means that indices begin with 0. The suffix l (r) is used to denote the left (right) half of a variable. For instance, x_{l} is the left half of the variable x and x_{r} is its right half. The suffixes ll, lr, rl, rr are used to

denote *quarters* of a variable. For instance, $x = x_{\mathsf{ll}}||x_{\mathsf{lr}}||x_{\mathsf{rl}}||x_{\mathsf{rr}}$. In general, the input of a function f is denoted x and its output y.

Some of the internal operations used in FOX are the addition and the multiplication in the finite field with 256 elements $\mathrm{GF}(2^8)$. We describe now the *representation* of $\mathrm{GF}(2^8)$ which is used in the FOX definition.

Definition 4.3.1 (Irreducible Polynomial $\mathsf{P}(\alpha)$). *The irreducible polynomial which represents* $\mathrm{GF}\left(2^8\right)$ *in the FOX block cipher family is the irreducible polynomial over* $\mathrm{GF}\left(2\right)$ *defined by*

$$\mathsf{P}(\alpha) = \alpha^8 + \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + 1.$$

Elements of the field are polynomials in α of degree at most 7 with coefficients in $\mathrm{GF}(2)$ and the addition and multiplication operations are performed modulo $\mathsf{P}(\alpha)$. The field elements are identified to 8-bit strings as follows.

Definition 4.3.2 (Representation of $\mathrm{GF}(2^8)$ Elements). *Let s be an 8-bit binary string*

$$s = s_{0(1)}||s_{1(1)}||s_{2(1)}||s_{3(1)}||s_{4(1)}||s_{5(1)}||s_{6(1)}||s_{7(1)}$$

The corresponding field element is then defined to be

$$s_{0(1)}\alpha^7 + s_{1(1)}\alpha^6 + s_{2(1)}\alpha^5 + s_{3(1)}\alpha^4 + s_{4(1)}\alpha^3 + s_{5(1)}\alpha^2 + s_{6(1)}\alpha + s_{7(1)}.$$

In this chapter, a *big-endian* ordering is assumed, *i.e* the index of the most significant part in a variable is equal to 0, while the index corresponding to the least significant part is the largest one. Here is an example: a 128-bit value $q_{(128)}$ can be written as

$$\begin{aligned} q_{(128)} &= r_{0(64)}||r_{1(64)} \\ &= s_{0(32)}||s_{1(32)}||s_{2(32)}||s_{3(32)} \\ &= t_{0(8)}||t_{1(8)}||\ldots||t_{14(8)}||t_{15(8)} \\ &= u_{0(1)}||u_{1(1)}||\ldots||u_{126(1)}||u_{127(1)} \end{aligned}$$

High-Level Structure

In this part, we describe the skeleton and the encryption/decryption processes for both FOX64 and FOX128. For this purpose, we will follow a top-down approach.

FOX64/k/r Skeleton The 64-bit version of FOX is the $(r-1)$-times iteration of a round function denoted lmor64, followed by the application of a slightly modified version of lmor64, named lmid64. lmio64 is a function used during the decryption operation. Formally, lmor64, lmio64 and lmid64 take

all a 64-bit input $x_{(64)}$, a 64-bit round key $rk_{(64)}$ and return a 64-bit output $y_{(64)}$:

$$\mathsf{lmor64}, \mathsf{lmio64}, \mathsf{lmid64} : \begin{cases} \{0,1\}^{64} \times \{0,1\}^{64} & \rightarrow & \{0,1\}^{64} \\ (x_{(64)}, rk_{(64)}) & \mapsto & y_{(64)} \end{cases}$$

FOX64 **Encryption** The encryption $c_{(64)}$ by FOX64/k/r of a 64-bit plaintext $p_{(64)}$ is defined as

$$c_{(64)} = \mathsf{lmid64}(\mathsf{lmor64}(\ldots(\mathsf{lmor64}(p_{(64)}, rk_{0(64)}), \ldots, rk_{r-2(64)}), rk_{r-1(64)})$$

where

$$rk_{(r \cdot 64)} = rk_{0(64)} || rk_{1(64)} || \ldots || rk_{r-1(64)}$$

is the subkey stream produced by the key schedule algorithm from the key $k_{(\ell)}$.

FOX64 **Decryption** The decryption $p_{(64)}$ by FOX64/k/r of a 64-bit ciphertext $c_{(64)}$ is defined as

$$p_{(64)} = \mathsf{lmid64}(\mathsf{lmio64}(\ldots(\mathsf{lmio64}(c_{(64)}, rk_{r-1(64)}), \ldots, rk_{1(64)}), rk_{0(64)})$$

where

$$rk_{(r \cdot 64)} = rk_{0(64)} || rk_{1(64)} || \ldots || rk_{r-1(64)}$$

is the subkey stream produced by the key schedule algorithm from the key $k_{(\ell)}$, as for the encryption.

FOX128/k/r **Skeleton** Similarly to the definition of FOX64, the 128-bit version of FOX is the $(r-1)$-times iteration of a round function denoted elmor128, followed by the application of a modified version of elmor128 named elmid128. elmio128 is a function used during the decryption operation. Formally, elmor128, elmio128 and elmid128 all take a 128-bit input $x_{(128)}$, a 128-bit round key $rk_{(128)}$ and return a 128-bit output $y_{(128)}$:

$$\mathsf{elmor128}, \mathsf{elmio128}, \mathsf{elmid128} : \begin{cases} \{0,1\}^{128} \times \{0,1\}^{128} & \rightarrow & \{0,1\}^{128} \\ (x_{(128)}, rk_{(128)}) & \mapsto & y_{(128)} \end{cases}$$

FOX128 **Encryption** The encryption $c_{(128)}$ by FOX128/k/r of a 128-bit plaintext $p_{(128)}$ is defined as

$$c_{(128)} =$$
$$\mathsf{elmid128}(\mathsf{elmor128}(\ldots(\mathsf{elmor128}(p_{(128)}, rk_{0(128)}), \ldots, rk_{r-2(128)}), rk_{r-1(128)})$$

where

$$rk_{(r \cdot 128)} = rk_{0(128)} || rk_{1(128)} || \ldots || rk_{r-1(128)}$$

is the subkey stream produced by the key schedule algorithm from the key $k_{(\ell)}$.

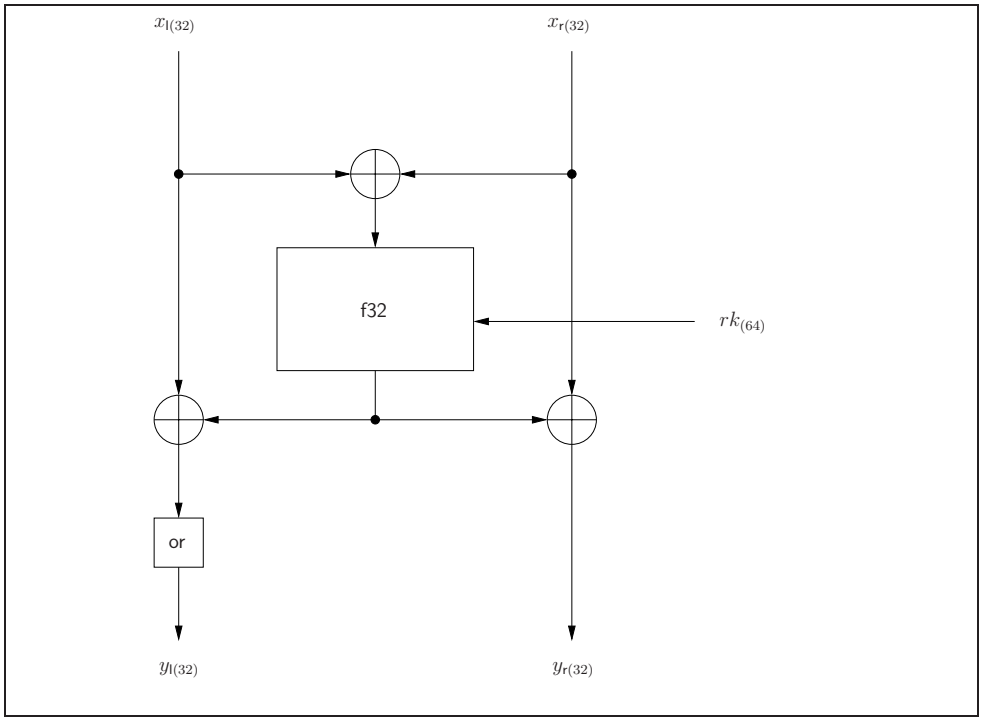

Figure 4.13: lmor64 Round Function

FOX128 **Decryption** The decryption $p_{(128)}$ by FOX128/k/r of a 128-bit ciphertext $c_{(128)}$ is defined as

$$p_{(128)} = \mathsf{elmid128}(\mathsf{elmio128}(\ldots(\mathsf{elmio128}(C_{(128)}, rk_{r-1(128)}), \ldots, rk_{1(128)}), rk_{0(128)})$$

where

$$rk_{(r\cdot 128)} = rk_{0(128)}||rk_{1(128)}||\ldots||rk_{r-1(128)}$$

is the subkey stream produced by the key schedule algorithm from the key $k_{(\ell)}$, as for the encryption operation.

Internal Functions

In this part, we describe formally all the functions used internally in the core of both algorithms FOX64/k/r and FOX128/k/r.

Definitions of lmor64, lmid64, lmio64 In the 64-bit version of the algorithm, one uses three slightly different round functions. The first one, lmor64, illustrated in Fig. 4.13, is built as a Lai-Massey scheme combined with an orthomorphism[8] or. This function transforms a 64-bit input $x_{(64)}$

[8] An orthomorphism o on a group $(\mathcal{G}, +)$ is a permutation $x \mapsto \mathsf{o}(x)$ on $\mathcal{G}$ such that $x \mapsto \mathsf{o}(x) - x$ is also a permutation.

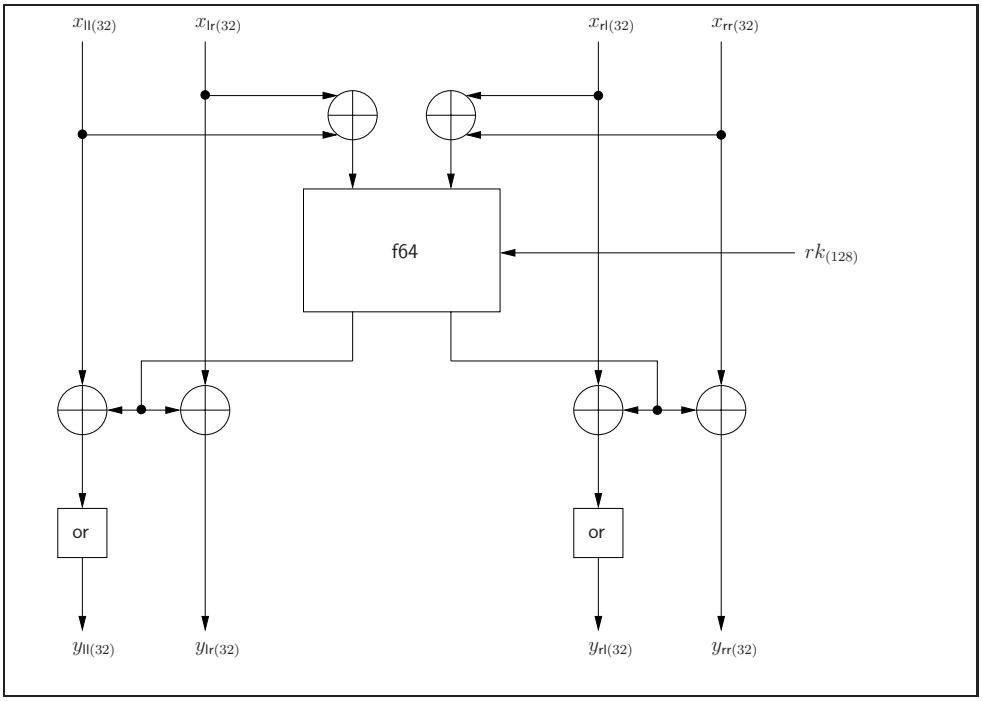

Figure 4.14: Round function elmor128

split in two parts $x_{(64)} = x_{\mathsf{l}(32)}||x_{\mathsf{r}(32)}$ and a 64-bit round key $rk_{(64)}$ in a 64-bit output $y_{(64)} = y_{\mathsf{l}(32)}||y_{\mathsf{r}(32)}$ as follows:

$$\begin{aligned} y_{(64)} &= y_{\mathsf{l}(32)}||y_{\mathsf{r}(32)} = \mathsf{lmor64}\left(x_{\mathsf{r}(32)}||x_{\mathsf{r}(32)}\right) \\ &= \mathsf{or}\left(x_{\mathsf{l}(32)} \oplus \mathsf{f32}\left(x_{\mathsf{l}(32)} \oplus x_{\mathsf{r}(32)}, rk_{(64)}\right)\right)\,||\, \\ &\quad \left(x_{\mathsf{r}(32)} \oplus \mathsf{f32}\left(x_{\mathsf{l}(32)} \oplus x_{\mathsf{r}(32)}, rk_{(64)}\right)\right) \end{aligned}$$

The lmid64 function is a slightly modified version of lmor64, namely it is the same one without the orthomorphism or:

$$\begin{aligned} y_{(64)} &= y_{\mathsf{l}(32)}||y_{\mathsf{r}(32)} = \mathsf{lmid64}\left(x_{\mathsf{l}(32)}||x_{\mathsf{r}(32)}\right) \\ &= \left(x_{\mathsf{l}(32)} \oplus \mathsf{f32}\left(x_{\mathsf{l}(32)} \oplus x_{\mathsf{r}(32)}, rk_{(64)}\right)\right)\,||\, \\ &\quad \left(x_{\mathsf{r}(32)} \oplus \mathsf{f32}\left(x_{\mathsf{l}(32)} \oplus x_{\mathsf{r}(32)}, rk_{(64)}\right)\right) \end{aligned}$$

Finally, lmio64 is defined by

$$\begin{aligned} y_{(64)} &= y_{\mathsf{l}(32)}||y_{\mathsf{r}(32)} = \mathsf{lmio64}\left(x_{\mathsf{l}(32)}||x_{\mathsf{r}(32)}\right) \\ &= \mathsf{io}\left(x_{\mathsf{l}(32)} \oplus \mathsf{f32}\left(x_{\mathsf{l}(32)} \oplus x_{\mathsf{r}(32)}, rk_{(64)}\right)\right)\,||\, \\ &\quad \left(x_{\mathsf{r}(32)} \oplus \mathsf{f32}\left(x_{\mathsf{l}(32)} \oplus x_{\mathsf{r}(32)}, rk_{(64)}\right)\right) \end{aligned}$$

where io is the inverse of the orthormorphism or.

Definitions of elmor128**,** elmid128**,** elmio128 In the 128-bit version of the algorithm, one uses three slightly different round functions, as in the 64-bit version. The first one, elmor128, illustrated in Fig. 4.14, is built as an *Extended Lai-Massey scheme* combined with two orthomorphisms or. This function transforms a 128-bit input $x_{(128)}$ split in four parts $x_{(128)} = x_{\mathsf{ll}(32)}||x_{\mathsf{lr}(32)}||x_{\mathsf{rl}(32)}||x_{\mathsf{rr}(32)}$ and a 128-bit round key $rk_{(128)}$ in a 128-bit output $y_{(128)} = y_{\mathsf{ll}(32)}||y_{\mathsf{lr}(32)}||y_{\mathsf{rl}(32)}||y_{\mathsf{rr}(32)}$ as follows:

$$\begin{aligned} y_{(128)} &= y_{\mathsf{ll}(32)}||y_{\mathsf{lr}(32)}||y_{\mathsf{rl}(32)}||y_{\mathsf{rr}(32)} = \mathsf{elmor128}\left(x_{\mathsf{ll}(32)}||x_{\mathsf{lr}(32)}||x_{\mathsf{rl}(32)}||x_{\mathsf{rr}(32)}\right) \\ &= \mathsf{or}\left(x_{\mathsf{ll}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{l}(32)}\right)\Big|\Big| \\ &\quad \left(x_{\mathsf{lr}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{l}(32)}\right)\Big|\Big| \\ &\quad \mathsf{or}\left(x_{\mathsf{rl}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{r}(32)}\right)\Big|\Big| \\ &\quad \left(x_{\mathsf{rr}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{r}(32)}\right) \end{aligned}$$

The elmid128 function is a slightly modified version of elmor128, namely it is the same one without the orthomorphism or:

$$\begin{aligned} y_{(128)} &= y_{\mathsf{ll}(32)}||y_{\mathsf{lr}(32)}||y_{\mathsf{rl}(32)}||y_{\mathsf{rr}(32)} = \mathsf{elmid128}\left(x_{\mathsf{ll}(32)}||x_{\mathsf{lr}(32)}||x_{\mathsf{rl}(32)}||x_{\mathsf{rr}(32)}\right) \\ &= \left(x_{\mathsf{ll}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{l}(32)}\right)\Big|\Big| \\ &\quad \left(x_{\mathsf{lr}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{l}(32)}\right)\Big|\Big| \\ &\quad \left(x_{\mathsf{rl}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{r}(32)}\right)\Big|\Big| \\ &\quad \left(x_{\mathsf{rr}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{r}(32)}\right) \end{aligned}$$

Finally, elmio128 is defined by

$$\begin{aligned} y_{(128)} &= y_{\mathsf{ll}(32)}||y_{\mathsf{lr}(32)}||y_{\mathsf{rl}(32)}||y_{\mathsf{rr}(32)} = \mathsf{elmio128}\left(x_{\mathsf{ll}(32)}||x_{\mathsf{lr}(32)}||x_{\mathsf{rl}(32)}||x_{\mathsf{rr}(32)}\right) \\ &= \mathsf{io}\left(x_{\mathsf{ll}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{l}(32)}\right)\Big|\Big| \\ &\quad \left(x_{\mathsf{lr}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{l}(32)}\right)\Big|\Big| \\ &\quad \mathsf{io}\left(x_{\mathsf{rl}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{r}(32)}\right)\Big|\Big| \\ &\quad \left(x_{\mathsf{rr}(32)} \oplus \mathsf{f64}\left((x_{\mathsf{ll}(32)} \oplus x_{\mathsf{lr}(32)})||(x_{\mathsf{rl}(32)} \oplus x_{\mathsf{rr}(32)}), rk_{(128)}\right)_{\mathsf{r}(32)}\right) \end{aligned}$$

Definitions of or **and** io The orthomorphism or is a function taking a 32-bit input $x_{(32)} = x_{\mathsf{l}(16)}||x_{\mathsf{r}(16)}$ and returning a 32-bit output $y_{(32)} = y_{\mathsf{l}(16)}||y_{\mathsf{r}(16)}$. It is defined as

$$y_{\mathsf{l}(16)}||y_{\mathsf{r}(16)} = \mathsf{or}\left(x_{\mathsf{l}(16)}||x_{\mathsf{r}(16)}\right) = x_{\mathsf{r}(16)}||\left(x_{\mathsf{l}(16)} \oplus x_{\mathsf{r}(16)}\right)$$

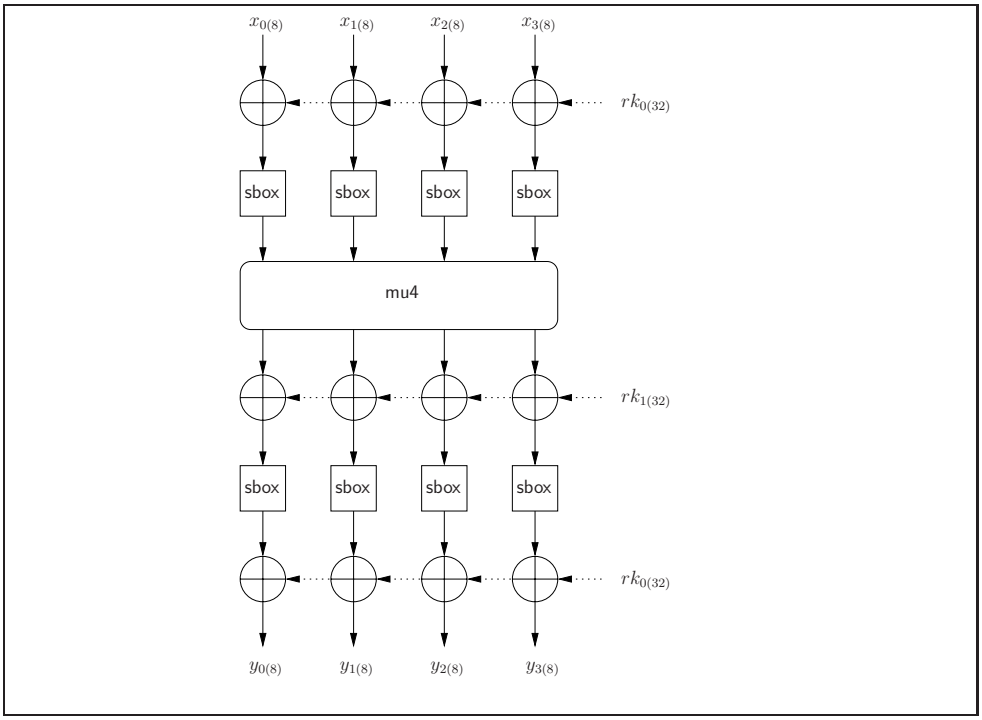

Figure 4.15: Function f32

or is in fact a one-round Feistel scheme with the identity function as round function. The inverse function of or, denoted io, is defined as

$$y_{\mathsf{l}(16)}||y_{\mathsf{r}(16)} = \mathsf{io}\left(x_{\mathsf{l}(32)}||x_{\mathsf{r}(32)}\right) = \left(x_{\mathsf{l}(16)} \oplus x_{\mathsf{r}(16)}\right)||x_{\mathsf{l}(16)}$$

Definition of f32 The function f32 builds the core of FOX64/k/r. It is built of three main parts: a substitution part, denoted sigma4, a diffusion part, denoted mu4, and a round key addition part (see Fig. 4.15). Formally, the f32 function takes a 32-bit input $x_{(32)}$, a 64-bit round key $rk_{(64)} = rk_{0(32)}||rk_{1(32)}$ and returns a 32-bit output $y_{(32)}$. The f32 function is then formally defined as

$$\begin{aligned} y_{(32)} &= \mathsf{f32}\left(x_{(32)}, rk_{(64)}\right) \\ &= \mathsf{sigma4}(\mathsf{mu4}(\mathsf{sigma4}(x_{(32)} \oplus rk_{0(32)})) \oplus rk_{1(32)}) \oplus rk_{0(32)} \end{aligned}$$

Definition of f64 The function f64 builds the core of FOX128/k/r. It is built of three main parts: a substitution part, denoted sigma8, a diffusion part, denoted mu8, and a round key addition part (see Fig. 4.16). Formally, the f64 function takes a 64-bit input $x_{(64)}$, a 128-bit round key $rk_{(128)} = rk_{0(64)}||rk_{1(64)}$ and returns a 64-bit output $y_{(64)}$. The f64 function is then

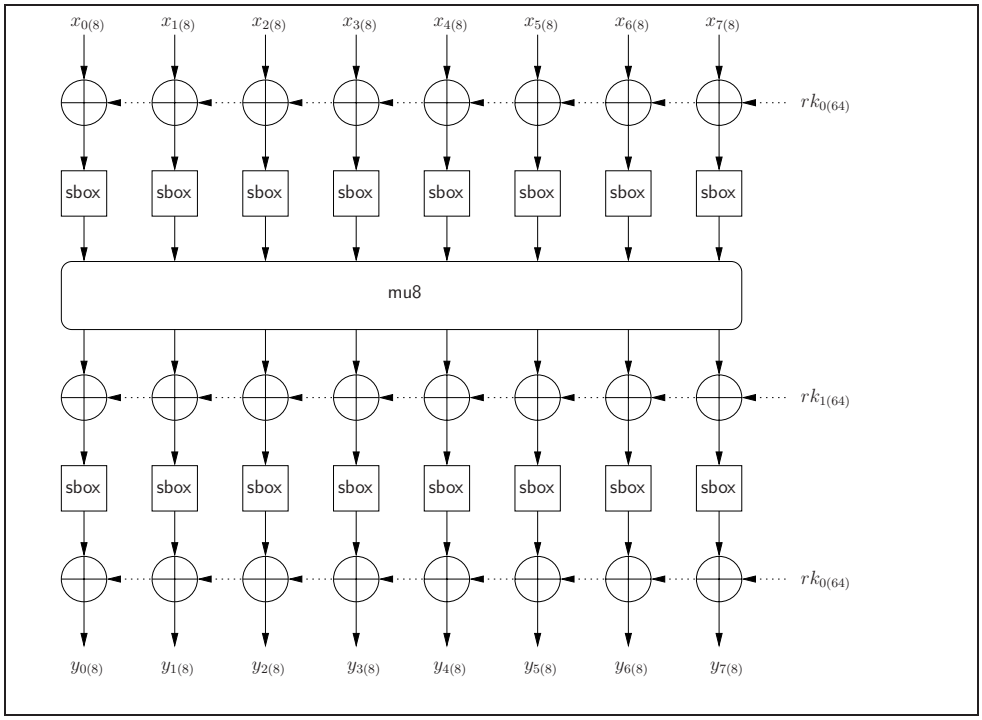

Figure 4.16: Function f64

defined as

$$\begin{aligned} y_{(64)} &= \mathsf{f64}\left(x_{(64)}, rk_{(128)}\right) \\ &= \mathsf{sigma8}(\mathsf{mu8}(\mathsf{sigma8}(x_{(64)} \oplus rk_{0(64)})) \oplus rk_{1(64)}) \oplus rk_{0(64)} \end{aligned}$$

Definition of sigma4, sigma8 **and** sbox The function sigma4 takes a 32-bit input $x_{(32)} = x_{0(8)}||x_{1(8)}||x_{2(8)}||x_{3(8)}$ and returns a 32-bit output $y_{(32)}$. It is defined as

$$\begin{aligned} y_{(32)} &= \mathsf{sigma4}\left(x_{0(8)}||x_{1(8)}||x_{2(8)}||x_{3(8)}\right) \\ &= \mathsf{sbox}(x_{0(8)})||\mathsf{sbox}(x_{1(8)})||\mathsf{sbox}(x_{2(8)})||\mathsf{sbox}(x_{3(8)}) \end{aligned}$$

The function sigma8 takes a 64-bit input

$$x_{(64)} = x_{0(8)}||x_{1(8)}||x_{2(8)}||x_{3(8)}||x_{4(8)}||x_{5(8)}||x_{6(8)}||x_{7(8)}$$

and returns a 64-bit output $y_{(64)}$. It is defined as

$$\begin{aligned} y_{(64)} &= \mathsf{sigma8}\left(x_{0(8)}||x_{1(8)}||x_{2(8)}||x_{3(8)}||x_{4(8)}||x_{5(8)}||x_{6(8)}||x_{7(8)}\right) \\ &= \mathsf{sbox}(x_{0(8)})||\mathsf{sbox}(x_{1(8)})||\mathsf{sbox}(x_{2(8)})||\mathsf{sbox}(x_{3(8)})|| \\ &\quad \mathsf{sbox}(x_{4(8)})||\mathsf{sbox}(x_{5(8)})||\mathsf{sbox}(x_{6(8)})||\mathsf{sbox}(x_{7(8)}) \end{aligned}$$

Finally, the sbox function is the lookup-up table defined in Fig. 4.17. We

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	5D	DE	00	B7	D3	CA	3C	0D	C3	F8	CB	8D	76	89	AA	12
1.	88	22	4F	DB	6D	47	E4	4C	78	9A	49	93	C4	C0	86	13
2.	A9	20	53	1C	4E	CF	35	39	B4	A1	54	64	03	C7	85	5C
3.	5B	CD	D8	72	96	42	B8	E1	A2	60	EF	BD	02	AF	8C	73
4.	7C	7F	5E	F9	65	E6	EB	AD	5A	A5	79	8E	15	30	EC	A4
5.	C2	3E	E0	74	51	FB	2D	6E	94	4D	55	34	AE	52	7E	9D
6.	4A	F7	80	F0	D0	90	A7	E8	9F	50	D5	D1	98	CC	A0	17
7.	F4	B6	C1	28	5F	26	01	AB	25	38	82	7D	48	FC	1B	CE
8.	3F	6B	E2	67	66	43	59	19	84	3D	F5	2F	C9	BC	D9	95
9.	29	41	DA	1A	B0	E9	69	D2	7B	D7	11	9B	33	8A	23	09
A.	D4	71	44	68	6F	F2	0E	DF	87	DC	83	18	6A	EE	99	81
B.	62	36	2E	7A	FE	45	9C	75	91	0C	0F	E7	F6	14	63	1D
C.	0B	8B	B3	F3	B2	3B	08	4B	10	A6	32	B9	A8	92	F1	56
D.	DD	21	BF	04	BE	D6	FD	77	EA	3A	C8	8F	57	1E	FA	2B
E.	58	C5	27	AC	E3	ED	97	BB	46	05	40	31	E5	37	2C	9E
F.	0A	B1	B5	06	6C	1F	A3	2A	70	FF	BA	07	24	16	C6	61

Figure 4.17: Mapping sbox

read this table as follows: to compute sbox(4C), one selects first the row named 4. (*i.e.* the fifth row), and then one selects the column named .C (*i.e.* the thirteenth column) and we get finally

$$\mathsf{sbox}(\mathtt{4C}) = 15$$

Definition of mu4 The diffusive part of f32 is a linear $(4,4)$-multipermutation defined on $\mathrm{GF}(2^8)$. Formally, it is a function taking a 32-bit input

$$x_{(32)} = x_{0(8)}||x_{1(8)}||x_{2(8)}||x_{3(8)}$$

and returning a 32-bit output

$$y_{(32)} = y_{0(8)}||y_{1(8)}||y_{2(8)}||y_{3(8)}$$

and defined by

$$\begin{pmatrix} y_{0(8)} \\ y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \alpha \\ 1 & c & \alpha & 1 \\ c & \alpha & 1 & 1 \\ \alpha & 1 & c & 1 \end{pmatrix} \times \begin{pmatrix} x_{0(8)} \\ x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \end{pmatrix}$$

where

$$c = \alpha^{-1} + 1 = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + 1$$

All the additions and multiplications are defined in $\mathrm{GF}(2^8)$ using the representation described in §4.3.2.

Definition of mu8 The diffusive part of f64 is a linear $(8, 8)$-multipermutation defined on $\mathrm{GF}(2^8)$. Formally, it is a function taking a 64-bit input

$$x_{(64)} = x_{0(8)}||x_{1(8)}||x_{2(8)}||x_{3(8)}||x_{4(8)}||x_{5(8)}||x_{6(8)}||x_{7(8)}$$

and returning a 64-bit output

$$y_{(64)} = y_{0(8)}||y_{1(8)}||y_{2(8)}||y_{3(8)}||y_{4(8)}||y_{5(8)}||y_{6(8)}||y_{7(8)}$$

f64 is defined as

$$\begin{pmatrix} y_{0(8)} \\ y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \\ y_{4(8)} \\ y_{5(8)} \\ y_{6(8)} \\ y_{7(8)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & a \\ 1 & a & b & c & d & e & f & 1 \\ a & b & c & d & e & f & 1 & 1 \\ b & c & d & e & f & 1 & a & 1 \\ c & d & e & f & 1 & a & b & 1 \\ d & e & f & 1 & a & b & c & 1 \\ e & f & 1 & a & b & c & d & 1 \\ f & 1 & a & b & c & d & e & 1 \end{pmatrix} \times \begin{pmatrix} x_{0(8)} \\ x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \\ x_{4(8)} \\ x_{5(8)} \\ x_{6(8)} \\ x_{7(8)} \end{pmatrix}$$

where

$$\begin{aligned} a &= \alpha + 1 \\ b &= \alpha^{-1} + \alpha^{-2} = \alpha^7 + \alpha \\ c &= \alpha \\ d &= \alpha^2 \\ e &= \alpha^{-1} = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 \\ f &= \alpha^{-2} = \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha \end{aligned}$$

All the additions and multiplications are defined in $\mathrm{GF}(2^8)$ using the representation described in §4.3.2.

Key-Schedule Algorithms

The key schedule is the algorithm which derives the subkey material

$$rk_{(r\cdot 64)} = rk_{0(64)}||rk_{1(64)}||\ldots||rk_{r-1(64)}$$

and

$$rk_{(r\cdot 128)} = rk_{0(128)}||rk_{1(128)}||\ldots||rk_{r-1(128)}$$

(for FOX64 and FOX128, respectively) from the key $k_{(\ell)}$.

Design	Block size	Key size	Key-Schedule Version	ek
FOX64	64	$0 \leq \ell \leq 128$	KS64	128
FOX64	64	$136 \leq \ell \leq 256$	KS64h	256
FOX128	128	$0 \leq \ell \leq 256$	KS128	256

Figure 4.18: Key-Schedule Algorithms Characteristics

General Overview A FOX key $k_{(\ell)}$ must have a bit-length ℓ such that $0 \leq \ell \leq 256$, and ℓ must be a multiple of 8. Depending on the key length and the block size, a member of the FOX block cipher family may use one among three different key-schedule algorithm versions, denoted respectively KS64, KS64h and KS128. A constant, ek, depends on these values as well. The table in Fig. 4.18 defines precisely the relation between the key size, the block size, the constant ek and the key-schedule algorithm version.

The three different versions of the key-schedule algorithm are constituted of four main parts: a padding part, denoted P, expanding $k_{(\ell)}$ into ek bits, a mixing part, denoted M, a diversification part, denoted D, whose core consists mainly in a linear feedback shift register denoted LFSR, and finally, a non-linear part, denoted NLx (see Fig. 4.19 and Alg. 4.1 for a high-level overview of the key-schedule algorithm design). As outlined above, the key-schedule algorithm definition depends on a the number of rounds r, on the key length ℓ and on the cipher (FOX64 or FOX128). In fact, NLx is the only part which differs between the different versions, and we will denote the three variants NL64, NL64h and NL128.

```
/* Preprocessing */
pkey ← P(k)
mkey ← M(pkey)
/* Initialization of the loop */
i ← 1
/* Loop */
while i ≤ r do
   dkey ← D(mkey, i, r)
   Output rk_{i−1(x)} ← NLx(dkey)
   i ← i + 1
end while
```

Algorithm 4.1: Key-Schedule Algorithm (High-Level Description)

Definition of KS64 This key-schedule algorithm is designed to be used by FOX64 with keys smaller or equal to 128 bits. It takes the following parameters as input: a key k of length ℓ bits, with $0 \leq \ell \leq 128$ and a number of rounds r. It returns in output r 64-bit subkeys. KS64 is formally

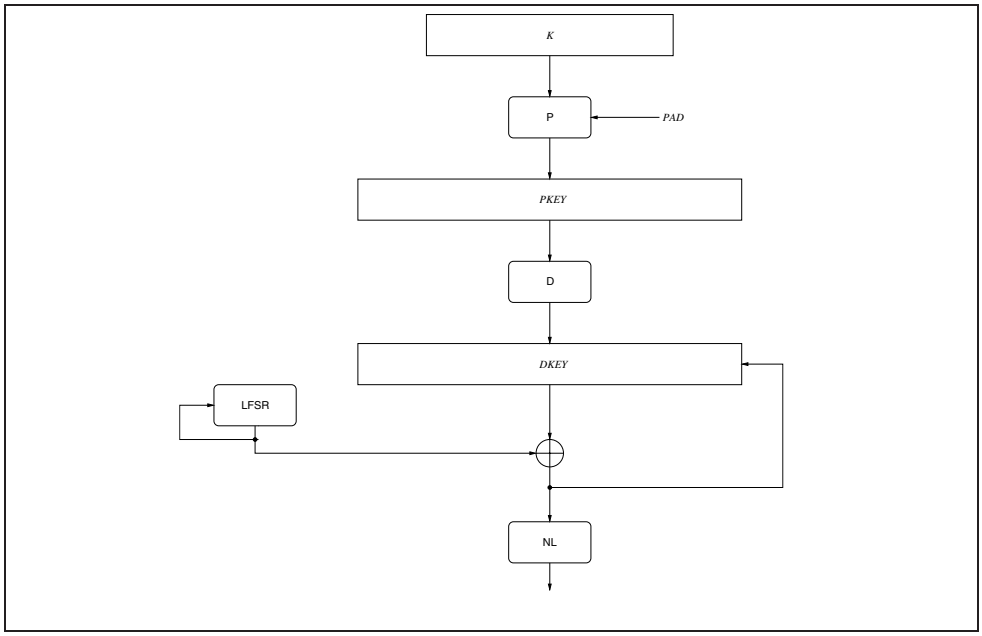

Figure 4.19: Key-Schedule Algorithm (High-Level Overview)

defined in Alg. 4.2.

Definition of KS64h This key schedule algorithm is designed to be used by FOX64 with keys larger than 128 bits. It takes the following parameters as input: a key k of length ℓ bits, with $136 \leq \ell \leq 256$ and a number of rounds r. It returns in output r 64-bit subkeys. KS64h is formally defined in Alg. 4.3.

Definition of KS128 This key schedule algorithm is designed to be used by FOX128. It takes the following parameters as input: a key k of length ℓ bits, with $0 \leq \ell \leq 256$ and a number of rounds r. It returns in output r 128-bit subkeys. KS128 is formally defined in Alg. 4.4.

Definition of P The P-part, taking ek and ℓ as input, is basically a function expanding a bit string by $\frac{ek-\ell}{8}$ bytes. More precisely, then P concatenates the input key k with the first $ek - \ell$ bits of the constant pad, giving $pkey$ as output. The P function is defined formally in Alg. 4.5. The pad constant value is defined in the following section.

Definition of pad The constant pad is defined as being the first 256 bits of the hexadecimal development of $e - 2$:

$$e - 2 = \sum_{n=0}^{+\infty} \frac{1}{n!} - 2$$

```
/* Preprocessing */
if ℓ < ek then
  pkey = P(k)
  mkey = M(pkey)
else
  pkey = k
  mkey = pkey
end if
/* Initialization of the loop */
i = 1
/* Loop */
while i ≤ r do
  dkey = D(mkey, i, r)
  Output rk_{i−1(64)} = NL64(dkey)
  i = i + 1
end while
```

Algorithm 4.2: Key-Schedule Algorithm KS64

```
/* Preprocessing */
if ℓ < ek then
  pkey = P(k)
  mkey = M(pkey)
else
  pkey = k
  mkey = pkey
end if
/* Initialization of the loop */
i = 1
/* Loop */
while i ≤ r do
  dkey = D(mkey, i, r)
  Output rk_{i−1(64)} = NL64h(dkey)
  i = i + 1
end while
```

Algorithm 4.3: Key-Schedule Algorithm KS64h

```
/* Preprocessing */
if ℓ < ek then
   pkey = P(k)
   mkey = M(pkey)
else
   pkey = k
   mkey = pkey
end if
/* Initialization of the loop */
i = 1
/* Loop */
while i ≤ r do
   dkey = D(mkey, i, r)
   Output rk_{i−1(128)} = NL128(dkey)
   i = i + 1
end while
```

Algorithm 4.4: Key-Schedule Algorithm KS128

```
Output pkey = k||pad_{[0...ek−ℓ−1]}
```

Algorithm 4.5: P-Part

Thus, it is the concatenation of the four following 64-bit constants

$$\begin{aligned} \mathsf{pad} \;=\; & \texttt{0xB7E151628AED2A6A} \;\;\| \\ & \texttt{0xBF7158809CF4F3C7} \;\;\| \\ & \texttt{0x62E7160F38B4DA56} \;\;\| \\ & \texttt{0xA784D9045190CFEF} \end{aligned}$$

Definition of M The M-part is used to mix the padded key $pkey$, such that the constant words are mixed uo by using the randomness provided by the key. This is done with help of a Fibonacci recursion. It takes as input a key $pkey$ with length ek (expressed in bits). More formally, the padded key $pkey$ is seen as an array of $\frac{ek}{8}$ bytes $pkey_{i(8)}, 0 \leq i \leq \frac{ek}{8} - 1$, and is mixed according to

$$mkey_{i(8)} = pkey_{i(8)} \oplus \left(mkey_{i-1(8)} + mkey_{i-2(8)} \bmod 2^8\right) \quad 0 \leq i \leq \frac{ek}{8} - 1$$

with the convention that

$$\text{mkey}_{-2(8)} = \texttt{0x6A} \;\text{ and }\; \text{mkey}_{-1(8)} = \texttt{0x76}$$

Note here that $+$ denotes the addition performed modulo 2^8 while $\oplus$ denotes the addition in GF $\left(2^8\right)$, which is actually a XOR operation.

Definition of D The D-part is a diversification part. It takes a key $mkey$ having a length in bits equal to ek, the total round number r, and the current round number i, with $1 \leq i \leq r$; it modifies $mkey$ with help of the output of a 24-bit Linear Shift Feedback Register (LFSR) denoted LFSR. More precisely, $mkey$ is seen as an array of $\left\lfloor \frac{ek}{24} \right\rfloor$ 24-bit values $mkey_{j(24)}$, with $0 \leq j \leq \left\lfloor \frac{ek}{24} \right\rfloor - 1$ concatenated with one residue byte $mkeyrb_{(8)}$ (if $ek = 128$) or two residue bytes $mkeyrb_{(16)}$ (if $ek = 256$), and is modified according to

$$dkey_{j(24)} \quad = \quad mkey_{j(24)} \oplus \mathsf{LFSR}\left((i-1) \cdot \left\lceil \frac{ek}{24} \right\rceil + j, r\right)$$

for $0 \leq j \leq \left\lfloor \frac{ek}{24} \right\rfloor - 1$; the $dkeyrb_{(8)}$ value ($dkeyrb_{(16)}$) is obtained by XORing the most 8 (16) significant bits of $\mathsf{LFSR}((i-1) \cdot \left\lceil \frac{ek}{24} \right\rceil + \left\lfloor \frac{ek}{24} \right\rfloor, r)$ with $mkeyrb_{(8)}$ ($mkeyrb_{(16)}$), respectively. The remaining 16 (8) bits of the LFSR routine output are discarded.

Definition of LFSR The diversification part D needs a stream of pseudo-random values; it is produced by a 24-bit linear feedback shift register, denoted LFSR. This algorithm takes two inputs, the total number of rounds r and a number of preliminary clocking c. It is based on the following primitive polynomial of degree 24 over GF(2).

Definition 4.3.3 (Primitive Polynomial $\mathsf{PKS}(\xi)$**).** *The polynomial representing* $\mathrm{GF}\left(2^{24}\right)$ *in the FOX block cipher family is the primitive polynomial over* $\mathrm{GF}\,(2)$ *defined by*

$$\mathsf{PKS}(\xi) = \xi^{24} + \xi^4 + \xi^3 + \xi + 1$$

The register is initialized with the value $\mathtt{0x6A}||r_{(8)}||\overline{r_{(8)}}$, where $r_{(8)}$ is expressed as an 8-bit value, and $\overline{r_{(8)}}$ is its bitwise complemented version (i.e. $r_{(8)} = \overline{r_{(8)}} \oplus \mathtt{0xFF}$). LSFR is described formally in Alg. 4.6.

Definition of NL64 The NL64-part takes a single input: the 128-bit value $dkey$ corresponding to the current round. The $dkey$ value passes through a substitution layer (made of four parallel sigma4 functions), a diffusion layer (made of four parallel mu4 functions) and a mixing layer called mix64. Then, the constant $\mathsf{pad}_{[0...127]}$ is XORed and the result is flipped if and only if $k = ek$. The result passes through a second substitution layer, it is hashed down to 64 bits and the resulting value is encrypted first with a lmor64 round function, where the subkey is the left half of the $dkey$ value and second by a lmid64 function, where the subkey is the right half of $dkey$. The resulting value is defined to be the 64-bit round key. Fig. 4.20 illustrates the NL64 process and Alg. 4.7 describes it formally.

/* Initialization */
$reg = \texttt{0x6A}||r||\overline{r}$
/* Pre-Clocking */
$p = 0$
while $p < c$ **do**
  $p = p + 1$
  if (reg AND 0x800000) $\neq$ 0x000000 **then**
    $reg = (reg \ll 1) \oplus \texttt{0x00001B}$
  else
    $reg = (reg \ll 1)$
  end if
end while
Output reg

Algorithm 4.6: LFSR Algorithm

$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}$=$dkey$
$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}$=sigma4($t_{0(32)}$)||sigma4($t_{1(32)}$)||sigma4($t_{2(32)}$)||sigma4($t_{3(32)}$)
$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}$=mu4($t_{0(32)}$)||mu4($t_{1(32)}$)||mu4($t_{2(32)}$)||mu4($t_{3(32)}$)
$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}$=mix64($t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}$)
$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}$=($t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}$)$\oplus$pad$_{[0..127]}$
if $k = ek$ **then**
  $t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}$=$\overline{t_{0(32)}}||\overline{t_{1(32)}}||\overline{t_{2(32)}}||\overline{t_{3(32)}}$
end if
$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}$=sigma4($t_{0(32)}$)||sigma4($t_{1(32)}$)||sigma4($t_{2(32)}$)||sigma4($t_{3(32)}$)
$t_{0(32)}||t_{1(32)}$=($t_{0(32)} \oplus t_{2(32)}$)||($t_{1(32)} \oplus t_{3(32)}$)
$t_{0(32)}||t_{1(32)}$=lmor64($t_{0(32)}||t_{1(32)}$,$dkey_{[0\ldots63]}$)
$t_{0(32)}||t_{1(32)}$=lmid64($t_{0(32)}||t_{1(32)}$,$dkey_{[64\ldots127]}$)
Output $t_{0(32)}||t_{1(32)}$ as round subkey.

Algorithm 4.7: NL64 Part

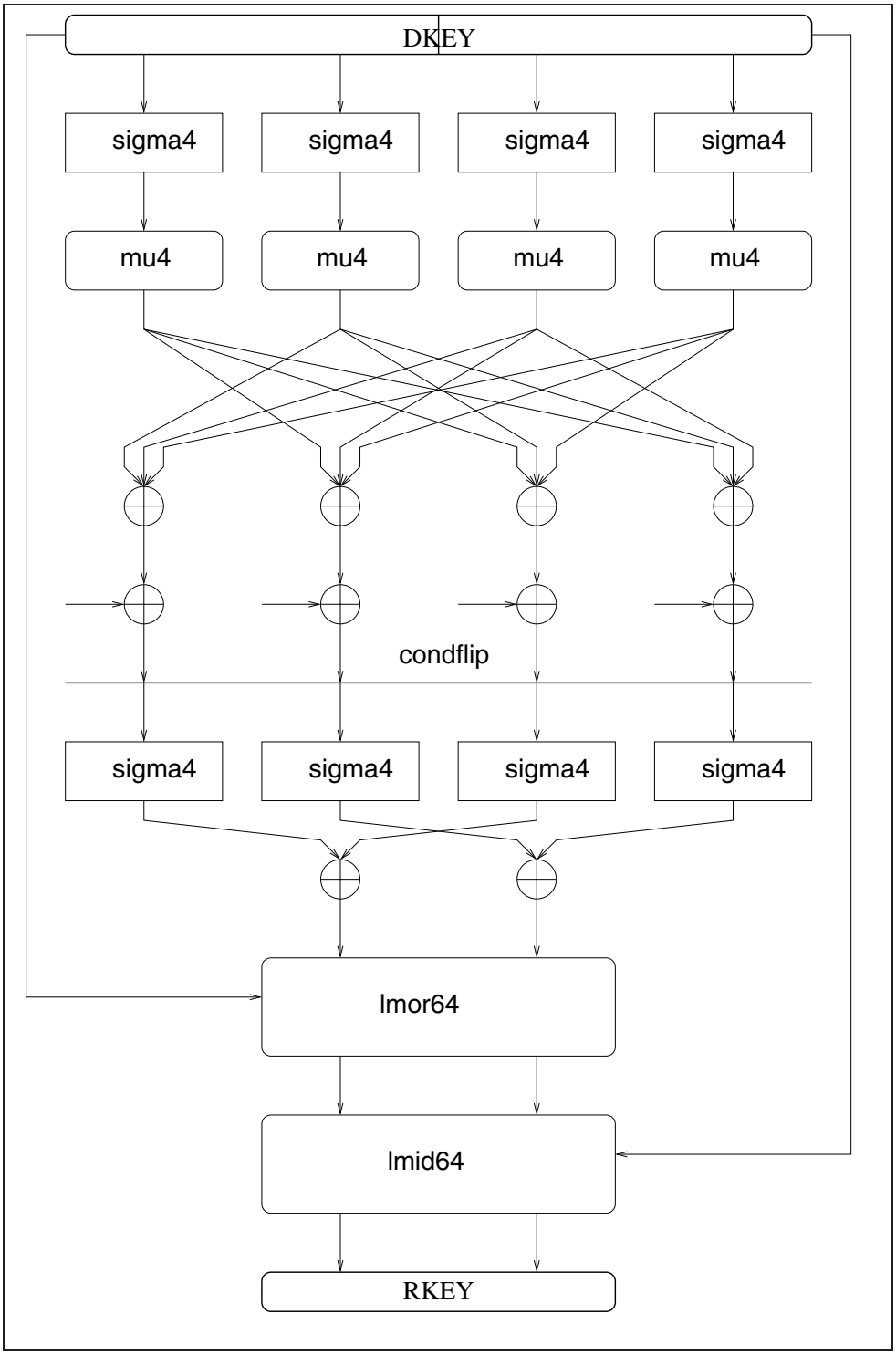

Figure 4.20: NL64 Part

Definition of NL64h The NL64h-part takes a single input: the 256-bit value *dkey* corresponding to the current round. The *dkey* value passes through a substitution layer (made of eight parallel sigma4 functions), a diffusion layer (made of eight parallel mu4 functions) and a mixing layer called mix64h. Then, the constant pad is XORed and the result is flipped if and only if $k = ek$. The result passes through a second substitution layer, it is hashed down to 64 bits and the resulting value is encrypted first with three lmor64 round functions, where the respective subkeys are the three left quarters of the *dkey* value and secondly by a lmid64 function, where the subkey is the rightmost quarter of *dkey*. The resulting value is defined to be the 64-bit round key. Fig. 4.21 illustrates the NL64h process and Alg. 4.8 describes it formally.

/* Initialization */

$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}||t_{4(32)}||t_{5(32)}||t_{6(32)}||t_{7(32)}=dkey$

/* Substitution Layer */

$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}=\mathsf{sigma4}(t_{0(32)})||\mathsf{sigma4}(t_{1(32)})||\mathsf{sigma4}(t_{2(32)})||\mathsf{sigma4}(t_{3(32)})$

$t_{4(32)}||t_{5(32)}||t_{6(32)}||t_{7(32)}=\mathsf{sigma4}(t_{4(32)})||\mathsf{sigma4}(t_{5(32)})||\mathsf{sigma4}(t_{6(32)})||\mathsf{sigma4}(t_{7(32)})$

/* Diffusion Layer */

$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}=\mathsf{mu4}(t_{0(32)})||\mathsf{mu4}(t_{1(32)})||\mathsf{mu4}(t_{2(32)})||\mathsf{mu4}(t_{3(32)})$

$t_{4(32)}||t_{5(32)}||t_{6(32)}||t_{7(32)}=\mathsf{mu4}(t_{4(32)})||\mathsf{mu4}(t_{5(32)})||\mathsf{mu4}(t_{6(32)})||\mathsf{mu4}(t_{7(32)})$

$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}||t_{4(32)}||t_{5(32)}||t_{6(32)}||t_{7(32)}=$

$\mathsf{mix64h}(t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}||t_{4(32)}||t_{5(32)}||t_{6(32)}||t_{7(32)})$

$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}||t_{4(32)}||t_{5(32)}||t_{6(32)}||t_{7(32)}=$

$(t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}||t_{4(32)}||t_{5(32)}||t_{6(32)}||t_{7(32)})\oplus\mathsf{pad}$

if $k = ek$ **then**

$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}||t_{4(32)}||t_{5(32)}||t_{6(32)}||t_{7(32)}=$

$\overline{t_{0(32)}}||\overline{t_{1(32)}}||\overline{t_{2(32)}}||\overline{t_{3(32)}}||\overline{t_{4(32)}}||\overline{t_{5(32)}}||\overline{t_{6(32)}}||\overline{t_{7(32)}}$

end if

/* Substitution Layer */

$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}=\mathsf{sigma4}(t_{0(32)})||\mathsf{sigma4}(t_{1(32)})||\mathsf{sigma4}(t_{2(32)})||\mathsf{sigma4}(t_{3(32)})$

$t_{4(32)}||t_{5(32)}||t_{6(32)}||t_{7(32)}=\mathsf{sigma4}(t_{4(32)})||\mathsf{sigma4}(t_{5(32)})||\mathsf{sigma4}(t_{6(32)})||\mathsf{sigma4}(t_{7(32)})$

/* Hashing Layer */

$t_{0(32)}||t_{1(32)}||t_{2(32)}||t_{3(32)}=(t_{0(32)}\oplus t_{1(32)})||(t_{2(32)}\oplus t_{3(32)})||(t_{4(32)}\oplus t_{5(32)})||(t_{6(32)}\oplus t_{7(32)})$

$t_{0(32)}||t_{1(32)}=(t_{0(32)}\oplus t_{2(32)})||(t_{1(32)}\oplus t_{3(32)})$

/* Encryption Layer */

$t_{0(32)}||t_{1(32)}=\mathsf{lmor64}(t_{0(32)}||t_{1(32)},dkey_{[0\ldots63]})$

$t_{0(32)}||t_{1(32)}=\mathsf{lmor64}(t_{0(32)}||t_{1(32)},dkey_{[64\ldots127]})$

$t_{0(32)}||t_{1(32)}=\mathsf{lmor64}(t_{0(32)}||t_{1(32)},dkey_{[128\ldots191]})$

$t_{0(32)}||t_{1(32)}=\mathsf{lmid64}(t_{0(32)}||t_{1(32)},dkey_{[192\ldots256]})$

Output $t_{0(32)}||t_{1(32)}$ as round subkey.

Algorithm 4.8: NL64h Part

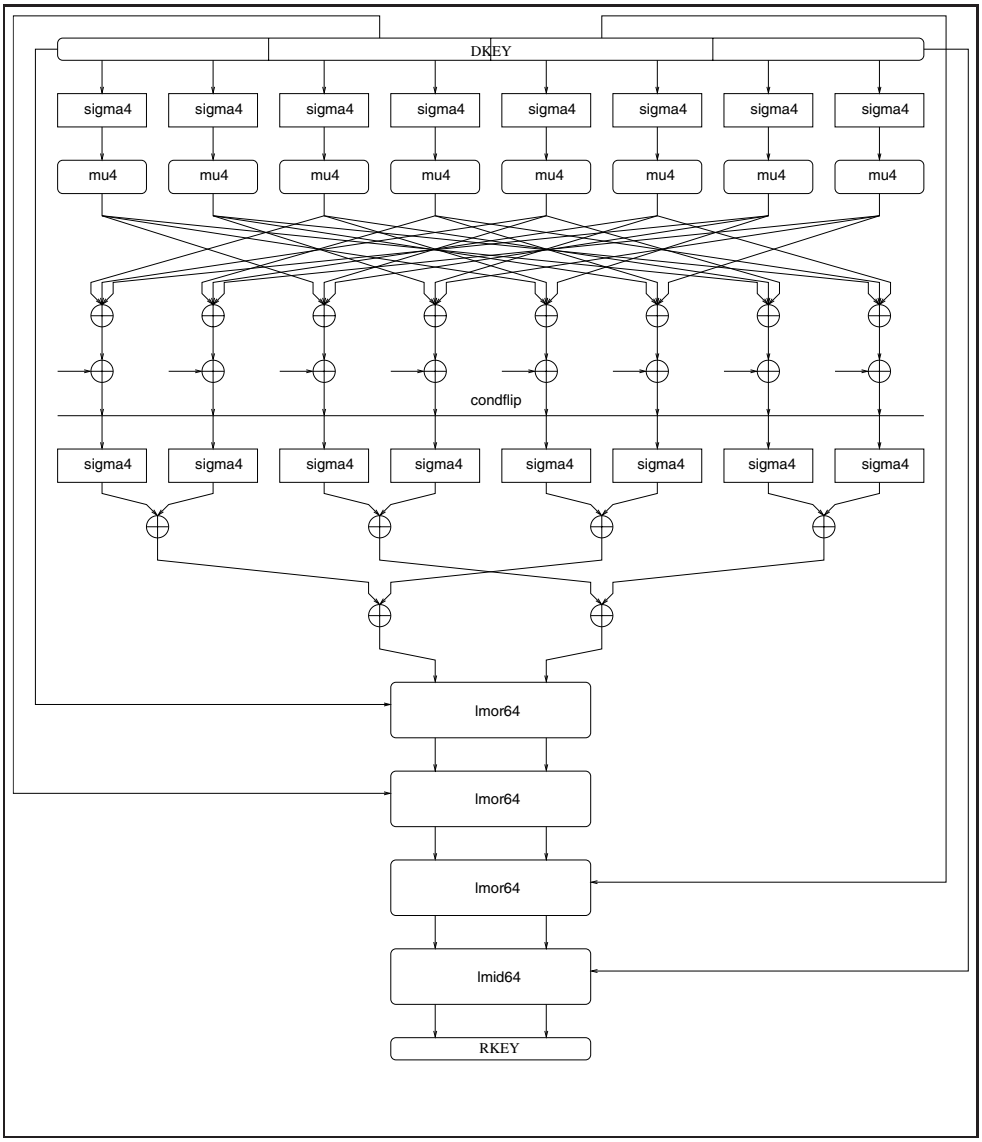

Figure 4.21: NL64h Part

Definition of NL128 The NL128-part takes a single different input: the 256-bit value *dkey* corresponding to the current round. Basically, the *dkey* value passes through a substitution layer (made of four parallel sigma8 functions), a diffusion layer (made of four parallel mu8 functions) and a mixing layer called mix128. Then, the constant pad is XORed and the result is flipped if and only if $k = ek$. The result passes through a second substitution layer, it is hashed down to 128 bits and the resulting value is encrypted first with a elmor128 round function, where the subkey is the left half of the *dkey* value and second by a elmid128 function, where the subkey is the right half of *dkey*. The resulting value is defined to be the 128-bit round key. Fig. 4.22 illustrates the NL128 process and Alg. 4.9 describes it formally.

$t_{0(64)}||t_{1(64)}||t_{2(64)}||t_{3(64)}$=*dkey*
$t_{0(64)}||t_{1(64)}||t_{2(64)}||t_{3(64)}$=sigma8($t_{0(64)}$)||sigma8($t_{1(64)}$)||sigma8($t_{2(64)}$)||sigma8($t_{3(64)}$)
$t_{0(64)}||t_{1(64)}||t_{2(64)}||t_{3(64)}$=mu8($t_{0(64)}$)||mu8($t_{1(64)}$)||mu8($t_{2(64)}$)||mu8($t_{3(64)}$)
$t_{0(64)}||t_{1(64)}||t_{2(64)}||t_{3(64)}$=mix128($t_{0(64)}||t_{1(64)}||t_{2(64)}||t_{3(64)}$)
$t_{0(64)}||t_{1(64)}||t_{2(64)}||t_{3(64)}$=($t_{0(64)}||t_{1(64)}||t_{2(64)}||t_{3(64)}$)⊕pad
if $k = ek$ **then**
  $t_{0(64)}||t_{1(64)}||t_{2(64)}||t_{3(64)}$=$\overline{t_{0(64)}}||\overline{t_{1(64)}}||\overline{t_{2(64)}}||\overline{t_{3(64)}}$
end if
$t_{0(64)}||t_{1(64)}||t_{2(64)}||t_{3(64)}$=sigma8($t_{0(64)}$)||sigma8($t_{1(64)}$)||sigma8($t_{2(64)}$)||sigma8($t_{3(64)}$)
$t_{0(64)}||t_{1(64)}$=($t_{0(64)}\oplus t_{2(64)}$)||($t_{1(64)}\oplus t_{3(64)}$)
$t_{0(64)}||t_{1(64)}$=elmor128($t_{0(64)}||t_{1(64)}$,$dkey_{[128\ldots255]}$)
$t_{0(64)}||t_{1(64)}$=elmid128($t_{0(64)}||t_{1(64)}$,$dkey_{[0\ldots127]}$)
Output $t_{0(64)}||t_{1(64)}$ as round subkey.

Algorithm 4.9: NL128 Part

Definition of mix64 Given an input vector of four 32-bit values, denoted

$$x = x_{0(32)}||x_{1(32)}||x_{2(32)}||x_{3(32)}$$

the mix64 function consists in processing it by the following relations, resulting in an output vector denoted $y = y_{0(32)}||y_{1(32)}||y_{2(32)}||y_{3(32)}$. More formally, mix64 is defined as

$$\begin{aligned} y_{0(32)} &= x_{1(32)} \oplus x_{2(32)} \oplus x_{3(32)} \\ y_{1(32)} &= x_{0(32)} \oplus x_{2(32)} \oplus x_{3(32)} \\ y_{2(32)} &= x_{0(32)} \oplus x_{1(32)} \oplus x_{3(32)} \\ y_{3(32)} &= x_{0(32)} \oplus x_{1(32)} \oplus x_{2(32)} \end{aligned}$$

Definition of mix64h Given an input vector of eight 32-bit values, denoted

$$x = x_{0(32)}||x_{1(32)}||x_{2(32)}||x_{3(32)}||x_{4(32)}||x_{5(32)}||x_{6(32)}||x_{7(32)}$$

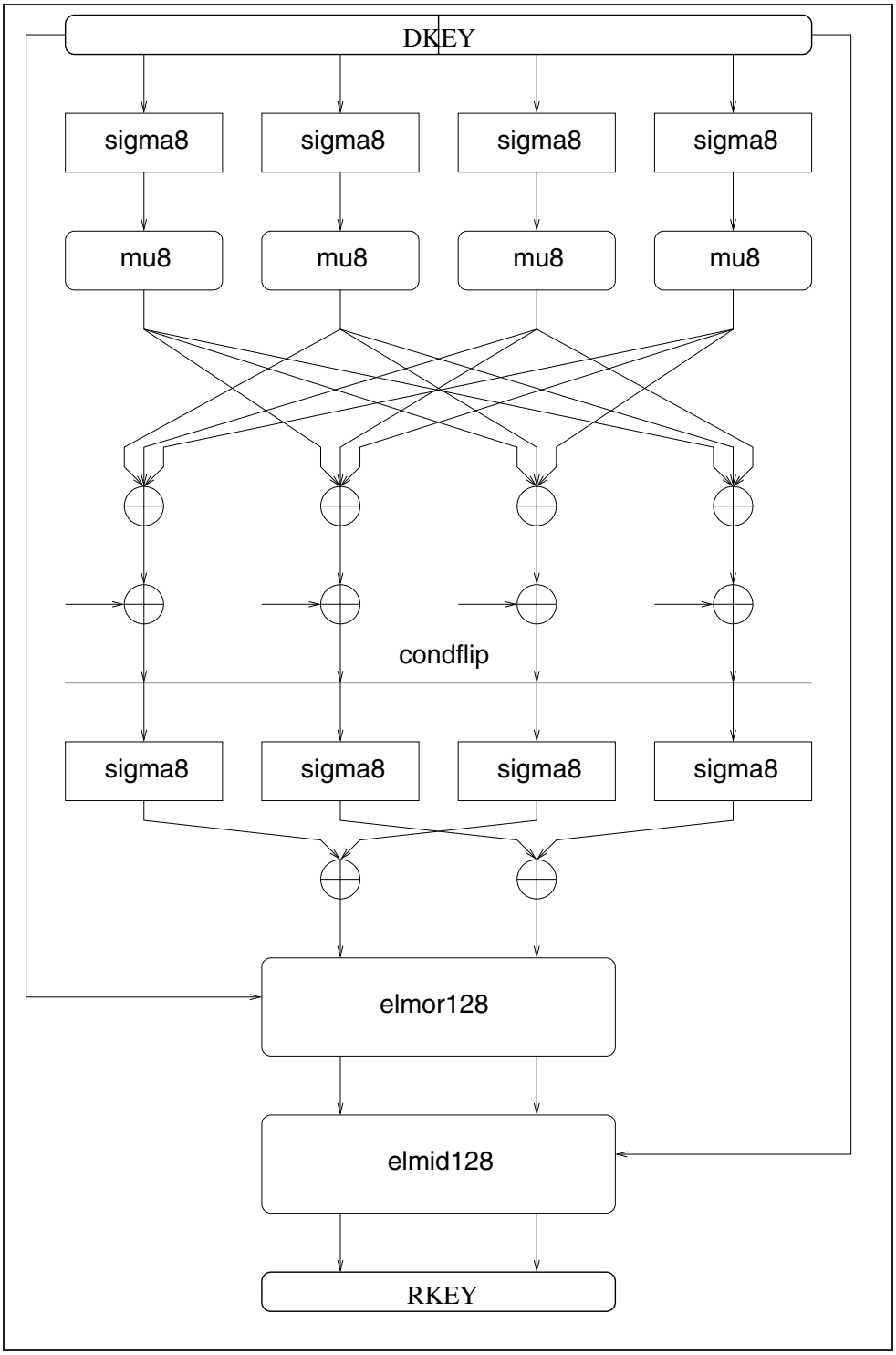

Figure 4.22: NL128 Part

the mix64h function consists in processing it by the following relations, resulting in an output vector denoted

$$y = y_{0(32)}||y_{1(32)}||y_{2(32)}||y_{3(32)}||y_{4(32)}||y_{5(32)}||y_{6(32)}||y_{7(32)}$$

More formally, mix64h is defined as

$$\begin{aligned}
y_{0(32)} &= x_{2(32)} \oplus x_{4(32)} \oplus x_{6(32)} \\
y_{1(32)} &= x_{3(32)} \oplus x_{5(32)} \oplus x_{7(32)} \\
y_{2(32)} &= x_{0(32)} \oplus x_{4(32)} \oplus x_{6(32)} \\
y_{3(32)} &= x_{1(32)} \oplus x_{5(32)} \oplus x_{7(32)} \\
y_{4(32)} &= x_{0(32)} \oplus x_{2(32)} \oplus x_{6(32)} \\
y_{5(32)} &= x_{1(32)} \oplus x_{3(32)} \oplus x_{7(32)} \\
y_{6(32)} &= x_{0(32)} \oplus x_{2(32)} \oplus x_{4(32)} \\
y_{7(32)} &= x_{1(32)} \oplus x_{3(32)} \oplus x_{5(32)}
\end{aligned}$$

Definition of mix128 Given an input vector of four 64-bit values, denoted $x = x_{0(64)}||x_{1(64)}||x_{2(64)}||x_{3(64)}$, the mix64 function consists in processing it by the following relations, resulting in an output vector denoted $y = y_{0(64)}||y_{1(64)}||y_{2(64)}||y_{3(64)}$. More formally, mix128 is defined as

$$\begin{aligned}
y_{0(64)} &= x_{1(64)} \oplus x_{2(64)} \oplus x_{3(64)} \\
y_{1(64)} &= x_{0(64)} \oplus x_{2(64)} \oplus x_{3(64)} \\
y_{2(64)} &= x_{0(64)} \oplus x_{1(64)} \oplus x_{3(64)} \\
y_{3(64)} &= x_{0(64)} \oplus x_{1(64)} \oplus x_{2(64)}
\end{aligned}$$

4.3.3 Rationales

In this part, we describe several rationales about important components building the FOX family of block ciphers.

Non-Linear Mapping sbox

As outlined earlier, our primary goal was to avoid a purely algebraic construction for the S-box[9]; a secondary goal was the possibility to implement it in a very efficient way on hardware using ASIC or FPGA technologies. The sbox function is a non-linear bijective mapping on 8-bit values. It consists of a Lai-Massey scheme with 3 rounds taking three different substitution boxes as round function where the orthormorphism of the third round is omitted; these "small" S-boxes are denoted S_1, S_2 and S_3, and their content is given in

[9] A drawback of this choice is that it makes the study of "scaled-down" versions of the cipher difficult, if not impossible.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x)$	2	5	1	9	E	A	C	8	6	4	7	F	D	B	0	3
$S_2(x)$	B	4	1	F	0	3	E	D	A	8	7	5	C	2	9	6
$S_3(x)$	D	A	B	1	4	3	8	9	5	7	2	C	F	0	6	E

Figure 4.23: The three small S-boxes of FOX.

Fig. 4.23. The orthomorphism or4 used in the Lai-Massey scheme is a single round of a 4-bit Feistel scheme with the identity function as round function. We describe now the generation process of the sbox transformation. First a set of three different candidates for small substitution boxes, each having a $\mathrm{LP}_{\max}$ and a $\mathrm{DP}_{\max}$ (as defined in §2.3.3) smaller than 2^{-2} were pseudo-randomly chosen. Then, the candidate sbox mapping was evaluated and tested regarding its $\mathrm{LP}_{\max}$ and $\mathrm{DP}_{\max}$ values until a good candidate was found. The chosen sbox satisfies $\mathrm{DP}_{\max}^{\mathsf{sbox}} = \mathrm{LP}_{\max}^{\mathsf{sbox}} = 2^{-4}$ and its algebraic degree is equal to 6.

Linear Multipermutations mu4/mu8

Both mu4 and mu8 are *linear multipermutations*. This kind of construction was early recognized as being optimal for which regards its diffusion properties (see [287,311] and the discussion in §4.1.3, page 150). As explained in §4.2, not all constructions are very efficient to implement, especially on low-end smartcard, which have usually very few available memory and computational power. We have thus chosen a permuted version of the circulating-like construction described as $\mathbf{M}_8$ for the structure of the matrix (see page 167).

Furthermore, in order to be efficiently implementable, the elements of the matrix, which are elements of $\mathrm{GF}(2^8)$, should be efficient to multiply to. As the only really efficient operations are the addition, the multiplication by α and the division by α. Note that $\alpha^7 + \alpha = \alpha^{-1} + \alpha^{-2}$, $\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 = \alpha^{-1}$, and that $\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha = \alpha^{-2}$.

Key-Schedule Algorithms

The FOX key-schedule algorithms were designed with several rationales in mind: first, the function, which takes a key k and the round number r and returns r subkeys should be a cryptographic pseudorandom, collision resistant and one-way function. Second, the sequence of subkeys should be generated in any direction without any complexity penalty. Third, all the bytes of *mkey* should be randomized even when the key size is strictly smaller than *ek*. Finally, the key-schedule algorithm should resist *related-cipher attacks* as described by Wu in [332] (see §2.3.6 as well), since FOX can possibly use different number of rounds.

We are convinced that "strong" key-schedule algorithms have significant advantages in terms of security, even if the price to pay is a smaller key agility, as discussed earlier. In the case of FOX, we believe that the time needed to compute the subkeys, which is about equal to the time needed to encrypt 6 blocks of data (in the case of FOX64 with keys strictly larger than 128 bit, it takes the time to encrypt 12 blocks of data) remains acceptable in all kinds of applications. During the AES effort, it was suggested that an example of extreme case would be a high-speed network switch having to maintain a million of contexts and switching bewteen them every four blocks of data. Under such extreme constraints, one can still keep in memory one million fully expanded keys at a negligible cost.

The second central property of FOX key-schedule algorithms is ensured by the LFSR construction. As it is possible to back-clock it easily, the subkey generation process can be computed in the encryption as well as in the decryption direction with no loss of speed. The third property is ensured by our "Fibonacci-like" construction (which is a bijective mapping). Furthermore, *mkey* is expanded by XORing constants depending on r and ek with *no overlap* on these constants sequences (this was checked experimentally). Finally, the fourth property is ensured by the dependency of the subkey sequence to the actual round number of the algorithm instance for which the sequence will be used.

We state now a sequence of properties of the building blocks of the key-schedule algorithm.

P-Part The goal of the P-part consists in transforming the user-provided key, which may have any length multiple of 8 smaller or equal than 256, in a fixed-size value of 128-bit or 256-bit. The chosen padding constant $e - 2$ was checked regarding the following property.

Lemma 4.3.1. *It is impossible to find two values of k with a length strictly smaller than ek bits which lead to the same value of pkey.*

Proof. In order for two different inputs to produce the same output during the padding operation, one has to concatenate the smaller one with a padding value which is contained in the one used for the larger input; this is only possible if the first ℓ bytes of the padding constant are present in another location. The lemma follows from the fact that the first byte `0xB7` is unique in the constant. □

Note that in order to avoid that a padded key and non-padded key generate the same subkey sequence, a conditional negation has been incorporated in the NLx part of the key-schedule algorithm.

M-Part When using small keys, a large part of the key-schedule state is known to a potential adversary: it is the padding constant. The goal of

the M-part is hence to mix the entropy on all bytes. The following lemma insures that, when fed with two different inputs, the M-part will return two different outputs.

Lemma 4.3.2. *The* M*-part is a permutation.*

Proof. The lemma follows directly from the fact that the M-part is an invertible application. □

L-Part The goal of the L-part is to diversify the *dkey* register (which serves as input for the NLx-part) at each round. The main design goals are its simplicity and its reversibility: as a LFSR step is equivalent to the multiplication by a constant in a finite field, the inverse operation is a division by the same constant. It is thus possible to evaluate the L in both directions. It was furthermore checked that the outputs (being 144 or 264 bits) for all $12 \leq r \leq 255$ and for all round numbers $1 \leq i \leq r$ are unique.

NLx-Part The goal of the NL part is to generate a pseudorandom stream of data as "cryptographically secure" as possible and as fast as possible; it is actually the one-way part of the key-schedule. For this, it re-uses the round functions in its core, and it needs only a few supplementary operations.

4.3.4 Security Foundations

Security Properties of the Lai-Massey Scheme

Although less popular than the Feistel scheme or SPN structures, the Lai-Massey scheme offers similar (super-) pseudorandomness and decorrelation inheritance properties, as was demonstrated by Vaudenay [317]. Note that we will indifferently use the term "Lai-Massey scheme" to denote both versions, as we can see the Extended Lai-Massey scheme as a Lai-Massey scheme: we can swap the two inner inputs as in Fig. 4.24, and we note that the function $(x, y) \mapsto \mathsf{or32}(x)||\mathsf{or32}(y)$ builds an orthomorphism (see Lem. 4.3.3).

Lemma 4.3.3. *The application defined by*

$$\begin{cases} (\{0,1\}^{32})^2 & \rightarrow & (\{0,1\}^{32})^2 \\ (x,y) & \mapsto & (\mathsf{or}(x), \mathsf{or}(y)) \end{cases}$$

is an orthomorphism, where or(.) *is the orthomorphism defined in §4.3.2.*

Proof. First, we show that this application is a permutation. This follows from the fact that the inverse application is given by

$$(x', y') \mapsto (\mathsf{io}(x'), \mathsf{io}(y'))$$

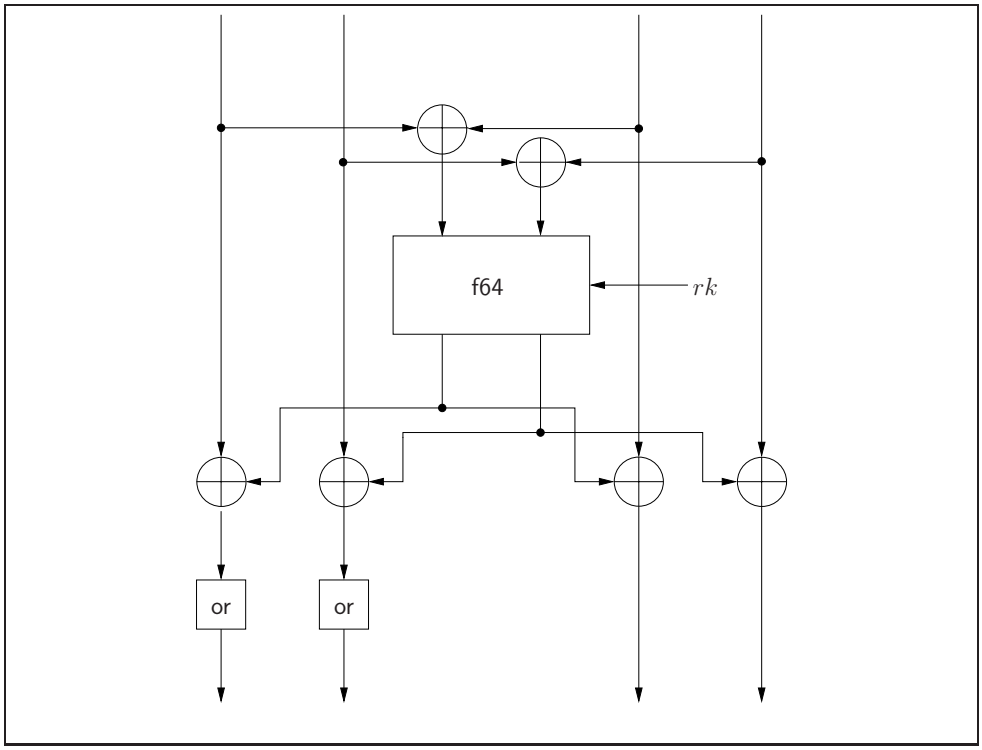

Figure 4.24: An alternate view of an extended Lai-Massey scheme

and that io is a permutation, too. Now, we have to check that

$$(x, y) \mapsto (\mathsf{or}(x) \oplus x, \mathsf{or}(y) \oplus y) \tag{4.4}$$

is also a permutation. This follows easily from the fact that Eq. (4.4) is an invertible application. □

From this point, we will make use of the following notation: given an orthomorphism o on a group $(\mathcal{G}, +)$ and given r functions $\mathsf{f}_1, \mathsf{f}_2, \ldots, \mathsf{f}_r$ on $\mathcal{G}$, we note an r-rounds Lai-Massey scheme using the r functions and the orthomorphism by $\Lambda^{\mathsf{o}}(\mathsf{f}_1, \ldots, \mathsf{f}_r)$. Then the following results are two Luby-Rackoff-like [191] results on the Lai-Massey scheme. We refer to [317, 320] for proofs thereof.

Theorem 4.3.1 (Vaudenay). *Let f_1^*, f_2^* and f_3^* be three independent random functions uniformly distributed on a group $(\mathcal{G}, +)$. Let o be an orthomorphism on $\mathcal{G}$. For any distinguisher limited to d chosen plaintexts, where $g = |\mathcal{G}|$ denotes the cardinality of the group, between $\Lambda^{\mathsf{o}}(\mathsf{f}_1^*, \mathsf{f}_2^*, \mathsf{f}_3^*)$ and a uniformly distributed random permutation c^*, we have*

$$\mathrm{Adv}(\Lambda^{\mathsf{o}}(\mathsf{f}_1^*, \mathsf{f}_2^*, \mathsf{f}_3^*), \mathsf{c}^*) \leq d(d-1)(g^{-1} + g^{-2}).$$

Theorem 4.3.2 (Vaudenay). *If* $\mathsf{f}_1, \ldots, \mathsf{f}_r$ *are* $r \geq 3$ *independent random functions on a group* $(\mathcal{G}, +)$ *of order* g *such that* $\mathrm{Adv}(\mathsf{f}_i, \mathsf{f}_i^*) \leq \frac{\varepsilon}{2}$ *for any adaptive distinguisher between* f_i *and* f_i^* *limited to* d *queries for* $1 \leq i \leq r$ *and if* o *is an orthomorphism on* $\mathcal{G}$*, we have*

$$\mathrm{Adv}(\Lambda^{\circ}(\mathsf{f}_1, \ldots, \mathsf{f}_r), \mathsf{c}^*) \leq \frac{1}{2}(3\varepsilon + d(d-1)(2g^{-1} + g^{-2}))^{\lfloor \frac{r}{3} \rfloor}.$$

Basically, the first result proves that the Lai-Massey scheme provides pseudorandomness on three rounds unless the f_i's are weak, like for the Feistel scheme [102]. Super-pseudorandomness corresponds to cases where a distinguisher can query chosen ciphertexts as well; in this scenario, the previous result holds when we consider $\Lambda^{\circ}(\mathsf{f}_1^*, \ldots, \mathsf{f}_4^*)$ with a fourth round. The second result proves that the decorrelation bias of the round functions of a Lai-Massey scheme is inherited by the whole structure: provided the f_i's are strong, so is the Lai-Massey scheme; in other words, a potential cryptanalysis will not be able to exploit the Lai-Massey's scheme only, but it will have to take advantage of weaknesses of the round functions' internal structure. We would like to stress out the importance of the orthomorphism o: by omitting it, it is possible to distinguish a Lai-Massey scheme using pseudorandom functions from a pseudorandom permutation with overwhelming probability, and this for any number of rounds. Indeed, denoting the input and the output of a Lai-Massey scheme by $x_{\mathsf{l}}||x_{\mathsf{r}}$ and $y_{\mathsf{l}}||y_{\mathsf{r}}$, respectively, the following equation holds with probability one:

$$x_{\mathsf{l}} \ominus x_{\mathsf{r}} = y_{\mathsf{l}} \ominus y_{\mathsf{r}} \tag{4.5}$$

where $\ominus$ denote the inverse of the additive group law used in the scheme.

One should not misinterpret the results in the Luby-Rackoff scenario in terms of the overall block cipher security: FOX's round functions are far to be indistinguishable from random functions, as it is the case of DES round functions, for instance: the fact that DES is vulnerable to linear and differential cryptanalysis does not contradict Luby-Rackoff results. However, Th. 4.3.1 and Th. 4.3.2 give proper credit to the high-level structure of FOX.

Resistance w/r to Linear and Differential Cryptanalysis

It is possible to prove some important results about the security of both f32 and f64 functions towards linear and differential cryptanalysis, too. As these functions may be viewed as classical *Substitution-Permutation Network* constructions, we will refer to some well-known results on their resistance towards linear and differential cryptanalysis proved in [132] by Hong *et al.* For the sake of completeness, we recall the framework of consideration and the results they obtained using it. Then, we apply their result to the round functions of FOX, and we draw some conclusions about its security towards linear and differential cryptanalysis in functions of the round number. This

will help us to fix the minimal number of rounds which results in a sufficient level of security.

Let S_i denote an $m \times m$ bijective substitution box, that is a bijection on $\{0,1\}^m$. We consider a standard kSPkSk structure (i.e. the one of FOX's round functions) on $m \times n$ bit strings, namely a key addition layer, a substitution layer, a diffusion layer, followed by a second key addition layer, a substitution layer, and a final key addition layer. We assume that the substitution layer consists of the parallel evaluation of n $m \times m$ S-boxes S_i for $1 \leq i \leq n$, that the diffusion layer can be expressed as an invertible $n \times n$ MDS matrix $\mathbf{M}$ with coefficients in $\mathrm{GF}(2^m)$, and that the key addition layer consists of XORing a mn-bit subkey to the state. Let us furthermore denote by

$$\pi_{\mathrm{DP}}^{\mathsf{S}} = \max_{1 \leq i \leq n} \mathrm{DP}_{\max}^{\mathsf{S}_i} \text{ and } \pi_{\mathrm{LP}}^{\mathsf{S}} = \max_{1 \leq i \leq n} \mathrm{LP}_{\max}^{\mathsf{S}_i}$$

the respective maximal differential and linear probabilities we can find in the S-boxes S_i, according to the terminology of Def. 2.3.2 and Def. 2.3.9, respectively. Finally, let us denote by

$$\beta = \mathfrak{B}(\mathbf{M}) = n + 1$$

the branch number of the diffusion layer $\mathbf{M}$ (according to Def. 4.1.5), which is defined to be maximal. Then the following theorem due to Hong *et al.* [132] states upper bounds on the maximal differential and linear hull probabilities, respectively.

Theorem 4.3.3 (Hong *et al.* [132]). *In a* kSPkSk *structure, if the round subkeys are statistically independent and uniformly distributed, then the probability of each differential with respect to* $\oplus$ *is upper bounded by*

$$\left(\pi_{\mathrm{DP}}^{\mathsf{S}}\right)^{\beta-1},$$

while the probability of each linear hull is upper bounded by

$$\left(\pi_{\mathrm{LP}}^{\mathsf{S}}\right)^{\beta-1}.$$

In the case of FOX64, since $\mathrm{DP}_{\max}^{\mathsf{sbox}} = \mathrm{LP}_{\max}^{\mathsf{sbox}} = 2^{-4}$, since mu4 (resp. mu8) has a branch number equal to five (resp. nine), and since one can assume that the subkeys are uniformly distributed and statistically independent, due to the nature of the key-schedule algorithm, one can reasonably apply Th. 4.3.3 and get the following result.

Theorem 4.3.4. *If the round subkeys are statistically independent and uniformly distributed, then the following bounds hold:*

$$\mathrm{LP}_{\max}^{\mathsf{f32}} = \mathrm{DP}_{\max}^{\mathsf{f32}} \leq 2^{-16},$$

and

$$\mathrm{LP}_{\max}^{\mathsf{f64}} = \mathrm{DP}_{\max}^{\mathsf{f64}} \leq 2^{-32}.$$

Let us now focus on embedding the round functions in the skeletons. For the sake of clarity[10], we prove now some interesting properties of an Extended Lai-Massey scheme regarding differential and linear characteristics.

Lemma 4.3.4. *In the Extended Lai-Massey scheme as defined in §4.3.2, any* differential *characteristic on two rounds must involve at least one* f64*-function.*

Proof. We follow a top-down approach. If we stack up two rounds of an Extended Lai-Massey scheme (see Fig. 4.25 for a detailed illustration of one round) and we force a differential characteristic at the input of the first f64-function to be equal to 0, then a differential characteristic at the input of the two rounds must have the form (a, b, a, b, c, d, c, d) with $a, b, c, d \in \{0, 1\}^{16}$ and a, b, c, d are not all equal to 0. At the end of the first round, the differential characteristic sounds $(b, a \oplus b, a, b, d, c \oplus d, c, d)$. At the input of the second f64-function, the differential characteristic is equal to $(a \oplus b, a, c \oplus d, c)$. We proceed by contraposition. If the input of the second f64-function is equal to zero, we have $a = c = 0$. As $a \oplus b$ and $c \oplus d$ must be both equal to 0, the we conclude that $a = b = c = d = 0$. This is a contradiction to our primary assumption about a, b, c and d, and the theorem follows. □

Lemma 4.3.5. *In the Extended Lai-Massey scheme as defined in Fig. 4.3.2, any* linear *characteristic on two rounds must involve at least one function* f64*.*

Proof. We follow a bottom-up approach. By forcing a linear characteristic to be equal to $(0, 0, 0, 0, 0, 0, 0, 0)$ at the end of the second f64-function, we note that the output linear characteristic must have the form $(a, a \oplus d, a \oplus d, d, b, b \oplus c, b \oplus c, c)$ with $a, b, c, d \in \{0, 1\}^{16}$ and a, b, c, d not all equal to 0. If we consider now the first f64-function, we note that a linear characteristic at its output must have the form $(d, a \oplus d, b, b \oplus c)$, which implies that $a = b = 0$ and then that $c = d = 0$, which is a contradiction to our assumption, and the theorem follows. □

By considering Th. 4.3.3, Lem. 4.3.4, and Lem. 4.3.5 together, we have thus the following result.

Theorem 4.3.5. *The differential (resp. linear) probability of any single-path characteristic in* FOX64$/k/r$ *is upper bounded by* $(\mathrm{DP}^{\mathsf{sbox}}_{\max})^{2r}$ *(resp.* $(\mathrm{LP}^{\mathsf{sbox}}_{\max})^{2r}$*). Similarly, the bounds are* $(\mathrm{DP}^{\mathsf{sbox}}_{\max})^{4r}$ *(resp.* $(\mathrm{LP}^{\mathsf{sbox}}_{\max})^{4r}$*) for* FOX128$/k/r$*.*

[10] These properties are actually trivial to prove in the case of a simple Lai-Massey scheme, and as discussed in §4.3.4, the Extended Lai-Massey scheme can be viewed as a simple Lai-Massey scheme.

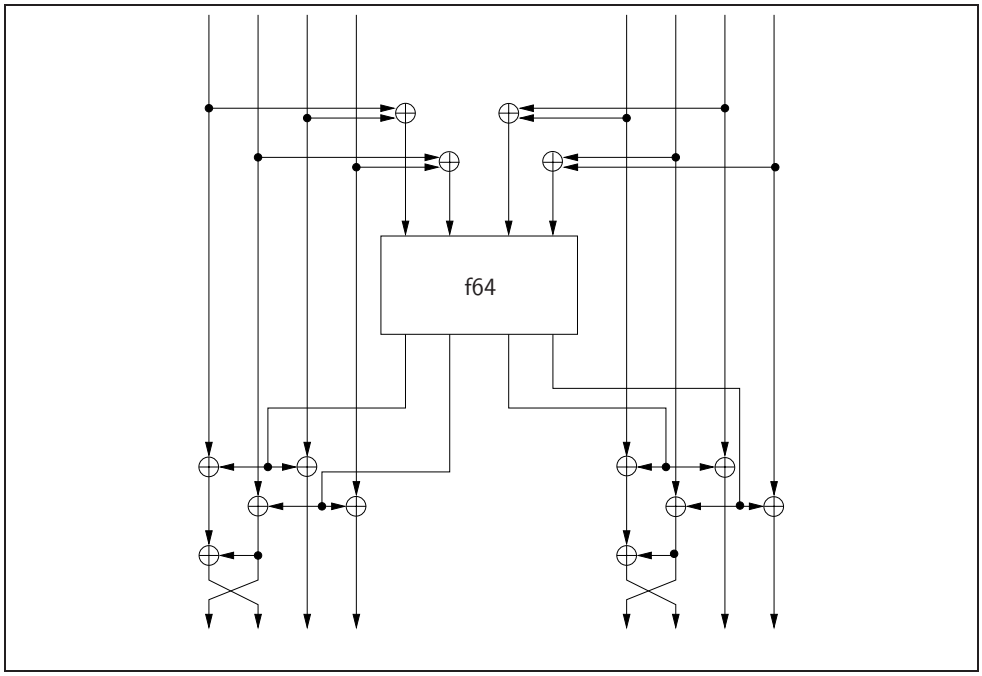

Figure 4.25: A detailed view of an extended Lai-Massey scheme

Note that it is a kind of "hybrid" proof of security towards linear and differential cryptanalysis, as we have considered differential and linear hulls in the round functions, but characteristics in the high-level schemes. Thus, we have in reality slightly stronger results that the ones stated in Th. 4.3.5. Finally, we conclude that it is impossible to find any useful differential or linear characteristic after 8 rounds for both FOX64 and FOX128. Hence, a minimal number of 12 rounds provides a minimal safety margin.

Resistance Towards Other Attacks

In this part, we discuss the resistance of FOX towards various types of attacks.

Statistical Attacks Due to the very high diffusion properties of FOX's round functions, the high algebraic degree of the sbox mapping, and the high number of rounds, we are strongly convinced that FOX will resist to known variants of linear and differential cryptanalysis (like differential-linear cryptanalysis [29, 183], boomerang [322] and rectangle attacks [28]), as well as generalizations thereof, like Knudsen's truncated and higher-order differentials [161], impossible differentials [26], and Harpes' partitioning cryptanalysis [126], for instance.

Slide and Related-Key Attacks Slide attacks [37, 38] exploit periodic key-schedule algorithms, which is not a property of FOX's key-schedule algorithms. Furthermore, due to very good diffusion and the high non-linearity of the key-schedule, related-key attacks are very unlikely to be effective against FOX.

Interpolation and Algebraic Attacks Interpolation attacks [142] take advantage of S-boxes exhibiting a simple algebraic structure. Since FOX's non-linear mapping sbox does not possess any simple relation over GF(2) or GF(2^8), such attacks are certainly not effective.

One of our main concerns was to avoid a pure algebraic construction for the sbox mapping, as it is the case for a large number of modern designs of block ciphers. Although such S-boxes have many interesting non-linear properties, they probably form the best conditions to express a block cipher as a system of sparse, over-defined low-degree multivariate polynomial equations over GF(2) or GF(2^8); this fact may lead to effective attacks, as argued by Courtois and Pieprzyk in [70].

Not choosing an algebraic construction for sbox does not necessarily ensure security towards algebraic attacks. Note that we base our non-linear mapping on "small" permutations, mapping 4 bits to 4 bits, and that, according to [70], *any* such mapping can always be written as an overdefined system of *at least* 21 quadratic equations: let us denote the input (resp. the output) of such a small S-box by $x_1||x_2||x_3||x_4$ (resp. by $y_1||y_2||y_3||y_4$), and if we consider a 16×37 matrix containing in each row the values of the $t = 37$ monomials

$$\{1, x_1, \ldots, x_4, y_1, \ldots, y_4, x_1x_2, \ldots, x_1y_1, \ldots, y_3y_4\}$$

for each of the 16 possible entries, we note that its rank can be at most 16, thus, for any S-box, there will be at least $\rho \geq 37 - 16 = 21$ quadratic equations. We have checked that the rank of these matrices for FOX's small S-boxes S_1, S_2, and S_3 are equal to 16, and there exist thus 21 quadratic equations describing it; furthermore, we are not aware of any quadratic relation over GF $\left(2^8\right)$ for sbox. Following the very same methodology than [70], it appears that XSL attacks *would* break members of the FOX family within a complexity[11] of 2^{139} to 2^{156}, depending on the block size and on the rounds number.

Namely, we can construct an overdefined multivariate system of quadratic equations describing FOX using the XSL approach, which aims at recovering all the subkeys, without taking care of the key-schedule algorithm. Let us assume that FOX has r rounds, and thus r subkeys with the same size than the plaintext. We need hence r known plaintext-ciphertext pairs to uniquely determine the key. We use from now on the same notations than

[11] Under the most pessimistic hypotheses.

in [70]. S is defined to be the total number of substitution boxes considered during an attack. Hence,

$$S_{\mathsf{FOX64}} = 3 \cdot 8 \cdot r^2$$

for FOX64, and

$$S_{\mathsf{FOX128}} = 3 \cdot 16 \cdot r^2$$

as each substitution box sbox is built from three small S-boxes on $\{0,1\}^s$, with $s = 4$. Let t denote the number of monomials (i.e. $t = 37$ in our case), let t' being the number of terms in the basis for one S-box that can be multiplied by some fixed variable and are still in the basis (we have $t' = 5$ in the case of FOX). Then, Courtois and Pieprzyk [70] estimate that the complexity of a XSL attack can be estimated to

$$T^\omega \text{ with } T \approx (t-\rho)^P \cdot \binom{S}{P}$$

where ω is the best possible exponent for Gaussian elimination, T represents the total number of terms, and where

$$P = \frac{t-\rho}{s + \frac{t'}{S}}$$

In the case of FOX, we get

$$P = \frac{16}{4 + \frac{5}{24r^2}} < 4$$

According to Courtois and Pieprzyk [70], in order that the attack works, as difference operation) it is necessary to choose P such that

$$\frac{R}{T - T'} \geq 1 \tag{4.6}$$

where

$$R \approx S \cdot s(t-\rho)^{P-1} \cdot \binom{S}{P-1}$$

represents the total number of equations, and

$$T' \approx t'(t-\rho)^{P-1} \cdot \binom{S-1}{P-1}$$

is the total number of terms in the basis that can be multiplied by some fixed variable and are still in the basis. Eq. (4.6), in the case which interests us, is already fullfiled for $P = 4$, but $R \approx 1$. As the overall complexity of the attack is very sensitive to the value of P, and according to Courtois and Pieprzyk [70],

		$P = 4$		$P = 5$	
	S	$\omega = 2.376$	$\omega = 3$	$\omega = 2.376$	$\omega = 3$
FOX64/k/12	3456	2^{139}	2^{175}	2^{171}	2^{216}
FOX64/k/16	6144	2^{147}	2^{185}	2^{181}	2^{228}
FOX128/k/12	6912	2^{148}	2^{187}	2^{183}	2^{231}
FOX128/k/16	12288	2^{156}	2^{197}	2^{192}	2^{243}

Figure 4.26: Estimations of the complexity of Courtois-Pieprzyk attacks against FOX

> *Though XSL attacks will probably always work for some P, we considered the minimum value P for which $\frac{R}{T-T'} \geq 1$. This condition is necessary, but probably not sufficient.*

we will consider the cases $P = 4$ as well as $P = 5$ in our estimations of the complexity of applying algebraic attacks to FOX.

Another subject of controversy is the value of ω, i.e. the complexity exponent of a Gaussian reduction. Courtois and Pieprzyk [70] assume that $\omega = 2.376$, which is the best known value obtained by Coppersmith and Winograd [62]. According to [70], the constant factor in this algorithm is unknown to the authors of [62], and is expected to be very big. Accordingly, it is disputed whether such an algorithm can be applied efficiently in practice. For this reason, we will consider both $\omega = 2.376$ and $\omega = 3$ in our estimations.

A summary of our estimations is given in Fig. 4.26. At the light of the previous discussion, we should interpret these figures with an extreme care: on the one hand, the real complexity of XSL attacks is by no means clear at the time of writing and is the subject of much controversy [236]; one the other hand, we feel that the advantages of a small hardware footprint overcome such a (possible) security decrease.

Integral Attacks Integral attacks [169] apply to ciphers operating on well-aligned data, like SPN structures. As the round functions of FOX are SPNs, one can wonder whether it is possible to find an integral distinguisher on the whole structure and we show now that it is indeed the case.

Let us consider the case of FOX64: we denote the input bytes by $x_{i(8)}$ with $0 \leq i \leq 7$ and the output of the third round lmid64 by $y_{i(8)}$ with $0 \leq i \leq 7$. We have the following integral distinguisher on 3 rounds of FOX64.

Theorem 4.3.6. *Let $x_{3(8)} = a$, $x_{7(8)} = a \oplus c$, and $x_{i(8)} = c$ for $i = 0, 1, 2, 4, 5, 6$, where c is an arbitrary constant. We consider plaintext struc-*

tures $x^{(j)}$ *for* $1 \leq j \leq 256$ *where* a *takes all 256 possibles byte values. Then,*

$$\begin{aligned}
\bigoplus_{j=1}^{256} y_0^{(j)} \oplus y_6^{(j)} &= 0, \\
\bigoplus_{j=1}^{256} y_1^{(j)} \oplus y_7^{(j)} &= 0, \\
\bigoplus_{j=1}^{256} y_0^{(j)} \oplus y_2^{(j)} \oplus y_4^{(j)} &= 0, \\
\bigoplus_{j=1}^{256} y_1^{(j)} \oplus y_3^{(j)} \oplus y_5^{(j)} &= 0.
\end{aligned}$$

Proof. See Fig. 4.27, where "C" denotes a constant byte, "A" denotes an active byte, and "S" denotes a byte, whose sum under the structure is equal to zero. □

This integral distinguisher can be used to break (four, five) six rounds of FOX64 (by guessing the one, two, or three last round keys and testing the integral criterion for each subkey candidate on a few structures of plaintexts) within a complexity of about (2^{72}, 2^{136}) 2^{200} partial decryptions and negligible memory. A similar property may be used to break up to 4 rounds of FOX128 (by guessing the last round key) with a complexity of about 2^{136} operations and negligible memory.

4.3.5 Implementation Issues

In this part, we discuss several issues about the implementation of the FOX family on low-end 8-bit architectures and on high-end 32/64-bit ones. Finally, we give results about the performances of various implementations we have written on different platforms.

8-bit Architectures

The resources representing the most important bottleneck in a block cipher implementation on a smartcard (which uses typically low-cost, 8-bit microprocessors) is of course the RAM usage. The amount of efficiently usable RAM available on a smartcard is typically in the order of 256 bytes. It may be a bit larger depending on the cases, but as this type of smart card is devoted to contain more than a simple encryption routine, FOX implementations on this kind of platforms will minimize the amount of necessary RAM. ROM is not so scarce as RAM on a smartcard, so the code size can be greater than the RAM usage. It is usually reasonable not to have a ROM size (instructions + possible precomputed tables) greater than 1024 bytes.

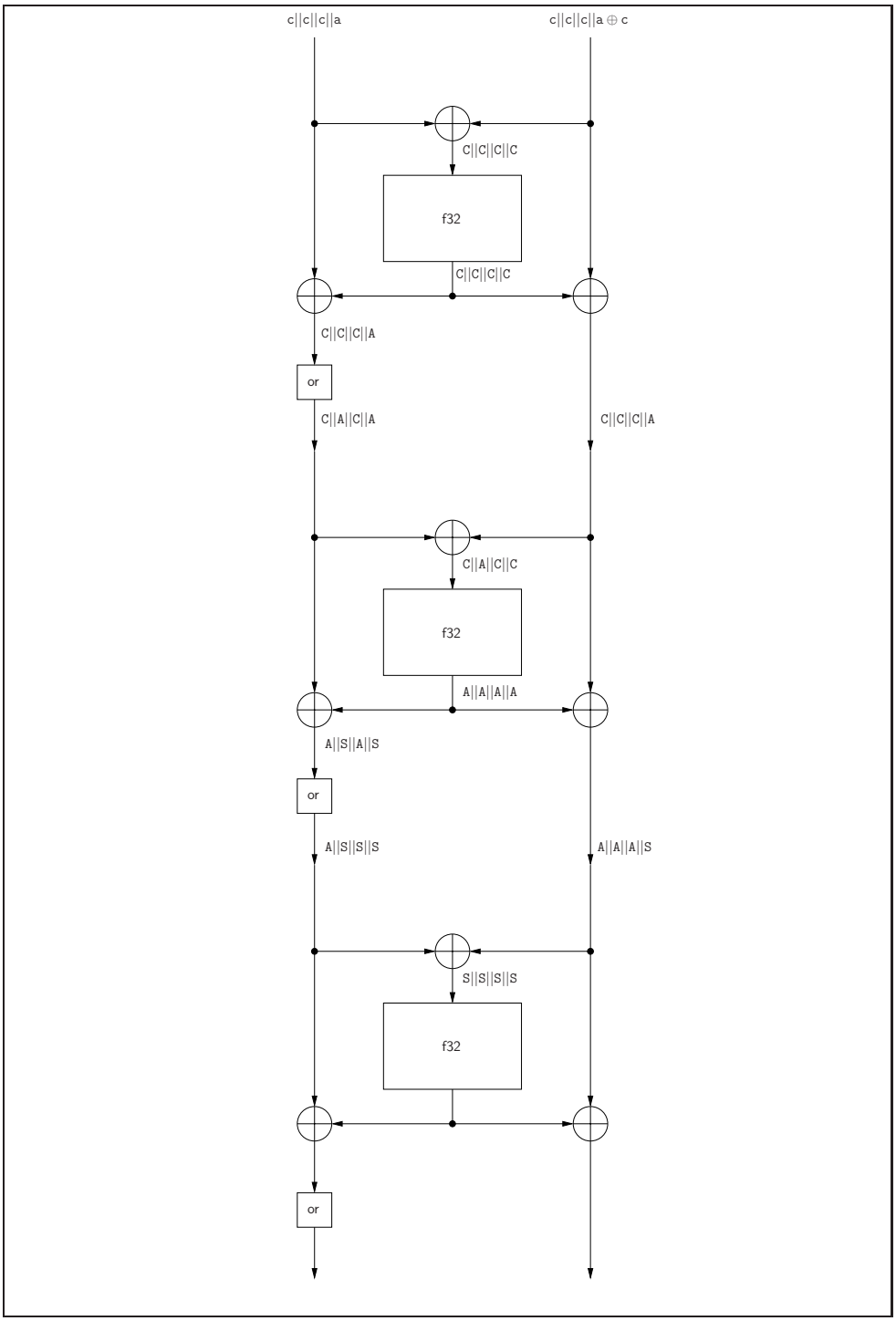

Figure 4.27: Integral Distinguisher in 3 rounds of FOX64.

Strategy	Precomputations	Data size
A	No precomputed data	24 B
B	sbox	256 B
C	sbox, talpha, dalpha	768 B
D	sbox, stalpha, sdalpha, stalpha2, sdalpha2	1280 B

Figure 4.28: Four different strategies to implement FOX on low-end microprocessors

Four Memory Usage Strategies Obviously, the most intensive computation are related to the evaluation of the sbox mapping and of the mu4 and mu8 mappings. We propose in the following four different (the last one concerning uniquely FOX128) strategies using various amounts of precomputed data to implement these mappings; they are summarized in Fig. 4.28. Note that the precomputed data may be stored in ROM and that the constants needed in the key-schedule algorithm are not taken into account. Strategy A can be applied when extremely few memory is available. For this, one computes on-the-fly the sbox mapping, as it is described in §4.3.3, page 192, and all the operations in GF $\left(2^8\right)$. The sole needed constants are the small substitution boxes S_1, S_2 and S_3 (see Fig. 4.23). Strategy A is clearly the slowest one. A significant speed gain can be obtained if one precomputes the sbox mapping (Strategy B), the finite field operations being all computed dynamically. A third possibility (Strategy C) is to precompute two more mappings: $\mathsf{talpha}(x)$ is a function mapping an element x to $\alpha \cdot x$, with the multiplication in GF $\left(2^8\right)$; $\mathsf{dalpha}(x)$ is a function mapping an element $x \in$ GF $\left(2^8\right)$ to $\alpha^{-1} \cdot x$. Finally, in the case of FOX128, a further speed gain may be obtained (Strategy D) by tabulating the five following mappings:

$$\begin{array}{lcl} \mathsf{sbox}(x) & : & x \mapsto \mathsf{sbox}(x) \\ \mathsf{stalpha}(x) & : & x \mapsto \mathsf{sbox}(x) \cdot \alpha \\ \mathsf{sdalpha}(x) & : & x \mapsto \mathsf{sbox}(x) \cdot \alpha^{-1} \\ \mathsf{stalpha2}(x) & : & x \mapsto \mathsf{sbox}(x) \cdot \alpha^{2} \\ \mathsf{sdalpha2}(x) & : & x \mapsto \mathsf{sbox}(x) \cdot \alpha^{-2} \end{array}$$

The implementation of the sigma4/mu4 layer is relatively straighforward:

$$\begin{array}{rcl} y_{0(8)} & = & \mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{2(8)}) \oplus \\ & & \alpha \cdot \mathsf{sbox}(x_{3(8)}) \\ y_{1(8)} & = & \mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{3(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{2(8)}) \\ & & \oplus \alpha^{-1} \cdot \mathsf{sbox}(x_{1(8)}) \\ y_{2(8)} & = & \mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{3(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{1(8)}) \\ & & \oplus \alpha^{-1} \cdot \mathsf{sbox}(x_{0(8)}) \end{array}$$

$$\begin{aligned} y_{3(8)} &= \mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{3(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{0(8)}) \\ &\quad \oplus \alpha^{-1} \cdot \mathsf{sbox}(x_{2(8)}) \end{aligned}$$

By carefully rewriting the above equations and by re-using some temporary results, one can easily minimize the number of sbox, talpha, dalpha evaluations and the number of $\oplus$ operations. However, the resulting implementation is strongly dependent of the chosen strategy.

The implementation of the sigma8/mu8 layer is not much complicated. By rewriting the operations as done above, one can easily obtain a fast implementation. For instance, in case of an implementation following memory strategy C, one can obtain the following computations:

$$\begin{aligned} y_{0(8)} &= \mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{3(8)}) \oplus \\ &\quad \mathsf{sbox}(x_{4(8)}) \oplus \mathsf{sbox}(x_{5(8)}) \oplus \mathsf{sbox}(x_{6(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{7(8)}) \\ y_{1(8)} &= \mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{7(8)}) \oplus \\ &\quad \alpha \cdot \big(\mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{3(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{4(8)}\big) \oplus \\ &\quad \alpha^{-1} \cdot \big(\mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{5(8)}) \oplus \alpha^{-1} \cdot (\mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{6(8)}))\big) \\ y_{2(8)} &= \mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{6(8)}) \oplus \mathsf{sbox}(x_{7(8)}) \oplus \\ &\quad \alpha \cdot \big(\mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{2(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{3(8)}\big) \oplus \\ &\quad \alpha^{-1} \cdot \big(\mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{4(8)}) \oplus \alpha^{-1} \cdot (\mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{5(8)}))\big) \\ y_{3(8)} &= \mathsf{sbox}(x_{5(8)}) \oplus \mathsf{sbox}(x_{6(8)}) \oplus \mathsf{sbox}(x_{7(8)}) \oplus \\ &\quad \alpha \cdot \big(\mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{6(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{2(8)}\big) \oplus \\ &\quad \alpha^{-1} \cdot \big(\mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{3(8)}) \oplus \alpha^{-1} \cdot (\mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{4(8)}))\big) \\ y_{4(8)} &= \mathsf{sbox}(x_{4(8)}) \oplus \mathsf{sbox}(x_{5(8)}) \oplus \mathsf{sbox}(x_{7(8)}) \oplus \\ &\quad \alpha \cdot \big(\mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{5(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{1(8)}\big) \oplus \\ &\quad \alpha^{-1} \cdot \big(\mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{6(8)}) \oplus \alpha^{-1} \cdot (\mathsf{sbox}(x_{3(8)}) \oplus \mathsf{sbox}(x_{6(8)}))\big) \\ y_{5(8)} &= \mathsf{sbox}(x_{3(8)}) \oplus \mathsf{sbox}(x_{4(8)}) \oplus \mathsf{sbox}(x_{7(8)}) \oplus \\ &\quad \alpha \cdot \big(\mathsf{sbox}(x_{4(8)}) \oplus \mathsf{sbox}(x_{6(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{0(8)}\big) \oplus \\ &\quad \alpha^{-1} \cdot \big(\mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{5(8)}) \oplus \alpha^{-1} \cdot (\mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{5(8)}))\big) \\ y_{6(8)} &= \mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{3(8)}) \oplus \mathsf{sbox}(x_{7(8)}) \oplus \\ &\quad \alpha \cdot \big(\mathsf{sbox}(x_{3(8)}) \oplus \mathsf{sbox}(x_{5(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{6(8)}\big) \oplus \\ &\quad \alpha^{-1} \cdot \big(\mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{4(8)}) \oplus \alpha^{-1} \cdot (\mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{4(8)}))\big) \\ y_{7(8)} &= \mathsf{sbox}(x_{1(8)}) \oplus \mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{7(8)}) \oplus \\ &\quad \alpha \cdot \big(\mathsf{sbox}(x_{2(8)}) \oplus \mathsf{sbox}(x_{4(8)}) \oplus \alpha \cdot \mathsf{sbox}(x_{5(8)}\big) \oplus \\ &\quad \alpha^{-1} \cdot \big(\mathsf{sbox}(x_{3(8)}) \oplus \mathsf{sbox}(x_{6(8)}) \oplus \alpha^{-1} \cdot (\mathsf{sbox}(x_{0(8)}) \oplus \mathsf{sbox}(x_{3(8)}))\big) \end{aligned}$$

This computation flow (consisting of 71 $\oplus$, 15 talpha and 15 dalpha evaluations) is obviously not optimal in terms of operations; by using redundant temporary computations, one can spare a few more operations.

We give now a constant-time implementation of talpha and dalpha. The routines talpha2 and dalpha2 can be implemented by iterating twice talpha and dalpha, respectively. Note that these implementations do not take into account security issues related to other side-channel attacks, like SPA/DPA.

```
;; Implementation of talpha() on 8051
;;
;; R0    : input
;; R0    : output

MOV A, R0                    ;; A := R0
RLC A                        ;; left rotation through carry
MOV R0, A                    ;; storing the result
CLR A                        ;; A := 0
SUBB A, #0                   ;; C set ? A = 0xFF : A = 0x00
ANL A, #F9                   ;; C set ? A = 0xF9 : A = 0x00
XRL A, R0                    ;; A := A XOR R0
MOV R0, A                    ;; R0 := A

;; Implementation of dalpha() on 8051
;;
;; R0    : input
;; R0    : output

MOV A, R0                    ;; A := R0
RRC A                        ;; left rotation through carry
MOV R0, A                    ;; storing the result
CLR A                        ;; A := 0
SUBB A, #0                   ;; C set ? A = 0xFF : A = 0x00
ANL A, #FC                   ;; C set ? A = 0xFC : A = 0x00
XRL A, R0                    ;; A := A XOR R0
MOV R0, A                    ;; R0 := A
```

32/64-bit Architectures

Most modern CPUs architecture are 32- or 64-bit ones. In this section, we list several ways to optimize an implementation of FOX in terms of speed (i.e. of throughput).

Subkeys Precomputation Most of the time, block ciphers are used to encrypt *several* blocks of data, so it is very time-sparing to precompute the subkeys once for all and to store them in a table. Typically, one needs 128 bytes of memory to store all the subkeys for an implementation of FOX64 with 16 rounds and twice as much for FOX128.

Implementation of f32 and f64 using Table-Lookups The f32 and f64 functions can be implemented very efficiently using a combinations of table-lookups and XORs. We will focus on the f32 function, but the considerations are similar for which concerns f64. Let $x_{0(8)}||x_{1(8)}||x_{2(8)}||x_{3(8)}$ be an input of f32. We denote by $t_{0(8)}||t_{1(8)}||t_{2(8)}||t_{3(8)}$ the temporary result obtained after the mu4 application. Let $rk_{0(8)}||rk_{1(8)}||rk_{2(8)}||rk_{3(8)}$ denote the first half of the round key. Finally, let $v_{i(8)} = x_{i(8)} \oplus rk_{i(8)}$ for $0 \leq i \leq 3$. We have

$$\begin{pmatrix} t_{0(8)} \\ t_{1(8)} \\ t_{2(8)} \\ t_{3(8)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \alpha \\ 1 & c & \alpha & 1 \\ c & \alpha & 1 & 1 \\ \alpha & 1 & c & 1 \end{pmatrix} \times \begin{pmatrix} \mathsf{sbox}(v_{0(8)}) \\ \mathsf{sbox}(v_{1(8)}) \\ \mathsf{sbox}(v_{2(8)}) \\ \mathsf{sbox}(v_{3(8)}) \end{pmatrix}$$

This equation may be rewritten as

$$\begin{aligned} \begin{pmatrix} t_{0(8)} \\ t_{1(8)} \\ t_{2(8)} \\ t_{3(8)} \end{pmatrix} \quad = \quad & \mathsf{sbox}(v_{0(8)}) \times \begin{pmatrix} 1 \\ 1 \\ c \\ \alpha \end{pmatrix} \oplus \mathsf{sbox}(v_{1(8)}) \times \begin{pmatrix} 1 \\ c \\ \alpha \\ 1 \end{pmatrix} \oplus \\ & \mathsf{sbox}(v_{2(8)}) \times \begin{pmatrix} 1 \\ \alpha \\ 1 \\ c \end{pmatrix} \oplus \mathsf{sbox}(v_{3(8)}) \times \begin{pmatrix} \alpha \\ 1 \\ 1 \\ 1 \end{pmatrix} \end{aligned}$$

Thus, one may precompte 4 tables of 256 4-bytes elements defined by

$$\mathtt{TBSM_0}[\mathsf{a}] = \begin{pmatrix} 1 \cdot \mathsf{sbox}(\mathsf{a}) \\ 1 \cdot \mathsf{sbox}(\mathsf{a}) \\ c \cdot \mathsf{sbox}(\mathsf{a}) \\ \alpha \cdot \mathsf{sbox}(\mathsf{a}) \end{pmatrix}, \qquad \mathtt{TBSM_1}[\mathsf{a}] = \begin{pmatrix} 1 \cdot \mathsf{sbox}(\mathsf{a}) \\ c \cdot \mathsf{sbox}(\mathsf{a}) \\ \alpha \cdot \mathsf{sbox}(\mathsf{a}) \\ 1 \cdot \mathsf{sbox}(\mathsf{a}) \end{pmatrix}$$

$$\mathtt{TBSM_2}[\mathsf{a}] = \begin{pmatrix} 1 \cdot \mathsf{sbox}(\mathsf{a}) \\ \alpha \cdot \mathsf{sbox}(\mathsf{a}) \\ 1 \cdot \mathsf{sbox}(\mathsf{a}) \\ c \cdot \mathsf{sbox}(\mathsf{a}) \end{pmatrix}, \qquad \mathtt{TBSM_3}[\mathsf{a}] = \begin{pmatrix} \alpha \cdot \mathsf{sbox}(\mathsf{a}) \\ 1 \cdot \mathsf{sbox}(\mathsf{a}) \\ 1 \cdot \mathsf{sbox}(\mathsf{a}) \\ 1 \cdot \mathsf{sbox}(\mathsf{a}) \end{pmatrix}$$

and write

$$\begin{pmatrix} t_{0(8)} \\ t_{1(8)} \\ t_{2(8)} \\ t_{3(8)} \end{pmatrix} = \mathtt{TBSM_0}[v_{0(8)}] \oplus \mathtt{TBSM_1}[v_{1(8)}] \oplus \mathtt{TBSM_2}[v_{2(8)}] \oplus \mathtt{TBSM_3}[v_{3(8)}]$$

Similarly, we can denote the temporary result after the second key-addition layer of f32 *before* the last substitution layer by $u_{0(8)}||u_{1(8)}||u_{2(8)}||u_{3(8)}$ and

by $w_{0(8)}||w_{1(8)}||w_{2(8)}||w_{3(8)}$, the temporary result *after* the last substitution layer, one can use the same strategy with the following tables:

$$\mathtt{TBS}_0[\mathtt{a}] = \begin{pmatrix} \mathsf{sbox}(\mathtt{a}) \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \mathtt{TBS}_1[\mathtt{a}] = \begin{pmatrix} 0 \\ \mathsf{sbox}(\mathtt{a}) \\ 0 \\ 0 \end{pmatrix}$$

$$\mathtt{TBS}_2[\mathtt{a}] = \begin{pmatrix} 0 \\ 0 \\ \mathsf{sbox}(\mathtt{a}) \\ 0 \end{pmatrix}, \qquad \mathtt{TBS}_3[\mathtt{a}] = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \mathsf{sbox}(\mathtt{a}) \end{pmatrix}$$

and write

$$\begin{pmatrix} w_{0(8)} \\ w_{1(8)} \\ w_{2(8)} \\ w_{3(8)} \end{pmatrix} = \mathtt{TBS}_0[u_{0(8)}] \oplus \mathtt{TBS}_1[u_{1(8)}] \oplus \mathtt{TBS}_2[u_{2(8)}] \oplus \mathtt{TBS}_3[u_{3(8)}]$$

As outlined before, the process is similar for the implementation of the f64 function. In this case, we have to define two times 8 tables of 256 64-bit elements. The following table summarizes the size of the various tables for a fully-precomputed implementation :

	number of tables	width [bytes]	total size [bytes]
FOX64	2×4	4	8192
FOX128	2×8	8	32768

Depending on the target processor, the nearest cache (i.e. the fastest memory) size may be smaller than 32768 bytes. In this case, one can spare half of the tables (at the cost of a few masking operations) by noting that all the TBS tables are "embedded" in the TBSM ones; this implementation strategy will by denoted *half-precomputed implementation*. This allows to reduce the fast memory needs to 4096 and 16384 bytes, respectively. Fig. 4.29 summarizes the best strategies for various amounts of L1 cache memory.

For most modern microprocessors (denoted by $*$ in Fig. 4.29), a fully-precomputed implementation of FOX64 and FOX128 is probably the fastest possible solution. For the processors denoted by $\bullet$, a half-precompted implementation is likely the best solution. The supplementary masking operations may be furthermore used to increase the instructions throughput on pipelined architectures.

Some microprocessors have a very small L1 data cache (they are denoted in $\star$ in Fig. 4.29). In the case of FOX128, even a half-precomputed implementation will result in many caches misses, inducing a performance

penalty. For early versions of Intel Pentium IV, a half-precomputed implementation of FOX64 is advantageous, while one can reduce the size of the precomputed data needed for a FOX128 implementation down to 8192 bytes at the cost of at most 18 supplementary `PSHUFW` instructions. Although these operations will result in a performance penalty, the latter will be reduced since the highly-parrallelizable structure of the f64 function allows to fully use the pipeline and thus to improve the instructions throughput. As most modern CPU architectures are pipelined ones, one can take this fact into account in order to improve performances of FOX implementations. There are two "dependency walls" in a FOX round function. The first one is just after the first subkey addition, the second one just after the second subkey addition. Inbetween, the additions of the table-lookup results may be done in any order, as an XOR is a commutative addition.

FOX128 is an excellent candidate for using the 64-bit instructions of actual 32-bit microprocessors. For instance, on the Intel architecture, the MMX/SSE/SSE2/SSE3 instruction sets may be used to "emulate" a 64-bit microprocessor. Furthermore, by expressing the Extended Lai-Massey scheme as in Fig. 4.24, one can compute very efficiently the two orthomorphisms as a single one on 64-bit architectures.

In order to get the best performances for FOX implementations written in a high-level language, one can get large speed differences when using different compilers. Furthermore, the choice of the data structure of the precomputed tables and of the data to be encrypted plays an important role: implementing a simple way to access these data will result in a speed increase.

Key-Schedule Algorithms For applications needing a high key-agility, one can implement the various key-schedule algorithms using the same guidelines and tricks as for the core algorithm, since they share many common features.

Performance Results

Fig. 4.30 summarizes the results obtained at the time of writing by our optimized implementations of the FOX family (in clock cycles to encrypt one block, with precomputed subkeys): We note that FOX64 is extremely fast on 32-bit architectures, while FOX128 is competitive on 64-bit architectures. Namely, according to the Nessie project [247], FOX64/12 is the fourth fastest 64-bit block cipher on Pentium 3 behind Nimbus, CAST128 and RC5. It is 19% faster than Misty1 (NESSIE's choice), 39% faster than IDEA, 57% faster than DESand about three times faster than Triple-DES. The generic version of FOX64 (with 16 rounds) is still 8% faster than IDEA. On the 64-bit architecture Alpha 21264, FOX128/12 is the third fastest block cipher behind NUSH and AES, according to [247], while FOX128 (16 rounds) with

Processor	cache size [kB]	Note	Best Strategy
Alpha 21164	8	(data)	$\star$
Alpha 21264	64	(data)	$*$
AMD Athlon XP	128	(data + code)	$*$
AMD Athlon MP	128	(data + code)	$*$
AMD Opteron	64	(data)	$*$
Intel Pentium III	16	(data)	$\bullet$
Intel Pentium IV	8/16 (Prescott)	(data)	$\star$ / $\bullet$
Intel Xeon	8	(data)	$\star$
Intel Itanium	16	(data)	$\bullet$
Intel Itanium2	16	(data)	$\bullet$
PowerPC G4	32	(data + code)	$\bullet$
PowerPC G5	32	(data)	$*$
UltraSparc II	16	(data)	$\bullet$
UltraSparc III	64	(data)	$*$

Figure 4.29: Best implementation strategies on 32/64-bit microprocessors

Cipher	Architecture	Implementation	$r = 12$	$r = 16$
FOX64/k/r	Intel Pentium 3	C (`gcc`)	316	406
FOX64/k/r	Intel Pentium 3	ASM	220	295
FOX64/k/r	Intel Pentium 4	C (`gcc`)	388	564
FOX64/k/r	AMD Athlon-XP	C (`gcc`)	306	390
FOX64/k/r	Alpha 21264	C (Compaq `cc`)	360	480
FOX128/k/r	Intel Pentium 3	C (`gcc`)	636	840
FOX128/k/r	AMD Athlon-XP	C (`gcc`)	544	748
FOX128/k/r	Alpha 21264	C (Compaq `cc`)	440	588

Figure 4.30: Performances of optimized implementations of FOX

256-bit keys is still 30% faster than Camellia, which is one of NESSIE's choices. Finally, we have an implementation of FOX64/12 (resp. FOX64/16) on 8051, a typical low-cost 8-bit architecture, needing 16 bytes of RAM, 896 bytes of ROM (precomputed data and precomputed subkeys) and 575 bytes of code size encrypting one block in 2958 (resp. 3950) clock cycles.

Chapter 5

Conclusion and Open Problems

As demonstrated, for instance, by the heuristic nature of the design of FOX, we are still far from a "provably secure", and still practically fast block cipher. Currently, the only possibility to assess the security level of a block cipher consists in studying and trying to break it (being its full version, or reduced-round versions thereof). Therefore, the security level of a given block cipher keeps merely a function of the quantity of time and efforts invested by cryptanalysts, and a "secure" block cipher has only a time-security curve slowly decreasing; the IDEA block cipher surely is a very good example possessing such a curve, as demonstrated by Fig. 3.10, page 137. Obviously, proposing a new block cipher simultaneously defines open problems. Ideally, the security of FOX should be investigated thoroughly by different people, notably for which concerns Courtois-Pieprzyk algebraic attacks, before benefiting from the confidence of the cryptologic community, and by extension, of the industrial world. We would like to outline here that even in the unlikely case where using 4-bit S-boxes would result in a clear security decrease, and render FOX impractically insecure, replacing them by stronger S-boxes keeps a rather easy task.

The study of generic distinguishers in a formal way is a small step towards a better understanding of *necessary* mathematical conditions on block ciphers for security. In this thesis, we hope to have demonstrated that using old statistical tools, well-known and well applied in many engineering fields, but which were perhaps neglected so far in the domain of block ciphers, can help us to easily derive interesting properties of computationally unbounded distinguishers in a rather easy way. Still, many open problems remain: we have thoroughly treated the class of attacks which we can model by a non-adaptive distinguisher between two binary random sources, but even finding tight bounds for non-adaptive distinguishers between two general random sources is still an open problem. Furthermore, discussing (and efficiently describing) more complicated dependences about the key is an open issue as

well, and we feel that the adaptive case is even more complicated; iterated attacks surely are an interesting intermediate case. In the hypothetical case where we would be able to precisely define a set of *sufficient* conditions for a block cipher to be secure, and this in a properly defined and practical security model, exhibiting a block cipher satisfying these properties looks like another non-trivial gap to fill. Hence, Vaudenay's *decorrelation theory*, which is nowadays the *only* general framework which proposes constructive solutions and which goes in the right direction, deserves, in our personal opinion, more attention.

List of Figures

List of Algorithms

Bibliography

[1] C. Adams. Constructing symmetric ciphers using the CAST design procedure. *Designs, Codes and Cryptography*, 12(3):283–316, 1997.

[2] C. Adams, H. Heys, S. Tavares, and M. Wiener. CAST256: a submission for the Advanced Encryption Standard. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[3] Advanced encryption standard (AES). Website `http://csrc.nist.gov/encryption/aes/`.

[4] G. Álvarez, D. de la Guía, F. Montoya, and A. Peinado. Akelarre: a new block cipher algorithm. In *SAC'96, Third Annual Workshop on Selected Areas in Cryptography, Queen's University, Kingston, Ontario, Canada. Workshop Record*, pages 1–14, 1996.

[5] H. Amirazizi and M. Hellman. Time-memory-processor tradeoffs. *IEEE Transactions on Information Theory*, 34(3):505–512, 1988.

[6] R. Anderson and E. Biham. Two practical and provably secure block ciphers: BEAR and LION. In D. Gollman, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996. Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 99–111. Springer-Verlag, 1996.

[7] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita. Camellia: a 128-bit block cipher suitable for multiple platforms - design and analysis. In D. Stinson and S. Tavares, editors, *Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000. Proceedings*, volume 2012 of *Lecture Notes in Computer Science*, pages 39–56. Springer-Verlag, 2001.

[8] K. Aoki, M. Kanda, T. Matsumoto, S. Moriai, K. Ohta, M. Ookubo, Y. Takashima, and H. Ueda. E2 – a candidate cipher for AES. First

AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[9] K. Aoki and H. Lipmaa. Fast implementation of AES candidates. Second AES Candidate Conference (AES2), Rome, Italy, March 22-23, 1999.

[10] K. Aoki and K. Ohta. Strict evaluation of the maximum average of differential probability and the maximum average of linear probability. *IEICE Transactions*, E80-A:1–8, 1997.

[11] K. Aoki and S. Vaudenay. On the use of GF-inversion as a cryptographic primitive. In *Selected Areas in Cryptography: 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 2003. Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*, pages 234–247. Springer-Verlag, 2004.

[12] ARM Ltd. Website `http://www.arm.com`.

[13] S. Babbage and L. Frisch. On Misty1 higher order differential cryptanalysis. In D. Won, editor, *Information Security and Cryptology – ICISC 2000: Third International Conference, Seoul, Korea, December 8-9, 2000. Proceedings*, volume 2015 of *Lecture Notes in Computer Science*, pages 22–36. Springer-Verlag, 2001.

[14] T. Baignères, P. Junod, and S. Vaudenay. How far can we go beyond linear cryptanalysis ? In P. Lee, editor, *Advances in Cryptology – Asiacrypt 2004: 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004. Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450. Springer-Verlag, 2004.

[15] T. Baignères. *A generalization of linear cryptanalysis*. Diploma thesis, École Polytechnique Fédérale de Lausanne (Switzerland), 2003.

[16] E. Barkan and E. Biham. In how many ways can you write Rijndael? In Y. Zheng, editor, *Advances in Cryptology – Asiacrypt 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002. Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 160–175. Springer-Verlag, 2002.

[17] P. Barreto and V. Rijmen. The Anubis block cipher. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[18] P. Barreto and V. Rijmen. The Khazad legacy-level block cipher. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[19] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science, October 20-22, 1997, Miami Beach, USA*, pages 393–403. IEEE, 1997.

[20] E. Biham. How to forge DES-encrypted messages in 2^{28} steps. Technical Report CS-0884, Computer Science Department, Technion, Haifa, Israel, 1996.

[21] E. Biham. A fast new DES implementation in software. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop, FSE'97, Haifa, Israel, January 1997. Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 260–272. Springer-Verlag, 1997.

[22] E. Biham. Cryptanalysis of Patarin's 2-round public-key cryptosystem with S-boxes (2R). In B. Preneel, editor, *Advances in Cryptology* – EUROCRYPT *2000: International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 2000. Proceedings*, volume 1807 of *Lecture Notes in Computer Science*, pages 408–416. Springer-Verlag, 2000.

[23] E. Biham, R. Anderson, and L. Knudsen. Serpent: a new block cipher proposal. In S. Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE'98, Paris, France, March 23-25, 1998. Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer-Verlag, 1998.

[24] E. Biham and A. Biryukov. An improvement of Davies' attack on DES. In A. De Santis, editor, *Advances in Cryptology* – EUROCRYPT *'94: Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 1994. Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 461–467. Springer-Verlag, 1995.

[25] E. Biham and A. Biryukov. An improvement of Davies' attack on DES. *Journal of Cryptology*, 10(3):195–205, 1997.

[26] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In J. Stern, editor, *Advances in Cryptology* – EUROCRYPT *'99: International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 1999. Proceedings*, volume 1592 of *Lecture Notes in Computer Science*, pages 12–23. Springer-Verlag, 1999.

[27] E. Biham, A. Biryukov, and A. Shamir. Miss-in-the-middle attacks on IDEA and Khufru. In L. Knudsen, editor, *Fast Software Encryption:*

6th International Workshop, FSE'99, Rome, Italy, March 1999. Proceedings, volume 1636 of *Lecture Notes in Computer Science*, pages 124–138. Springer-Verlag, 1999.

[28] E. Biham, O. Dunkelman, and N. Keller. The rectangle attack - rectangling the Serpent. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 2001. Proceedings*, volume 2045 of *Lecture Notes in Computer Science*, pages 340–357. Springer-Verlag, 2001.

[29] E. Biham, O. Dunkelman, and N. Keller. Enhancing differential-linear cryptanalysis. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002. Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 254–266. Springer-Verlag, 2002.

[30] E. Biham and N. Keller. Cryptanalysis of reduced variants of Rijndael. Third AES Candidate Conference (AES3), New-York, NY, USA, April 10-12, 2000.

[31] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems (extended abstract). In A. Menezes and S. Vanstone, editors, *Advances in Cryptology – CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990. Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer-Verlag, 1990.

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

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

[34] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In B. Kaliski, editor, *Advances in Cryptology – CRYPTO '97: 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 1997. Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1997.

[35] A. Biryukov, J. Nakahara, B. Preneel, and J. Vandewalle. New weak-key classes of IDEA. In R. Deng, S. Qing, F. Bao, and J. Zhou, editors, *Information and Communications Security: 4th International Conference, ICICS 2002, Singapore, December 9-12, 2002. Proceedings*, volume 2513 of *Lecture Notes in Computer Science*, pages 315–326. Springer-Verlag, 2002.

[36] A. Biryukov and A. Shamir. Structural cryptanalysis of SASAS. In B. Pfitzmann, editor, *Advances in Cryptology* – EUROCRYPT *2001: International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 2001. Proceedings*, volume 2045 of *Lecture Notes in Computer Science*, pages 394–405. Springer-Verlag, 2001.

[37] A. Biryukov and D. Wagner. Slide attacks. In L. Knudsen, editor, *Fast Software Encryption: 6th International Workshop, FSE'99, Rome, Italy, March 1999. Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer-Verlag, 1999.

[38] A. Biryukov and D. Wagner. Advanced slide attacks. In B. Preneel, editor, *Advances in Cryptology* – EUROCRYPT *2000: International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 2000. Proceedings*, volume 1807 of *Lecture Notes in Computer Science*, pages 589–606. Springer-Verlag, 2000.

[39] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport layer security (TLS) extensions. RFC 3546, 2003. Available on `http://www.ietf.org`.

[40] M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener. Minimal key lengths for symmetric ciphers to provide adequate commercial security, 1996. Available at `http://www.schneier.com/paper-keylength.html`.

[41] M. Blaze and B. Schneier. The MacGuffin cipher algorithm. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994. Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 97–110. Springer-Verlag, 1995.

[42] U. Blöcher and M. Dichtl. Problems with the linear cryptanalysis of DES using more than one active S-box per round. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994. Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 265–274. Springer-Verlag, 1995.

[43] Bluetooth$^{\text{tm}}$. *Bluetooth Specifications, version 1.2*, 2003. Available on `https://www.bluetooth.org`.

[44] J. Borst. The block cipher: GrandCru. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[45] J. Borst, L. Knudsen, and V. Rijmen. Two attacks on reduced IDEA (extended abstract). In W. Fumy, editor, *Advances in Cryptology – Eurocrypt'97: International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 1997. Proceedings*, volume 1233 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 1997.

[46] J. Borst, B. Preneel, and J. Vandewalle. Linear cryptanalysis of RC5 and RC6. In L. Knudsen, editor, *Fast Software Encryption: 6th International Workshop, FSE'99, Rome, Italy, March 1999. Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, 1999.

[47] E. Brickel, J. Moore, and M. Purtill. Structure in the S-boxes of DES. In A. Odlyzko, editor, *Advances in Cryptology – Crypto'86, Santa Barbara, California, USA, 1986. Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 3–7. Springer-Verlag, 1987.

[48] L. Brown. *Analysis of the DES and the design of the LOKI encryption scheme.* PhD thesis, Dept. Computer Science, UC UNSW, ADFA, Canberra, Australia, 1991.

[49] L. Brown, M. Kwan, J. Pieprzyk, and J. Seberry. Improving resistance to differential cryptanalysis and the redesign of LOKI. In H. Imai, R. Rivest, and Matsumoto, editors, *Advances in Cryptology – Asiacrypt'91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991. Proceedings*, volume 739 of *Lecture Notes in Computer Science*, pages 36–50. Springer-Verlag, 1993.

[50] L. Brown, J. Pieprzyk, and J. Seberry. LOKI - a cryptographic primitive for authentication and secrecy applications. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology – Auscrypt'90, International Conference on Cryptology, Sydney, Australia, January 8-11, 1990. Proceedings*, volume 453 of *Lecture Notes in Computer Science*, pages 229–236. Springer-Verlag, 1990.

[51] L. Brown, J. Pieprzyk, and J. Seberry. Introducing the new LOKI97 block cipher. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[52] L. Brown and J. Seberry. Key scheduling in DES type cryptosystems. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology – Auscrypt'90, International Conference on Cryptology, Sydney, Australia, January 8-11, 1990. Proceedings*, volume 453 of *Lecture Notes in Computer Science*, pages 221–228. Springer-Verlag, 1990.

[53] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. Matyas, L. O'Connor, M. Peyravian, D. Safford, and N. Zunic. Mars – a candidate cipher for AES. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[54] C. Cachin. An information-theoretic model for steganography. In D. Aucsmith, editor, *Information Hiding, Second International Workshop, Portland, Oregon, USA, April 14-17, 1998. Proceedings*, volume 1525 of *Lecture Notes in Computer Science*, pages 306–318. Springer-Verlag, 1998.

[55] C. Cachin. An information-theoretic model for steganography. Available on `http://eprint.iacr.org/2000/028/`, 2000.

[56] A. Canteaut and M. Videau. Degree of composition of highly nonlinear functions and applications. In L. Knudsen, editor, *Advances in Cryptology* – EUROCRYPT *2002: International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002. Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 518–533. Springer-Verlag, 2002.

[57] B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux. Password interception in a SSL/TLS channel. In D. Boneh, editor, *Advances in Cryptology* – CRYPTO *2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 583–599. Springer-Verlag, 2003.

[58] F. Chabaud and S. Vaudenay. Links between differential and linear cryptanalysis. In A. De Santis, editor, *Advances in Cryptology* – EUROCRYPT*'94: Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 1994. Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 356–365. Springer-Verlag, 1995.

[59] J. Cheon, M. Kim, K. Kim, J.-Y. Lee, and S. Kang. Improved impossible differential cryptanalysis of Rijndael and Crypton. In K. Kim, editor, *Information Security and Cryptology – ICISC 2001: 4th International Conference, Seoul, Korea, December 6-7, 2001. Proceedings*, volume 2288 of *Lecture Notes in Computer Science*, pages 39–49. Springer-Verlag, 2002.

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

[61] D. Coppersmith, S. Halevi, and C. Jutla. Cryptanalysis of stream ciphers with linear masking. In M. Yung, editor, *Advances in Cryptology – Crypto 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18-22, 2002. Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, 2002.

[62] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[63] Toshiba Corporation. Specification of Hierocrypt-3. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[64] Toshiba Corporation. Specification on a block cipher: Hierocrypt-L1. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[65] N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In D. Boneh, editor, *Advances in Cryptology – Crypto 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings*, volume 2729, pages 176–194. Springer-Verlag, 2003.

[66] N. Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt. In P. Lee and C. Lim, editors, *Information Security and Cryptology – ICISC 2002: 5th International Conference, Seoul, Korea, November 28-29, 2002. Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 182–199. Springer-Verlag, 2003.

[67] N. Courtois, G. Castagnos, and L. Goubin. What do DES S-boxes say to each other ? Available on `http://eprint.iacr.org/2003/184/`, 2003.

[68] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In B. Preneel, editor, *Advances in Cryptology – Eurocrypt 2000: International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 2000. Proceedings*, volume 1807 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[69] N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *Advances in Cryptology – Eurocrypt 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003.*

Proceedings, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.

[70] N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Y. Zheng, editor, *Advances in Cryptology* – ASIACRYPT *2002: 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002. Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer-Verlag, 2002.

[71] T. Cover and J. Thomas. *Information Theory*. Wiley Series in Telecommunications. Wiley, 1991.

[72] CRYPTREC Project. Website `http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html`.

[73] I. Csiszár and J. Körner. *Information theory: coding theorems for discrete memoryless systems*. Academic Press, 1981.

[74] T. Cusick. Boolean functions satisfying a higher order strict avalanche criterion. In T. Helleseth, editor, *Advances in Cryptology* – EUROCRYPT *'93: Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 1993. Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 102–117. Springer-Verlag, 1993.

[75] T. Cusick and M. Wood. The RedocII cryptosystem. In A. Menezes and S. Vanstone, editors, *Advances in Cryptology* – CRYPTO *'90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990. Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 545–563. Springer-Verlag, 1990.

[76] J. Daemen. *Cipher and hash function design strategies based on linear and differential cryptanalysis*. PhD thesis, K. U. Leuven, Belgium, 1995.

[77] J. Daemen, R. Govaerts, and J. Vandewalle. Weak keys for IDEA. In D. Stinson, editor, *Advances in Cryptology* – CRYPTO *'93: 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993. Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 224–231. Springer-Verlag, 1994.

[78] J. Daemen, M. Peeters, G. Van Assche, and V. Rijmen. The Noekeon block cipher. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[79] J. Daemen and V. Rijmen. The block cipher Rijndael. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[80] J. Daemen and V. Rijmen. The block cipher Rijndael. In J.-J. Quisquater and B. Schneier, editors, *Smart Card Research and Applications, Third International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998. Proceedings*, volume 1820 of *Lecture Notes in Computer Science*, pages 277–284. Springer-Verlag, 2000.

[81] J. Daemen and V. Rijmen. *The Design of Rijndael*. Information Security and Cryptography. Springer, 2002.

[82] J. Damen, L. Knudsen, and V. Rijmen. The block cipher SQUARE. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop, FSE'97, Haifa, Israel, January 1997. Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer-Verlag, 1997.

[83] D. Davies and S. Murphy. Pairs and triples of DES S-boxes. *Journal of Cryptology*, 8(1):1–25, 1995.

[84] H. Demirci. Square-like attacks on reduced rounds of IDEA. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 147–159. Springer-Verlag, 2003.

[85] H. Demirci, A. Selçuk, and E. Türe. A new meet-in-the-middle attack on the IDEA block cipher. In *Selected Areas in Cryptography: 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 2003. Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*, pages 117–129. Springer-Verlag, 2004.

[86] B. den Boer. Cryptoanalysis of FEAL. Presented at the rump session of Crypto'87, 1987.

[87] D. Denning. *Cryptography and data security*. Addison-Wesley, 1983.

[88] Y. Desmedt and J.-J. Quisquater. An exhaustive key search machine breaking one million DES keys. Presented during Eurocrypt'87, 1987.

[89] A. di Porto and W. Wolfovicz. VINO: a block cipher including variable permutations. In R. Anderson, editor, *Fast Software Encryption, Cambridge Security Workshop, Cambridge, UK, December 9-11, 1993.*

Proceedings, volume 809 of *Lecture Notes in Computer Science*, pages 205–210. Springer-Verlag, 1994.

[90] T. Dierks and C. Allen. The TLS protocol version 1.0. RFC 2246, 1999. Available at `http://www.ietf.org`.

[91] W. Diffie and M. Hellman. Multiuser cryptographic techniques. In *Proceedings of 1976 AFIPS National Computer Conference*, volume 45, pages 109–112. AFIPS Press, 1976.

[92] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[93] W. Diffie and M. Hellman. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10(6):74–84, 1977.

[94] W. Diffie and M. Hellman. Privacy and authentication: an introduction to cryptography. *Proceedings of IEEE*, 67(3):397–427, 1979.

[95] Project `distributed.net`. Website `http://www.distributed.net`.

[96] H. Eberle. A high-speed DES implementation for network applications. In E. Brickell, editor, *Advances in Cryptology* – CRYPTO *'92: 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992. Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 521–539. Springer-Verlag, 1993.

[97] Electronic Frontier Foundation (EFF). Website `http://www.eff.org`.

[98] Electronic Frontier Fondation. *Cracking DES*. O'Reilly, 1998.

[99] ETSI/SAGE. *Kasumi Specification, Part of the Specification of the 3GPP Confidentiality and Integrity Algorithms*, 1999. Available on `http://www.etsi.org`.

[100] S. Even and O. Goldreich. On the power of cascade ciphers. *ACM Transactions on Computer Systems*, 3(2):108–116, 1985.

[101] S. Even and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of Cryptology*, 10(3):151–162, 1997.

[102] H. Feistel. Cryptography and data security. *Scientific American*, 228(5):15–23, 1973.

[103] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting. Improved cryptanalysis of Rijndael. In B. Schneier, editor, *Fast Software Encryption: 7th International Workshop, FSE*

2000, New York, NY, USA, April 2000. Proceeding, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230. Springer-Verlag, 2001.

[104] N. Ferguson, R. Schroeppel, and D. Whiting. A simple algebraic representation of Rijndael. In S. Vaudenay and A. Youssef, editors, *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001, Toronto, Ontario, Canada, August 16-17, 2001. Revised Papers*, volume 2259 of *Lecture Notes in Computer Science*, pages 103–111. Springer-Verlag, 2001.

[105] A. Fiat and M. Naor. Rigorous time/space tradeoffs for inverting functions. In *Proceedings of the 23th Annual ACM Symposium on Theory of Computing*, pages 534–541. ACM Press, 1991.

[106] A. Fiat and M. Naor. Rigorous time/space tradeoffs for inverting functions. *SIAM Journal of Computing*, 29(3):790–803, 1999.

[107] R. Forré. The strict avalanche criterion: spectral properties of boolean functions and an extended definition. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO'88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988. Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 450–468. Springer-Verlag, 1990.

[108] J. Fuller and W. Millan. Linear redundancy in S-boxes. In T. Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003. Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[109] S. Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly and Associates, 1994.

[110] G. Garon and R. Outerbridge. DES watch: an examination of the sufficiency of the data encryption standard for financial institution information security in the 1990s. *SIGSAC Review*, 9(4):29–45, 1991.

[111] D. Georgoudis, D. Leroux, and B. Chaves. The "FROG" encryption algorithm. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[112] H. Gilbert. *Cryptanalyse statistique des algorithmes de chiffrement et sécurité des schémas d'authentification*. PhD thesis, Université de Paris-Sud, U.F.R. scientifique d'Orsay, 1997.

[113] H. Gilbert and G. Chassé. A statistical attack of the FEAL cryptosystem. In A. Menezes and S. Vanstone, editors, *Advances in Cryptology – Crypto'90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990. Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 22–33. Springer-Verlag, 1990.

[114] H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern, and S. Vaudenay. Decorrelated fast cipher: an AES candidate. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[115] H. Gilbert and M. Minier. A collision attack on seven rounds of Rijndael. Third AES Candidate Conference (AES3), New-York, NY, USA, April 10-12, 2000.

[116] Government Committee of the USSR for Standards. *GOST - Gosudarstvennyi Standard 28147-89, "Cryptographic protection for data processing systems"*, 1989.

[117] L. Granboulan. Flaws in differential cryptanalysis of Skipjack. In M. Matsui, editor, *Fast Software Encryption: 8th International Workshop, FSE 2001, Yokohama, Japan, April 2-4, 2001. Revised Papers*, volume 2355 of *Lecture Notes in Computer Science*, pages 328–335. Springer-Verlag, 2002.

[118] L. Granboulan, P. Nguyen, F. Noilhan, and S. Vaudenay. DFCv2. In D. Stinson and S. Tavares, editors, *Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000. Proceedings*, volume 2012 of *Lecture Notes in Computer Science*, pages 57–71. Springer-Verlag, 2001.

[119] G. Grimmett and D. Stirzacker. *Probability and random processes*. Oxford University Press, 2001.

[120] L. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings, 28th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 212–219. ACM Press, 1996. Also available on `http://arxiv.org/abs/quant-ph/9605043`.

[121] H. Handschuh and H. Gilbert. χ^2 cryptanalysis of the SEAL encryption algorithm. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop, FSE'97, Haifa, Israel, January 1997. Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1997.

[122] H. Handschuh and H. Heys. A timing attack on RC5. In S. Tavares and H. Meijer, editors, *Selected Areas in Cryptography: 5th Annual International Workshop, SAC'98, Kingston, Ontario, Canada, August 1998. Proceedings*, volume 1556 of *Lecture Notes in Computer Science*, pages 306–318. Springer-Verlag, 1999.

[123] H. Handschuh and D. Naccache. SHACAL. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[124] H. Handschuh and S. Vaudenay. A universal encryption standard. In H. Heys and C. Adams, editors, *Selected Areas in Cryptography: 6th Annual International Workshop, SAC'99, Kingston, Ontario, Canada, August 1999. Proceedings*, volume 1758 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 2000.

[125] C. Harpes, G. Kramer, and J. L. Massey. A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma. In L. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology – EUROCRYPT'95: International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 1995. Proceedings*, volume 921 of *Lecture Notes in Computer Science*, pages 24–38. Springer-Verlag, 1995.

[126] C. Harpes and J. Massey. Partitioning cryptanalysis. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop, FSE'97, Haifa, Israel, January 1997. Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 13–27. Springer-Verlag, 1997.

[127] Y. Hatano, H. Tanaka, and T. Kaneko. An optimized algebraic method for higher order differential attack. In M. Fossorier, T. Høholdt, and A. Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 2643 of *Lecture Notes in Computer Science*, pages 61–70. Springer-Verlag, 2003.

[128] P. Hawkes. Differential-linear weak key classes of IDEA. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May/June 1998. Proceedings*, volume 1403 of *Lecture Notes in Computer Science*, pages 112–126. Springer-Verlag, 1998.

[129] M. E. Hellman. A cryptanalytic time-memory tradeoff. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[130] H. Heys and S. Tavares. Avalanche characteristics of substitution-permutation networks. *IEEE Transactions on Computers*, 44(9):1131–1139, 1995.

[131] H. Heys and S. Tavares. Substitution-permutation networks resistant to differential and linear cryptanalysis. *Journal of Cryptology*, 9(1):1–19, 1996.

[132] S. Hong, S. Lee, J. Lim, J. Sung, D. Cheon, and I. Cho. Provable security against differential and linear cryptanalysis for the SPN structure. In B. Schneier, editor, *Fast Software Encryption: 7th International Workshop, FSE 2000, New York, NY, USA, April 2000. Proceeding*, volume 1978 of *Lecture Notes in Computer Science*, pages 273–283. Springer-Verlag, 2001.

[133] F. Hoornaert, J. Goubert, and Y. Desmedt. Efficient hardware implementation of the DES. In G. Blakley and D. Chaum, editors, *Advances in Cryptology, Proceedings of* CRYPTO *'84, Santa Barbara, California, USA, August 19-22, 1984. Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 147–173. Springer-Verlag, 1985.

[134] Y. Hu, Y. Zhang, and G. Xiao. Integral cryptanalysis of SAFER+. *Electronic Letters*, 35(17):1458–1459, August 1999.

[135] D. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers (IRE)*, 40:1098–1101, 1951.

[136] Intel corp. Website `http://www.intel.com`.

[137] ISO. *Information processing – modes of operation for an n-bit block cipher algorithm.* ISO/IEC 10116, International Organization for Standardization, Genève, Switzerland, 1991.

[138] M. Jacobson and K. Huber. The Magenta block cipher algorithm. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[139] T. Jakobsen. Cryptanalysis of block ciphers with probabilistic non-linear relations of low degree. In H. Krawczyk, editor, *Advances in Cryptology –* CRYPTO *'98: 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 1998. Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 212–222. Springer-Verlag, 1998.

[140] T. Jakobsen. *Higher-order cryptanalysis of block ciphers.* PhD thesis, Department of Mathematics, Technical University of Denmark, 1999.

[141] T. Jakobsen and C. Harpes. Non-uniformity measures for generalized linear cryptanalysis and partitioning cryptanalysis. In J. Pribyl, editor, PRAGOCRYPT *'96. Proceedings.* CTU Publishing House, 1996.

[142] T. Jakobsen and L. Knudsen. The interpolation attack against block ciphers. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop, FSE'97, Haifa, Israel, January 1997. Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40. Springer-Verlag, 1997.

[143] T. Jakobsen and L. Knudsen. Attacks on block ciphers of low algebraic degree. *Journal of Cryptology*, 14(3):197–210, 2001.

[144] P. Junod. On the complexity of Matsui's attack. In S. Vaudenay and A. Youssef, editors, *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001, Toronto, Ontario, Canada, August 16-17, 2001. Revised Papers*, volume 2259 of *Lecture Notes in Computer Science*, pages 199–211. Springer-Verlag, 2001.

[145] P. Junod. On the optimality of linear, differential and sequential distinguishers. In E. Biham, editor, *Advances in Cryptology* – EUROCRYPT *2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003. Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, 2003.

[146] P. Junod and S. Vaudenay. Device and a method for encrypting and decrypting a block of data. European patent application EP 03011696.6, 2003.

[147] P. Junod and S. Vaudenay. FOX specifications version 1.0. Technical Report EPFL/IC/2003/82, École Polytechnique Fédérale, Lausanne, Switzerland, 2003.

[148] P. Junod and S. Vaudenay. Method for generating pseudo-random sequences. European patent application EP 03103307.9, 2003.

[149] P. Junod and S. Vaudenay. Optimal key ranking procedures in a statistical cryptanalysis. In T. Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003. Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*, pages 235–246. Springer-Verlag, 2003.

[150] P. Junod and S. Vaudenay. FOX: a new family of block ciphers. In H. Handschuh and A. Hasan, editors, *Selected Areas in Cryptography: 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004. Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 114–129. Springer-Verlag, 2004.

[151] P. Junod and S. Vaudenay. FOX specifications version 1.1. Technical Report EPFL/IC/2004/75, École Polytechnique Fédérale, Lausanne, Switzerland, 2004.

[152] P. Junod and S. Vaudenay. Perfect diffusion primitives for block ciphers - building efficient MDS matrices. In H. Handschuh and A. Hasan, editors, *Selected Areas in Cryptography: 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004. Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 84–99. Springer-Verlag, 2004.

[153] B. Kaliski and M. Robshaw. Linear cryptanalysis using multiple approximations. In Y. Desmedt, editor, *Advances in Cryptology – Crypto'94: 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994. Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 26–39. Springer-Verlag, 1994.

[154] J. Kam and G. Davida. Structured design of substitution-permutation encryption networks. *IEEE Transactions on Computers*, 28(10):747, 1979.

[155] L. Keliher, H. Meijer, and S. Tavares. Improving the upper bound on the maximum average linear hull probability for Rijndael. In S. Vaudenay and A. Youssef, editors, *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001, Toronto, Ontario, Canada, August 16-17, 2001. Revised Papers*, volume 2259 of *Lecture Notes in Computer Science*, pages 112–128. Springer-Verlag, 2001.

[156] L. Keliher, H. Meijer, and S. Tavares. New method for upper bounding the maximum average linear hull probability for SPNs. In B. Pfitzmann, editor, *Advances in Cryptology – Eurocrypt 2001: International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 2001. Proceedings*, volume 2045 of *Lecture Notes in Computer Science*, pages 420–436. Springer-Verlag, 2001.

[157] L. Keliher, H. Meijer, and S. Tavares. Completion of improved upper bound on the maximum average linear hull probability for Rijndael. Available on `http://eprint.iacr.org/2004/074/`, 2004.

[158] J. Kelsey, T. Kohno, and B. Schneier. Amplified boomerang attacks against reduced-round MARS and Serpent. In B. Schneier, editor, *Fast Software Encryption: 7th International Workshop, FSE 2000, New York, NY, USA, April 2000. Proceeding*, volume 1978 of *Lecture Notes in Computer Science*, pages 75–93. Springer-Verlag, 2001.

[159] J. Kelsey, B. Schneier, and D. Wagner. Mod n cryptanalysis, with applications against RC5P and M6. In L. Knudsen, editor, *Fast Software Encryption: 6th International Workshop, FSE'99, Rome, Italy,*

March 1999. Proceedings, volume 1636 of *Lecture Notes in Computer Science*, pages 139–155. Springer-Verlag, 1999.

[160] J. Kilian and P. Rogaway. How to protect DES against exhaustive key search. *Journal of Cryptology*, 14(1):17–35, 2001.

[161] L. Knudsen. Truncated and higher order differentials. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994. Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, 1995.

[162] L. Knudsen. Contemporary block ciphers. In I. Damgård, editor, *Lectures on Data Security – Modern Cryptology in Theory and Practice*, volume 1561 of *Lecture Notes in Computer Science*, pages 105–126. Springer-Verlag, 1998.

[163] L. Knudsen. DEAL: a 128-bit block cipher. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[164] L. Knudsen. The security of Feistel ciphers with six rounds or less. *Journal of Cryptology*, 15(3):207–222, 2002.

[165] L. Knudsen and T. Berson. Truncated differentials of SAFER. In D. Gollman, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996. Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 15–26. Springer-Verlag, 1996.

[166] L. Knudsen and J. Mathiassen. A chosen-plaintext linear attack on DES. In B. Schneier, editor, *Fast Software Encryption: 7th International Workshop, FSE 2000, New York, NY, USA, April 2000. Proceeding*, volume 1978 of *Lecture Notes in Computer Science*, pages 262–272. Springer-Verlag, 2001.

[167] L. Knudsen and M. Robshaw. Non-linear approximations in linear cryptanalysis. In U. Maurer, editor, *Advances in Cryptology* – EUROCRYPT *'96: International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 1996. Proceedings*, volume 1070 of *Lecture Notes in Computer Science*, pages 224–236. Springer-Verlag, 1996.

[168] L. Knudsen, M. Robshaw, and D. Wagner. Truncated differentials and Skipjack. In M. Wiener, editor, *Advances in Cryptology* – EUROCRYPT *2002: International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002. Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 165–180. Springer-Verlag, 1999.

[169] L. Knudsen and D. Wagner. Integral cryptanalysis (extended abstract). In J. Daemen and V. Rijmen, editors, *Fast Software Encryption: 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002. Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer-Verlag, 2002.

[170] P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO'96: 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996. Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[171] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO'99: 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999. Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

[172] F. Koeune and J.-J. Quisquater. A timing attack against Rijndael. Technical Report CG-1999/1, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 1999.

[173] A. Konheim. Cryptography. Seminars of Excellence, ORSYS Institute, Amsterdam, June 9-11, 1986.

[174] Z. Kukorelly. The piling-up lemma and dependent random variables. In M. Walker, editor, *Cryptography and Coding, 7th IMA International Conference, Cirencester, UK, December 20-22, 1999. Proceedings*, volume 1746 of *Lecture Notes in Computer Science*, pages 186–190. Springer-Verlag, 1999.

[175] K. Kusuda and T. Matsumoto. Optimization of time-memory trade-off cryptanalysis and its application to DES, FEAL32 and Skipjack. *IEICE Transactions on Fundamentals*, E79-A(1):35–48, 1996.

[176] M. Kwan. Bitslice DES. Available at `http://www.darkside.com.au/bitslice/index.html`.

[177] M. Kwan. The design of the ICE encryption algorithm. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop, FSE'97, Haifa, Israel, January 1997. Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 69–82. Springer-Verlag, 1997.

[178] M. Kwan. Reducing the gate count of bitslice DES. Available on `http://eprint.iacr.org/2000/051/`, 2000.

[179] X. Lai. *On the design and security of block ciphers*, volume 1 of *ETH Series in Information Processing*. Hartung-Gorre Verlag, 1992.

[180] X. Lai. Higher order derivatives and differential cryptanalysis. In *Symposium on Communication, Coding and Cryptography*, pages 227–233. Kluwer Academic Publishers, 1994.

[181] X. Lai and J. Massey. A proposal for a new block encryption standard. In I. Damgård, editor, *Advances in Cryptology* – EUROCRYPT *'90: Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 1990. Proceedings*, volume 473 of *Lecture Notes in Computer Science*, pages 389–404. Springer-Verlag, 1991.

[182] X. Lai, J. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In D. Davies, editor, *Advances in Cryptology* – EUROCRYPT *'91: Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 1991. Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer-Verlag, 1991.

[183] K. Langford and E. Hellman. Differential-linear cryptanalysis. In Y. Desmedt, editor, *Advances in Cryptology* – CRYPTO *'94: 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994. Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 17–25. Springer-Verlag, 1994.

[184] A. Lebedev and A. Volchkov. NUSH. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[185] A. Lenstra and E. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

[186] C. Lim. Crypton: a new 128-bit block cipher. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[187] C. Lim. A revised version of Crypton – Crypton V1.0. In L. Knudsen, editor, *Fast Software Encryption: 6th International Workshop, FSE'99, Rome, Italy, March 1999. Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 31–45. Springer-Verlag, 1999.

[188] H. Lipmaa. Fast software implementations of SC2000. In A. Chan and V. Gligor, editors, *Information Security, 5th International Conference, ISC 2002 Sao Paulo, Brazil, September 30 - October 2, 2002. Proceedings*, volume 2433 of *Lecture Notes in Computer Science*, pages 63–74. Springer-Verlag, 2002.

[189] Y. Lu and S. Vaudenay. Cryptanalysis of Bluetooth keystream generator two-level E0. In P. Lee, editor, *Advances in Cryptology* – ASI-

ACRYPT *2004: 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004. Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 483–499. Springer-Verlag, 2004.

[190] Y. Lu and S. Vaudenay. Faster correlation attack on Bluetooth keystream generator E0. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004. Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 407–425. Springer-Verlag, 2004.

[191] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[192] S. Lucks. Faster Luby-Rackoff ciphers. In D. Gollman, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996. Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 189–203. Springer-Verlag, 1996.

[193] S. Lucks. Attacking triple encryption. In S. Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE'98, Paris, France, March 23-25, 1998. Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 189–205. Springer-Verlag, 1998.

[194] S. Lucks. Attacking seven rounds of Rijndael under 192- and 256-bit keys. Third AES Candidate Conference (AES3), New-York, NY, USA, April 10-12, 2000.

[195] S. Lucks. The saturation attack - a bait for Twofish. In M. Matsui, editor, *Fast Software Encryption: 8th International Workshop, FSE 2001, Yokohama, Japan, April 2-4, 2001. Revised Papers*, volume 2355 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2002.

[196] A. Machado. The Nimbus cipher: a proposal for NESSIE. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[197] F. MacWilliams and N. Sloane. *The theory of error-correcting codes.* North-Holland, 1977.

[198] W. Mao. *Modern cryptography – theory and practice.* Prentice-Hall, 2004.

[199] J. Massey. SAFER-K: a byte-oriented block-ciphering algorithm. In R. Anderson, editor, *Fast Software Encryption, Cambridge Security*

Workshop, Cambridge, UK, December 9-11, 1993. Proceedings, volume 809 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 1994.

[200] J. Massey, G. Khachatrian, and M. Kuregian. Nomination of SAFER+ as candidate algorithm for the Advanced Encryption Standard (AES). First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[201] J. Massey, G. Khachatrian, and M. Kuregian. Nomination of SAFER++ as candidate algorithm for the New European Schemes for Signatures, Integrity, and Encryption (NESSIE), 2000. First Open NESSIE Workshop, Leuven, Belgium, November 13-14.

[202] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology* – EUROCRYPT*'93: Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 1993. Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1993.

[203] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Y. Desmedt, editor, *Advances in Cryptology* – CRYPTO*'94: 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994. Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1994.

[204] M. Matsui. On correlation between the order of S-boxes and the strength of DES. In A. De Santis, editor, *Advances in Cryptology* – EUROCRYPT*'94: Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 1994. Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer-Verlag, 1995.

[205] M. Matsui. New block encryption algorithm MISTY. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop, FSE'97, Haifa, Israel, January 1997. Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 53–67. Springer-Verlag, 1997.

[206] M. Matsui and A. Yamagishi. A new method for known plaintext attack of FEAL cipher. In R. Rueppel, editor, *Advances in Cryptology* – EUROCRYPT*'92: Workshop on the Theory and Application of Cryptographic Techniques, Balatonfüred, Hungary, May 1992. Proceedings*, volume 658 of *Lecture Notes in Computer Science*, pages 81–91. Springer-Verlag, 1993.

[207] U. Maurer. A provably-secure strongly-randomized cipher. In I. Damgård, editor, *Advances in Cryptology* – EUROCRYPT *'90: Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 1990. Proceedings*, volume 473 of *Lecture Notes in Computer Science*, pages 361–373. Springer-Verlag, 1991.

[208] U. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[209] U. Maurer. A simplified and generalized treatment of Luby-Rackoff pseudorandom generators. In R. Rueppel, editor, *Advances in Cryptology* – EUROCRYPT *'92: Workshop on the Theory and Application of Cryptographic Techniques, Balatonfüred, Hungary, May 1992. Proceedings*, volume 658 of *Lecture Notes in Computer Science*, pages 239–255. Springer-Verlag, 1993.

[210] U. Maurer. A unified and generalized treatment of authentication theory. In C. Puech and R. Reischuk, editors, *STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996. Proceedings*, volume 1046 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.

[211] U. Maurer. Authentication theory and hypothesis testing. *IEEE Transactions on Information Theory*, 46(4):1350–1356, 2000.

[212] U. Maurer. Indistinguishability of random systems. In L. Knudsen, editor, *Advances in Cryptology* – EUROCRYPT *2002: International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002. Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 110–133. Springer-Verlag, 2002.

[213] U. Maurer and Pietrzak K. The security of many-round Luby-Rackoff pseudo-random permutations. In E. Biham, editor, *Advances in Cryptology* – EUROCRYPT *2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003. Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 544–561. Springer-Verlag, 2003.

[214] U. Maurer and K. Pietrzak. Composition of random systems: when two weak make one strong. In M. Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004. Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 410–427. Springer-Verlag, 2004.

[215] U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle

methodology. In M. Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004. Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer-Verlag, 2004.

[216] L. May, L. Penna, and A. Clark. An implementation of bitsliced DES on the Pentium MMX™ processor. In E. Dawson, A. Clark, and C. Boyd, editors, *Information Security and Privacy, 5th Australasian Conference, ACISP 2000, Brisbane, Australia, July 10-12, 2000. Proceedings*, volume 1841 of *Lecture Notes in Computer Science*, pages 112–122. Springer-Verlag, 2000.

[217] L. McBride. Q: a proposal for NESSIE. First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.

[218] MediaCrypt AG, Zürich, Switzerland. Website `http://www.mediacrypt.com`.

[219] W. Meier. On the security of the IDEA block cipher. In T. Helleseth, editor, *Advances in Cryptology* – EUROCRYPT *'93: Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 1993. Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 371–385. Springer-Verlag, 1993.

[220] W. Meier and O. Staffelbach. Nonlinearity criteria for cryptographic functions. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology* – EUROCRYPT *'89: Workshop on the Theory and Application of Cryptographic Techniques, Houthalen, Belgium, April 1989. Proceedings*, volume 434 of *Lecture Notes in Computer Science*, pages 549–562. Springer-Verlag, 1989.

[221] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of applied cryptography*. The CRC Press series on discrete mathematics and its applications. CRC-Press, 1997.

[222] R. Merkle. *Secrecy, authentication and public key systems*. UMI Research Press, 1979.

[223] R. Merkle. Fast software encryption functions. In A. Menezes and S. Vanstone, editors, *Advances in Cryptology* – CRYPTO *'90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990. Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 476–501. Springer-Verlag, 1990.

[224] R. Merkle and M. Hellman. On the security of multiple encryption. *Communications of the ACM*, 24:465–467, 1981.

[225] M. Minier and H. Gilbert. Stochastic cryptanalysis of Crypton. In B. Schneier, editor, *Fast Software Encryption: 7th International Workshop, FSE 2000, New York, NY, USA, April 2000. Proceeding*, volume 1978 of *Lecture Notes in Computer Science*, pages 121–133. Springer-Verlag, 2001.

[226] S. Miyaguchi. The FEAL-8 cryptosystem and a call for attack. In G. Brassard, editor, *Advances in Cryptology* – CRYPTO *'89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989. Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 624–627. Springer-Verlag, 1990.

[227] S. Miyaguchi. The FEAL cipher family. In A. Menezes and S. Vanstone, editors, *Advances in Cryptology* – CRYPTO *'90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990. Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 627–638. Springer-Verlag, 1990.

[228] J. Monnerat and S. Vaudenay. On some weak extensions of AES and BES. In J. Lopez, S. Qing, and E. Okamoto, editors, *Information and Communications Security, 6th International Conference, ICICS 2004, Malaga, Spain, October 27-29, 2004. Proceedings*, volume 3269 of *Lecture Notes in Computer Science*, pages 414–426. Springer-Verlag, 2004.

[229] S. Moriai, K. Aoki, and K. Ohta. The best linear expression search of FEAL. *IEICE Transactions*, E79-A(1):2–11, 1996.

[230] S. Moriai, T. Shimoyama, and T. Kaneko. Higher order differential attack of a CAST cipher. In S. Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE'98, Paris, France, March 23-25, 1998. Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 17–31. Springer-Verlag, 1998.

[231] S. Moriai and S. Vaudenay. On the pseudorandomness of top-level schemes of block ciphers. In T. Okamoto, editor, *Advances in Cryptology* – ASIACRYPT *2000: 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000. Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 289–302. Springer-Verlag, 2000.

[232] D. M'Raïhi, D. Naccache, J. Stern, and S. Vaudenay. XMX: A firmware-oriented block cipher based on modular multiplications. In E. Biham, editor, *Fast Software Encryption: 4th International Workshop, FSE'97, Haifa, Israel, January 1997. Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 166–171. Springer-Verlag, 1997.

[233] F. Muller. A new attack against Khazad. In C. Laih, editor, *Advances in Cryptology* – ASIACRYPT *2003: 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003. Proceedings*, volume 2894 of *Lecture Notes in Computer Science*, pages 347–358. Springer-Verlag, 2003.

[234] S. Murphy, F. Piper, M. Walker, and P. Wild. Likelihood estimation for block cipher keys. Technical report, Information Security Group, University of London, England, 1995.

[235] S. Murphy and M. Robshaw. Essential algebraic structure within the AES. In M. Yung, editor, *Advances in Cryptology* – CRYPTO *2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18-22, 2002. Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2002.

[236] S. Murphy and M. Robshaw. Comments on the security of the AES and the XSL technique. *Electronic Letters*, 39(1):36–38, 2003.

[237] J. Nakahara. *Cryptanalysis and design of block ciphers*. PhD thesis, Faculteit Toegepaste Wenschappen, Katholieke Universiteit Leuven (Belgium), 2003.

[238] J. Nakahara, P. Barreto, B. Preneel, J. Vandewalle, and Y. Kim. Square attacks on reduced-round PES and IDEA block ciphers. In B. Macq and J.-J. Quisquater, editors, *Proceedings of 23rd Symposium on Information Theory in the Benelux, Louvain-la-Neuve, Belgium, May 29-31, 2002*, pages 187–195, 2002.

[239] J. Nakahara, B. Preneel, and J. Vandewalle. The Biryukov-Demirci attack on IDEA and MESH ciphers. Technical report, COSIC, ESAT, Katholieke Universiteit Leuven, Leuven, Belgium, 2003.

[240] J. Nakahara, B. Preneel, and J. Vandewalle. The Biryukov-Demirci attack on reduced-round versions of IDEA and MESH block ciphers. In H. Wang, J. Pieprzyk, and V. Varadharajan, editors, *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*, volume 3108 of *Lecture Notes in Computer Science*, pages 98–109. Springer-Verlag, 2004.

[241] M. Naor and O. Reingold. On the construction of pseudorandom permutation: Luby-Rackoff revisited. *Journal of Cryptology*, 12(1):29–66, 1999.

[242] National Bureau of Standards, U. S. Department of Commerce. *Data Encryption Standard (DES)*, FIPS 46, 1977.

[243] National Bureau of Standards, U. S. Department of Commerce. *DES modes of operations*, FIPS 81, 1980.

[244] National Institute of Standards and Technology, U. S. Department of Commerce. *Skipjack and KEA algorithm specifications*, 1998. Available on `http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack.pdf`.

[245] National Institute of Standards and Technology, U. S. Department of Commerce. *Advanced Encryption Standard (AES)*, 2001.

[246] National Institute of Standards and Technology, U. S. Department of Commerce. *Data Encryption Standard*, NIST FIPS PUB 46-3, 1999.

[247] New European Schemes for Signatures, Integrity, and Encryption (NESSIE). Website `https://www.cryptonessie.org`.

[248] J. Neyman and E. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions A of the Royal Society*, 231:289–337, 1933.

[249] K. Nyberg. Perfect nonlinear S-boxes. In D. Davies, editor, *Advances in Cryptology* – EUROCRYPT *'91: Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 1991. Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 378–386. Springer-Verlag, 1991.

[250] K. Nyberg. Linear approximation of block ciphers. In A. De Santis, editor, *Advances in Cryptology* – EUROCRYPT *'94: Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 1994. Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 439–444. Springer-Verlag, 1995.

[251] K. Nyberg and L. Knudsen. Provable security against a differential cryptanalysis. In Y. Desmedt, editor, *Advances in Cryptology* – CRYPTO *'94: 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994. Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 566–574. Springer-Verlag, 1994.

[252] K. Nyberg and L. Knudsen. Provable security against a differential attack. *Journal of Cryptology*, 8(1):27–38, 1995.

[253] L. O'Connor. Enumerating nondegenerate permutations. In D. Davies, editor, *Advances in Cryptology* – EUROCRYPT *'91: Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 1991. Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 368–377. Springer-Verlag, 1991.

[254] L. O'Connor. On the distribution of characteristics in bijective mappings. In T. Helleseth, editor, *Advances in Cryptology* – EUROCRYPT *'93: Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 1993. Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 360–370. Springer-Verlag, 1993.

[255] L. O'Connor. On the distribution of characteristics in composite permutations. In D. Stinson, editor, *Advances in Cryptology* – CRYPTO *'93: 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993. Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 403–412. Springer-Verlag, 1994.

[256] Ph. Oechslin. Making a faster cryptanalytic time-memory tradeoff. In D. Boneh, editor, *Advances in Cryptology* – CRYPTO *2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer-Verlag, 2003.

[257] S. Park, H. Sung, S. Chee, E.-J. Yoon, and J. Lim. On the security of Rijndael-like structures against differential and linear cryptanalysis. In Y. Zheng, editor, *Advances in Cryptology* – ASIACRYPT *2002: 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002. Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 176–191. Springer-Verlag, 2002.

[258] S. Park, H. Sung, S. Lee, and J. Lim. Improving the upper bound on the maximum differential and maximum linear hull probability for SPN structures and AES. In T. Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003. Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[259] M. Parker. Generalized S-Box linearity. Technical report, Technical report `nes/doc/uib/wp5/020/a`, NESSIE Project, 2003. Available on `https://www.cryptonessie.org`.

[260] J. Patarin. How to construct pseudorandom permutations from a single pseudorandom function. In R. Rueppel, editor, *Advances in Cryptology* – EUROCRYPT *'92: Workshop on the Theory and Application of Cryptographic Techniques, Balatonfüred, Hungary, May 1992. Proceedings*, volume 658 of *Lecture Notes in Computer Science*, pages 256–266. Springer-Verlag, 1993.

[261] J. Patarin. About Feistel schemes with six (or more) rounds. In S. Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE'98, Paris, France, March 23-25, 1998. Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 103–121. Springer-Verlag, 1998.

[262] J. Patel and C. Read. *Handbook of the normal distribution*. Statistics: textbooks and monographs. Marcel Dekker, 1996.

[263] S. Patel, Z. Ramzan, and G. Sundaram. Towards making Luby-Rackoff ciphers optimal and practical. In L. Knudsen, editor, *Fast Software Encryption: 6th International Workshop, FSE'99, Rome, Italy, March 1999. Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 171–185. Springer-Verlag, 1999.

[264] R. Phan. Impossible differential cryptanalysis of 7-rounds Advanced Encryption Standard (AES). *Information Processing Letters*, 91(1):33–38, 2004.

[265] J. Pieprzyk. How to construct pseudorandom permutations from single pseudorandom functions. In I. Damgård, editor, *Advances in Cryptology* – EUROCRYPT *'90: Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 1990. Proceedings*, volume 473 of *Lecture Notes in Computer Science*, pages 140–150. Springer-Verlag, 1991.

[266] J. Pieprzyk and G. Finkelstein. Towards effective non-linear cryptosystem design. *IEE Proceedings, Part E*, 135(6):325–335, 1988.

[267] T. Pornin. Optimal resistance against the Davies and Murphy attack. In K. Ohta and D. Pei, editors, *Advances in Cryptology* – ASIACRYPT *'98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18-22, 1998. Proceedings*, volume 1514 of *Lecture Notes in Computer Science*, pages 148–159. Springer-Verlag, 1998.

[268] T. Pornin. *Implantation et optimisation des primitives cryptographiques*. PhD thesis, Département d'informatique, École Normale Supérieure, Paris (France), 2001.

[269] B. Preneel, A. Biryukov, E. Oswald, B. Van Rompay, L. Granboulan, E. Dottax, S. Murphy, A. Dent, J. White, M. Dichtl, S. Pyka, M. Schafheutle, P. Serf, E. Biham, E. Barkan, O. Dunkelman, J.-J. Quisquater, M. Ciet, F. Sica, L. Knudsen, M. Parker, and H. Raddum. NESSIE Security Report v2.0. Technical Report

NES/DOC/ENS/WP5/D20/2, New European Schemes for Signatures, Integrity, and Encryption (NESSIE), 2003. Available at https://www.cryptonessie.org.

[270] M. Raff. On approximating the point binomial. *Journal of the American Statistical Association*, 51:292–303, 1956.

[271] Z. Ramzan and L. Reyzin. On the round security of symmetric-key cryptographic primitives. In M. Bellare, editor, *Advances in Cryptology – Crypto 2000: 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 2000. Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 376–393. Springer-Verlag, 2000.

[272] B. Reichardt and D. Wagner. Markov truncated differential cryptanalysis of Skipjack. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 110–128. Springer-Verlag, 2003.

[273] A. Rényi. *Probability Theory*. Elsevier, 1970.

[274] J. Rice. *Mathematical statistics and data analysis*. Duxbury Press, 1995.

[275] V. Rijmen, J. Daemen, B. Preneel, A. Bossalaers, and E. De Win. The cipher Shark. In D. Gollman, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996. Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 99–111. Springer-Verlag, 1996.

[276] V. Rijmen, B. Preneel, and E. De Win. On weaknesses of non-surjective round functions. *Designs, Codes and Cryptography*, 12(3):253–266, 1997.

[277] R. Rivest. The RC5 encryption algorithm. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994. Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 86–96. Springer-Verlag, 1995.

[278] R. Rivest. A description of the RC2(r) encryption algorithm. RFC 2268, 1998. Available on http://www.ietf.org.

[279] R. Rivest, M. Robshaw, R. Sidney, and Y. Yin. The RC6 block cipher. v1.1. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[280] O. Rothaus. On bent functions. *Journal of Combinatorial Theory*, A20:300–305, 1976.

[281] RSA Security. Website `http://www.rsasecurity.com`.

[282] I. Schaumüller-Bichl. Cryptanalysis of the Data Encryption Standard by the method of formal coding. In T. Beth, editor, *Cryptography: Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany, March 29 - April 2, 1982*, volume 149 of *Lecture Notes in Computer Science*, pages 235–255. Springer-Verlag, 1983.

[283] B. Schneier. *Applied cryptography: protocols, algorithms and source code in C*. John Wiley and Sons, 1994.

[284] B. Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In R. Anderson, editor, *Fast Software Encryption, Cambridge Security Workshop, Cambridge, UK, December 9-11, 1993. Proceedings*, volume 809 of *Lecture Notes in Computer Science*, pages 191–204. Springer-Verlag, 1994.

[285] B. Schneier and J. Kelsey. Unbalanced Feistel networks and block cipher design. In D. Gollman, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996. Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 121–144. Springer-Verlag, 1996.

[286] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Twofish: A 128-bit block cipher. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[287] C. Schnorr and S. Vaudenay. Black box cryptanalysis of hash networks based on multipermutations. In A. De Santis, editor, *Advances in Cryptology* – EUROCRYPT *'94: Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 1994. Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 47–57. Springer-Verlag, 1995.

[288] R. Schroeppel and H. Orman. Overview of the hasty pudding cipher. First AES Candidate Conference (AES1), Ventura, California, USA, August 20-22, 1998.

[289] A. Selçuk and Bıçak. On probability of success in linear and differential cryptanalysis. In S. Cimato, C. Galdi, and G. Persiano, editors, *Security in Communication Networks: Third International Conference,* SCN *2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, volume 2576 of *Lecture Notes in Computer Science*, pages 174–185. Springer-Verlag, 2003.

[290] A. Selcuk. New results in linear cryptanalysis of RC5. In S. Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE'98, Paris, France, March 23-25, 1998. Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 1998.

[291] A. Selcuk. On bias estimation in linear cryptanalysis. In B. Roy and E. Okamoto, editors, *Progress in Cryptology* - INDOCRYPT *2000: First International Conference in Cryptology in India, Calcutta, India, December 2000. Proceedings*, volume 1977 of *Lecture Notes in Computer Science*, pages 52–66. Springer-Verlag, 2000.

[292] A. Shamir. On the security of DES. In H. Williams, editor, *Advances in Cryptology* – CRYPTO *'85, Santa Barbara, California, USA, August 18-22, 1985. Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 280–281. Springer-Verlag, 1986.

[293] A. Shamir and A. Kipnis. Cryptanalysis of the HFE public-key cryptosystem. In M. Wiener, editor, *Advances in Cryptology* – CRYPTO *'99: 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999. Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 19–30. Springer-Verlag, 1999.

[294] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, 1948.

[295] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[296] A. Shimizu and S. Miyaguchi. Fast data encipherment algorithm FEAL. In D. Chaum and W. Price, editors, *Advances in Cryptology* – EUROCRYPT *'87: Workshop on the Theory and Application of Cryptographic Techniques, Amsterdam, The Netherlands, April 1987. Proceedings*, volume 304 of *Lecture Notes in Computer Science*, pages 267–278. Springer-Verlag, 1988.

[297] T. Shimoyama and T. Kaneko. Quadratic relation of S-box and its application to the linear attack of full round DES. In H. Krawczyk, editor, *Advances in Cryptology* – CRYPTO *'98: 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 1998. Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 200–211. Springer-Verlag, 1998.

[298] T. Shimoyama, H. Yanami, K. Yokoyama, M. Takenaka, K. Itoh, J. Yajima, N. Torii, and H. Tanaka. Specification and supporting document

of the block cipher SC2000, 2000. First Open NESSIE Workshop, Leuven, Belgium, November 13-14.

[299] D. Siegmund. *Sequential Analysis - Tests and Confidence Intervals*. Springer-Verlag, 1985.

[300] M. Sivabalan, S. Tavares, and L. Peppard. On the design of SP networks from an information theoretic point of view. In E. Brickell, editor, *Advances in Cryptology – CRYPTO'92: 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992. Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 260–279. Springer-Verlag, 1993.

[301] J. Smith. The design of Lucifer: a cryptographic device for data communications. Technical report, IBM T.J. Watson Research Center, Yorktown Heights, N.Y., USA, 1971.

[302] F.-X. Standaert. *Secure and efficient use of reconfigurable hardware devices in symmetric cryptography*. PhD thesis, Faculté des sciences appliquées, Université Catholique de Louvain, Belgium, 2004.

[303] F.-X. Standaert, G. Rouvroy, G. Piret, J.-J Quisquater, and D. Legat. Key-dependent approximations in cryptanalysis: an application of multiple Z4 and non-linear approximations. In *24th Symposium on Information Theory in the Benelux*, 2003.

[304] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. A time-memory tradeoff using distinguished points: New analysis & FPGA results. In B. Kaliski, Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002: 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002. Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 593–610. Springer-Verlag, 2002.

[305] J. Stern and S. Vaudenay. CS-Cipher. In S. Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE'98, Paris, France, March 23-25, 1998. Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 189–205. Springer-Verlag, 1998.

[306] H. Tanaka, K. Hisamatsu, and T. Kaneko. Strength of Misty1 without FL functions for higher order differential attack. In M. Fossorier, H. Imai, S. Lin, and A. Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 13th International Symposium, AAECC-13, Honolulu, Hawaii, USA, November 15-19, 1999. Proceedings*, volume 1719 of *Lecture Notes in Computer Science*, pages 221–230. Springer-Verlag, 1999.

[307] H. Tanaka, K. Hisamatsu, and T. Kaneko. On the strength of KASUMI without FL functions against higher order differential attack. In D. Won, editor, *Information Security and Cryptology – ICISC 2000: Third International Conference, Seoul, Korea, December 8-9, 2000. Proceedings*, volume 2015 of *Lecture Notes in Computer Science*, pages 14–21. Springer-Verlag, 2001.

[308] A. Tardy-Corfdir and H. Gilbert. A known plaintext attack of FEAL and FEAL-6. In J. Feigenbaum, editor, *Advances in Cryptology – Crypto'91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991. Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 172–182. Springer-Verlag, 1991.

[309] P. van Oorschot and M. Wiener. A known-plaintext attack on two-key triple encryption. In I. Damgård, editor, *Advances in Cryptology – Eurocrypt'90: Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 1990. Proceedings*, volume 473 of *Lecture Notes in Computer Science*, pages 318–325. Springer-Verlag, 1991.

[310] P. van Oorschot and M. Wiener. Improving implementable meet-in-the-middle attacks by orders of magnitude. In N. Koblitz, editor, *Advances in Cryptology – Crypto'96: 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996. Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 229–236. Springer-Verlag, 1996.

[311] S. Vaudenay. On the need for multipermutations: cryptanalysis of MD4 and SAFER. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994. Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 286–297. Springer-Verlag, 1995.

[312] S. Vaudenay. An experiment on DES statistical cryptanalysis. In *3rd ACM Conference on Computer and Communications Security*, pages 139–147. ACM Press, 1996.

[313] S. Vaudenay. Provable security for block ciphers by decorrelation. In M. Morvan, C. Meinel, and D. Krob, editors, *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science, Paris, France, February 25-27, 1998. Proceedings*, volume 1373 of *Lecture Notes in Computer Science*, pages 249–275. Springer-Verlag, 1998. Invited talk.

[314] S. Vaudenay. Feistel ciphers with L_2-decorrelation. In S. Tavares and H. Meijers, editors, *Selected Areas in Cryptography: 5th Annual*

International Workshop, SAC'98, Kingston, Ontario, Canada, August 1998. Proceedings, volume 1556 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 1999.

[315] S. Vaudenay. On the security of CS-cipher. In L. Knudsen, editor, *Fast Software Encryption: 6th International Workshop, FSE'99, Rome, Italy, March 1999. Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 260–274. Springer-Verlag, 1999.

[316] S. Vaudenay. Resistance against general iterated attacks. In J. Stern, editor, *Advances in Cryptology* – EUROCRYPT *'99: International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 1999. Proceedings*, volume 1592 of *Lecture Notes in Computer Science*, pages 255–271. Springer-Verlag, 1999.

[317] S. Vaudenay. On the Lai-Massey scheme. In K. Lam , T. Okamoto, and C. Xing, editors, *Advances in Cryptology* – ASIACRYPT *'99: International Conference on the Theory and Application of Cryptology and Information Security, Singapore, November 14-18, 1999. Proceedings*, volume 1716 of *Lecture Notes in Computer Science*, pages 8–19. Springer-Verlag, 2000.

[318] S. Vaudenay. Adaptive-attack norm for decorrelation and super-pseudorandomness. In D. Stinson and S. Tavares, editors, *Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, Waterloo, Ontario, Canada, August 14-15, 2000. Proceedings*, volume 2012 of *Lecture Notes in Computer Science*, pages 49–61. Springer-Verlag, 2001.

[319] S. Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS ... In L. Knudsen, editor, *Advances in Cryptology* – EUROCRYPT *2002: International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002. Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546. Springer-Verlag, 2002.

[320] S. Vaudenay. Decorrelation: a theory for block cipher security. *Journal of Cryptology*, 16(4):249–286, 2003.

[321] G. Vernam. Secret signaling system. US Patent #1310719, 1926.

[322] D. Wagner. The boomerang attack. In L. Knudsen, editor, *Fast Software Encryption: 6th International Workshop, FSE'99, Rome, Italy, March 1999. Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer-Verlag, 1999.

[323] D. Wagner. Towards a unifying view of block cipher cryptanalysis. In B. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004. Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 16–33. Springer-Verlag, 2004.

[324] A. Wald. *Sequential Analysis*. John Wiley and Sons, New-York, 1947.

[325] P. Wayner. Content-addressable search engines and DES-like systems. In E. Brickell, editor, *Advances in Cryptology – CRYPTO '92: 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992. Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 575–586. Springer-Verlag, 1993.

[326] A. Webster and S. Tavares. On the design of S-boxes. In H. Williams, editor, *Advances in Cryptology – CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985. Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 523–534. Springer-Verlag, 1986.

[327] D. Wheeler and R. Needham. TEA, a tiny encryption algorithm. In B. Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994. Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 363–366. Springer-Verlag, 1995.

[328] M. Wiener. Efficient DES key search. Technical Report TR-244, School of Computer Science, Carleton University, Ottawa, Canada, 1994. Presented at the rump session of CRYPTO'93.

[329] M. Wiener. Efficient DES key search. In W. Stallings, editor, *Practical cryptography for data internetworks*, pages 31–79. IEEE Computer Society Press, 1996.

[330] M. Wiener. Efficient DES key search – an update. *CryptoBytes*, 3(2):6–8, 1997.

[331] R. Winternitz and M. Hellman. Chosen-key attacks on a block cipher. *Cryptologia*, 11(1):16–20, 1987.

[332] H. Wu. Related-cipher attacks. In R. Deng, S. Qing, F. Bao, and J. Zhou, editors, *Information and Communications Security: 4th International Conference, ICICS 2002, Singapore, December 9-12, 2002. Proceedings*, volume 2513 of *Lecture Notes in Computer Science*, pages 447–455. Springer-Verlag, 2002.

[333] A. Youssef and S. Tavares. Linear approximations of injective S-boxes. *Electronic Letters*, 31(25):2165–2166, 1995.

[334] A. Youssef and S. Tavares. Number of nonlinear regular S-boxes. *Electronic Letters*, 31(19):1643–1644, 1995.

[335] A. Youssef and S. Tavares. Resistance of balanced S-boxes to linear and differential cryptanalysis. *Information Processing Letters*, 56:249–252, 1995.

[336] Y. Zheng, T. Matsumoto, and H. Imai. Impossibility and optimality results on constructing pseudorandom permutations. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology* – EUROCRYPT*'89: Workshop on the Theory and Application of Cryptographic Techniques, Houthalen, Belgium, April 1989. Proceedings*, volume 434 of *Lecture Notes in Computer Science*, pages 412–422. Springer-Verlag, 1989.

Appendix A

Probability Theory

In this thesis, we make a heavy use of results taken out of general probability theory; we recall now basic concepts thereof and useful notations, mostly according to [119].

A.1 Preliminaries

The set of all possible outcomes of an experiment is called the *sample space* and we will denote it by $\mathcal{S}$. An *event* ω is then a subset $\omega \in \mathcal{S}$ of $\mathcal{S}$. This drives us to define the concept of σ*-field*:

Definition A.1.1 (σ-field). *A sequence $\mathcal{F}$ of subsets of $\mathcal{S}$ is called a* σ-field *if it satisfies the following conditions:*

1. $\emptyset \in \mathcal{F}$;
2. *if* $\omega_1, \omega_2, \cdots \in \mathcal{F}$, *then* $\bigcup_{i=1}^{\infty} \omega_i \in \mathcal{F}$;
3. *if* $\omega \in \mathcal{F}$, *then* $\overline{\omega} \in \mathcal{F}$.

In order to be able to discuss the *likelihood* of occurence of events, we need to define properly a *probability measure*.

Definition A.1.2 (Probability Measure / Probability Space). *Let $\mathcal{S}$ be a sample space and $\mathcal{F}$ be a σ-field of subsets of $\mathcal{S}$. A* probability measure $\Pr$ *on $(\mathcal{S}, \mathcal{F})$ is a function $\Pr : \mathcal{F} \to [0,1]$ satisfying*

1. $\Pr[\emptyset] = 0$ *and* $\Pr[\mathcal{S}] = 1$;
2. *if $\omega_1, \omega_2, \ldots$ is a sequence of disjoint members of $\mathcal{F}$, i.e. $\omega_i \cap \omega_j = \emptyset$ for $i \neq j$, then*

$$\Pr\left[\bigcup_{i=1}^{\infty} \omega_i\right] = \sum_{i=1}^{\infty} \Pr[\omega_i]$$

The triple $(\mathcal{S}, \mathcal{F}, \Pr)$ *is called a* probability space.

Two events ω_1 and ω_2 are called *independent* if $\Pr[\omega_1 \cap \omega_2] = \Pr[\omega_1] \cdot \Pr[\omega_2]$. More generally, a family $\{\omega_i : i \in \mathcal{I}\}$ is called independent if

$$\Pr\left[\bigcap_{i\in\mathcal{J}} \omega_i\right] = \prod_{i\in\mathcal{J}} \Pr[\omega_i]$$

for all finite subsets $\mathcal{J}$ of $\mathcal{I}$. The *conditional probability of an event* ω_1 *given a event* ω_2 is then defined, provided $\Pr[\omega_2] > 0$, as

$$\Pr[\omega_1|\omega_2] = \frac{\Pr[\omega_1 \cap \omega_2]}{\Pr[\omega_2]}$$

One is not always interested in an experiment itself, but rather in some consequence of its random outcome. Such consequences, if real valued, may be thought of as function which map $\mathcal{S}$ into the real line $\mathbb{R}$. These functions are called *random variables*.

Definition A.1.3 (Random Variable / Distribution Function). *A* random variable X *is a function* $X : \mathcal{S} \to \mathbb{R}$ *with the property that* $\{s \in \mathcal{S} : X(s) \leq x\} \in \mathcal{F}$ *for each* $x \in \mathbb{R}$. *The* distribution function *of a random variable* X *is the function* $\mathsf{F}_X : \mathbb{R} \to [0, 1]$ *defined by*

$$\mathsf{F}_X(x) = \Pr_X[X \leq x]$$

In this thesis, we will frequently make use of *discrete* and *continuous* random variables. A random variable X is called *discrete* is it takes values in some countable subset of $\mathbb{R}$. The distribution function of a discrete random variable X is called a *probability mass function*. A random variable X is called *continous* if its distribution function can be expressed as

$$\mathsf{F}_X(x) = \int_{-\infty}^{x} f(t)\, dt \qquad x \in \mathbb{R}$$

for some integrable function $\mathsf{f} : \mathbb{R} \to [0, \infty[$ called the *probability density function* of X. Furthermore, one can generalize the concept of distribution function of a random variable to the one of *joint distribution function of a vector* $\boldsymbol{X}$: let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a random vector of n random variables. The joint distribution function of $\boldsymbol{X}$ on the probability space $(\mathcal{S}, \mathcal{F}, \Pr)$ is the function $\mathsf{F}_{\boldsymbol{X}} : \mathbb{R}^n \to [0, 1]$ defined by

$$\mathsf{F}_{\boldsymbol{X}}(\boldsymbol{x}) = \Pr_{\boldsymbol{X}}[\boldsymbol{X} \leq \boldsymbol{x}]$$

for $\boldsymbol{x} \in \mathbb{R}^n$. An important characteristic of a random variable is its *expectation* and its k-th moment.

Definition A.1.4 (Expectation/Moment). *The* expectation *of a discrete random variable X with probability mass function f_X is defined to be*

$$\mathrm{E}[X] = \sum_{x:\mathsf{f}_X(x)>0} x \cdot \mathsf{f}_X(x)$$

while the expectation of a continuous random variable with probability density function f_X is defined to be

$$\mathrm{E}[X] = \int_{-\infty}^{+\infty} x \cdot \mathsf{f}_X(x)\, dx$$

*If k is a positive integer, the k-*th moment *m_k of a random variable X is defined to be $m_k = \mathrm{E}[X^k]$; the k-*th central moment *σ_k of X is defined to be $\sigma_k = \mathrm{E}[(X - m_1)^k]$.*

In this thesis, we will mostly use the expectation and the second central moment of a probability distribution, the latter being called the *variance*.

A.2 Common Probability Distributions

Several probability distributions play a central role in our thesis. We define them formally and we state some of their properties.

Definition A.2.1 (Bernoulli Distribution). *A discrete random variable X is said to be distributed according to the* Bernoulli distribution $\mathsf{D}^{(p)}_{\mathrm{Bern}}$ *if it can take only two values, namely 0 or 1, with respective probabilities*

$$\begin{aligned} \Pr_{\mathsf{D}^{(p)}_{\mathrm{Bern}}}[X = 0] &= 1 - p \\ \Pr_{\mathsf{D}^{(p)}_{\mathrm{Bern}}}[X = 1] &= p. \end{aligned}$$

Definition A.2.2 (Binomial Distribution). *A discrete random variable X is said to be distributed according to the* binomial distribution $\mathsf{D}^{(n,p)}_{\mathrm{Bin}}$ *if it can take values $0 \leq k \leq n$ with probability*

$$\Pr_{\mathsf{D}^{(n,p)}_{\mathrm{Bin}}}[X = k] = \binom{n}{k} p^k (1-p)^{n-k}.$$

Definition A.2.3 (Multinomial Distribution). *Let $\boldsymbol{X}$ be a discrete r-dimensional random vector whose i-th component is denoted X_i. $\boldsymbol{X}$ is said to be distributed according to the* multinomial distribution $\mathsf{D}^{(n,\boldsymbol{p})}_{\mathrm{Multi}}$ *if it can take values $\boldsymbol{x} \in \mathbb{N}^r$ with probability*

$$\Pr_{\mathsf{D}^{(n,\boldsymbol{p})}_{\mathrm{Multi}}}[\boldsymbol{X} = \boldsymbol{x}] = \binom{n}{x_1! \quad \dots \quad x_r!} p_1^{x_1} \cdots p_r^{x_r}$$

under the constraints

$$\sum_{i=1}^{r} x_i = n \qquad \textit{and} \qquad \sum_{i=1}^{r} p_i = 1.$$

Definition A.2.4 (Normal Distribution). *A continuous random variable X is said to be distributed according to the* normal distribution D_ϕ *if it take values $x \in \mathbb{R}$ with probability*

$$\Pr_{\mathsf{D}_\phi}[\alpha \leq X \leq \beta] = \frac{1}{\sqrt{2\pi}} \int_\alpha^\beta e^{-\frac{t^2}{2}}\, dt.$$

Similarly, a continuous random variable X is said to be distributed according to the normal distribution $\mathsf{D}_\phi^{(\mu,\sigma)}$ with parameters μ and σ *if the random variable*

$$\frac{X-\mu}{\sigma}$$

is distributed according to D_ϕ.

We denote respectively the probability density function and the cumulative distribution function of the normal distribution by

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \text{ and } \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt.$$

Finally, the following result obviously plays a central role in this thesis.

Theorem A.2.1 (Central-Limit Theorem). *Let $X_1, \ldots, X_n$ be a sequence of mutually independent random variable with a common probability distribution. Suppose that $\mu = \mathrm{E}[X_i]$ and $\sigma^2 = \mathrm{Var}[X_i]$ exist and let $S_n = X_1 + \cdots + X_n$. Then for every fixed ξ,*

$$\lim_{n\to\infty} \Pr\left[\frac{S_n - n\mu}{\sigma\sqrt{n}} < \xi\right] = \Phi(\xi)$$

Appendix B

Statistical Information Theory

In this chapter, we recall some well-known results about Csiszár and Körner's *method of types* [73] and we recall how to apply it to derive Chernoff's information. We closely follow the organization of Chapter 12 in [71].

B.1 The Method of Types

Let us consider a discrete alphabet $\mathcal{X} = \{a_1, \ldots, a_{|\mathcal{X}|}\}$. The *type* $\mathsf{D}_{\boldsymbol{x}}$ (or *empirical probability distribution*) of a sequence $\boldsymbol{x} = (x_1, \ldots, x_n)$ of n symbols with $x_i \in \mathcal{X}$ for all $i \in \{1, \ldots, n\}$, is defined to be the relative proportion of occurrences of each symbol of $\mathcal{X}$, i.e.

$$\forall a \in \mathcal{X}, \quad \Pr_{\mathsf{D}_{\boldsymbol{x}}}[a] = \frac{N(a|\boldsymbol{x})}{n}$$

where $N(a|\boldsymbol{x})$ is the number of times the symbol a occurs in the sequence $\boldsymbol{x} \in \mathcal{X}^n$. We denote by $\mathcal{P}_n$ the set of types with denominator n. If $\mathsf{D}_P \in \mathcal{P}_n$, then the set of sequences of length n and type D_P is called the *type class of* D_P, and is noted $\mathcal{T}(\mathsf{D}_P)$, i.e.

$$\mathcal{T}(\mathsf{D}_P) = \{\boldsymbol{x} \in \mathcal{X}^n : \mathsf{D}_{\boldsymbol{x}} = \mathsf{D}_P\}$$

The essential power of the method of types arises from the following result, which shows that the number of types is at most polynomial in n.

Theorem B.1.1.

$$|\mathcal{P}_n| \leq (n+1)^{|\mathcal{X}|} \tag{B.1}$$

From this point, we will assume that the sequence $X_1, \ldots, X_n$ is drawn independently and identically distributed according to a distribution D_P. Let us define the *Kullback-Leibler distance*.

Definition B.1.1 (Kullback-Leibler Distance). *The* Kullback-Leibler distance $D(\mathsf{D}_P||\mathsf{D}_Q)$ *between two discrete probability distributions* D_P *and* D_Q *is defined to be*

$$D(\mathsf{D}_P||\mathsf{D}_Q) = \sum_{x\in\mathcal{X}} \Pr_P[x] \log_2 \left(\frac{\Pr_P[x]}{\Pr_Q[x]}\right).$$

Coming back to considerations, we note that all sequences with the same type have the same probability, as stated by the following theorem.

Theorem B.1.2. *If* $X_1, \ldots, X_n$ *are drawn iid according to* D_P*, then the probability of* $\boldsymbol{x}$ *depends only on its type and is given by*

$$\Pr_{P^n}[\boldsymbol{x}] = \prod_{i=1}^{n} \Pr_P[x_i] = 2^{-n(H(\boldsymbol{x})+D(\mathsf{D}_{\boldsymbol{x}}||\mathsf{D}_P))} \tag{B.2}$$

where $H(\boldsymbol{x})$ *is the* entropy[1] *of* $\boldsymbol{x}$ *and* $D(\mathsf{D}_{\boldsymbol{x}}||\mathsf{D}_P)$ *is the* Kullback-Leibler distance *between the distributions* $\mathsf{D}_{\boldsymbol{x}}$ *and* D_P*.*

The following theorem allows to give useful bounds on the size of a type class.

Theorem B.1.3. *For any* $\mathsf{D}_P \in \mathcal{P}_n$*,*

$$\frac{1}{(n+1)^{|\mathcal{X}|}} 2^{nH(\mathsf{D}_P)} \leq |\mathcal{T}(\mathsf{D}_P)| \leq 2^{nH(\mathsf{D}_P)} \tag{B.3}$$

With help of Theorem B.1.3, it is possible to prove the following result.

Theorem B.1.4. *For any* $\mathsf{D}_P \in \mathcal{P}_n$*, and any distribution* D_Q*, the probability of the type class* $\mathcal{T}(\mathsf{D}_P)$ *under* D_{Q^n} *satisfies*

$$\frac{1}{(n+1)^{|\mathcal{X}|}} 2^{-nD(\mathsf{D}_P||\mathsf{D}_Q)} \leq \Pr_{Q^n}[\mathcal{T}(\mathsf{D}_P)] \leq 2^{-nD(\mathsf{D}_P||\mathsf{D}_Q)} \tag{B.4}$$

B.2 Sanov's Theorem

The method of types and above summarized results can be used to show Sanov's Theorem (see Th. B.2.1). We recall first some notions of topology. A family τ of subsets of a set $\mathcal{X}$ is a *topology* of $\emptyset \in \tau$, if $\mathcal{X} \in \tau$, if any union of sets of τ belongs to τ, and if any finite intersection of elements of τ belongs to τ. Sets that belongs to τ are called *open sets*, while complements of open sets are called *closed sets*. The *interior* of a subset $\mathcal{A} \subset \mathcal{X}$ is the union of the open subsets of $\mathcal{A}$. The *closure of* $\mathcal{A}$, is the intersection of all closed sets containing $\mathcal{A}$.

[1] The *entropy* of a discrete random variable X distributed according to D_X is defined by $H(X) = -\sum_{x\in\mathcal{X}} \Pr_X[x] \log_2(\Pr_X[x])$.

Theorem B.2.1 (Sanov). *Let $X_1, \ldots, X_n$ be n iid random variables distributed according D_Q. Let $\mathcal{E} \subseteq \mathcal{P}_n$ be a set of probability distributions. Then*

$$\Pr_{Q^n}[\mathcal{E}] = \Pr_{Q^n}[\mathcal{E} \cap \mathcal{P}_n] \leq (n+1)^{|\mathcal{X}|} 2^{-nD(\mathsf{D}_{P^*}||\mathsf{D}_Q)} \tag{B.5}$$

where

$$\mathsf{D}_{P^*} = \arg\min_{\mathsf{D}_P \in \mathcal{E}} D(\mathsf{D}_P||\mathsf{D}_Q) \tag{B.6}$$

is the distribution in $\mathcal{E}$ that is closest to D_Q in relative entropy. If, in addition, the set $\mathcal{E}$ is the closure of its interior, then

$$\lim_{n\to+\infty} \frac{1}{n} \log \Pr_{Q^n}[\mathcal{E}] = -D(\mathsf{D}_{P^*}||\mathsf{D}_Q) \tag{B.7}$$

B.3 Chernoff's Information

We recall now the derivation of the highest achievable exponent for the probability of error of an optimal decision region when sampling n times the same random variable. From Lem. 3.1.1, we know that the optimum test is a likelihood-ratio test. We can rewrite this ratio as

$$\frac{\Pr_{X_0^n}[\boldsymbol{x}]}{\Pr_{X_1^n}[\boldsymbol{x}]} \geq \tau \iff D(\mathsf{D}_{\boldsymbol{x}}||\mathsf{D}_{X_1^n}) - D(\mathsf{D}_{\boldsymbol{x}}||\mathsf{D}_{X_0^n}) \geq \frac{1}{n}\log\tau$$

or, in other words, it is possible to rewrite the log-likelihood ratio as the difference between th relative entropy distance of the sample type to each of the two possible distributions. Let $\mathcal{A}$ denote the set on which hypothesis $\boldsymbol{x} \leftarrow \mathsf{D}_{X_0^n}$ is accepted. Then, since the set $\overline{\mathcal{A}}$ is convex, one can use Theorem B.2.1 to show that the error probability

$$\alpha^{(n)} = \Pr_{X_0^n}[\boldsymbol{x} \in \overline{\mathcal{A}}] \tag{B.8}$$

is essentially determined by the relative entropy of the closest member $\mathsf{D}_{X_0^*}$ of $\overline{\mathcal{A}}$ to D_{X_0}:

$$\lim_{n\to\infty} \frac{1}{n} \log \frac{\alpha^{(n)}}{2^{-nD(\mathsf{D}_{X_0^*}||\mathsf{D}_{X_0})}} = 0 \tag{B.9}$$

Similarly,

$$\lim_{n\to\infty} \frac{1}{n} \log \frac{\beta^{(n)}}{2^{-nD(\mathsf{D}_{X_1^*}||\mathsf{D}_{X_1})}} = 0 \tag{B.10}$$

where $\beta^{(n)} = \Pr_{X_1^n}[\boldsymbol{x} \in \mathcal{A}]$ and $\mathsf{D}_{X_1^*}$ is the closest element in $\mathcal{A}$ to distribution D_{X_1}. Now, minimizing $D(\mathsf{D}_X||\mathsf{D}_{X_1})$ subject to the constraint

$$D(\mathsf{D}_X||\mathsf{D}_{X_1}) - D(\mathsf{D}_X||\mathsf{D}_{X_0}) \geq \frac{1}{n}\log\tau \tag{B.11}$$

will result in the type in $\mathcal{A}$ that is closest to D_{X_1}. Setting up the minimization of D_{X_1} subject to $D(\mathsf{D}_X||\mathsf{D}_{X_1}) - D(\mathsf{D}_X||\mathsf{D}_{X_0}) = \frac{1}{n}\log\tau$ using Lagrange multipliers, we obtain that the minimizing D_X is of the form

$$\Pr_{X_1^*}[x] = \Pr_{\lambda^*}[x] = \frac{\Pr_{X_0}[x]^\lambda \Pr_{X_1}[x]^{1-\lambda}}{\sum_{a\in\mathcal{X}} \Pr_{X_0}[a]^\lambda \Pr_{X_1}[a]^{1-\lambda}} \tag{B.12}$$

where λ is chosen so that $D(\mathsf{D}_{X_{\lambda^*}}||\mathsf{D}_{X_0}) - D(\mathsf{D}_{X_{\lambda^*}}||\mathsf{D}_{X_1}) = \frac{\log\tau}{n}$. Furthermore, from the symmetry of the above equation, we have $\mathsf{D}_{X_0^*} = \mathsf{D}_{X_1^*}$.

We come back to our decision problem. In the Bayesian case, the overall probability of error is the weighted sum of the two probabilities of error, and we have

$$\lim_{n\to+\infty} \frac{1}{n} \log \frac{\pi_0\alpha^{(n)} + \pi_1\beta^{(n)}}{2^{-n\min\{D(\mathsf{D}_{X_\lambda}||\mathsf{D}_{X_0}), D(\mathsf{D}_{X_\lambda}||\mathsf{D}_{X_1})\}}} \tag{B.13}$$

where D_{X_λ} has the form of (B.12). Since $D(\mathsf{D}_{X_\lambda}||\mathsf{D}_{X_0})$ increases with λ and $D(\mathsf{D}_{X_\lambda}||\mathsf{D}_{X_1})$ decreases with λ, the maximum value of

$$\min\{D(\mathsf{D}_{X_\lambda}||\mathsf{D}_{X_0}), D(\mathsf{D}_{X_\lambda}||\mathsf{D}_{X_1})\} \tag{B.14}$$

is attained when they are equal. So choosing λ such that

$$D(\mathsf{D}_{X_\lambda}||\mathsf{D}_{X_0}) = D(\mathsf{D}_{X_\lambda}||\mathsf{D}_{X_1}) = C(\mathsf{D}_{X_0}, \mathsf{D}_{X_1}) \tag{B.15}$$

yields the highest achievable exponent for the probability error and is called the *Chernoff's information*.

Curriculum Vitæ

I was born in Bienne (Switzerland) in 1976. After having attended primary school in Alle and secondary school in Porrentruy, I obtained a scientific baccalaureate ("*maturité fédérale type C*") at the Lycée Cantonal of Porrentruy. In 1995, I began computer science studies at the Swiss Federal Institute of Technology (ETHZ) in Zurich with a strong emphasize in cryptology, information theory and theoretical computer science; I graduated in 2000 with a diploma thesis entitled "*Linear Cryptanalysis of DES*". During my studies and just before arriving in Lausanne, I shortly worked for Europay (Switzerland) AG where I was dealing with security issues in electronic payment systems. Since october 2000, I have been a full-time research and teaching assistant, as well as a PhD student under the supervision of Prof. Serge Vaudenay, in the Security and Cryptography Laboratory (LASEC) at the Swiss Federal Institute of Technology (EPFL) in Lausanne.